

Core MW Management Guide

Core Middleware

USER GUIDE

Copyright

© Ericsson AB 2010–2015. All rights reserved. No part of this document may be reproduced in any form without the written permission of the copyright owner.

Disclaimer

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

Trademark List

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



Contents

1	Introduction	1
1.1	Prerequisites	1
2	Core MW Command Overview	3
2.1	Deprecated Command	5
3	Administrative Operations	7
3.1	Reboot Cluster	7
3.2	Reboot Node	8
3.3	Lock Node	8
3.4	Unlock Node	9
3.5	Verify System Status	9
3.6	Save IMM Data	13
3.7	Get AMF Node of Given Hostname	14
3.8	Get Hostname of Given AMF Node	14
3.9	Configure IMM Access Control	14
3.10	Enable or Disable Core MW Features	14
4	Core MW Partial Backup and Partial Restore – Deprecated	19
4.1	Create Core MW Partial Backup – Deprecated	19
4.2	List Core MW Partial Backups – Deprecated	20
4.3	Delete Core MW Partial Backup – Deprecated	20
4.4	Restore Using Core MW Partial Backup – Deprecated	20
5	Software Management	23
5.1	Import Software Bundles and Campaigns	23
5.2	Delete Software Bundles and Campaigns	24
5.3	Read from Software Inventory	24
5.4	Use SMF Campaigns	25
5.5	Configure ECIM SWM	31
5.6	Configure ISP Events for Cluster Restarts	32
5.7	Configure Reboot Behavior of Single-Step Procedures	33
6	Performance Management	35
6.1	Create ECIM PM Jobs	35



6.2	Start ECIM PM Jobs	39
6.3	Stop ECIM PM Jobs	39
6.4	Delete ECIM PM Jobs	40
6.5	Report Status of ECIM PM Jobs	41
6.6	List ECIM PM Jobs	41



1 Introduction

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

1.1 Prerequisites

This section states the prerequisites that must be fulfilled.

1.1.1 Conditions

The following is required:

- The system must be ready to accept logon attempts from users.
- In general commands are run as root user or prefixed with `sudo` and run by user belonging to `system-adm` group. The `sudo` configuration enforces the user to provide the password before to the execution of the command. Any exception 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 Command Overview

Core MW Commands	Description
<code>cmw-amfnode-get <hostname></code>	Get the AMF node of given <code>hostname</code> . See Section 3.7 Get AMF Node of Given Hostname on page 14.
<code>cmw-campaign-commit <campaign-name></code>	Commit campaign. See Section 5.4.1 Execute an SMF Campaign on page 27.
<code>cmw-campaign-rollback <campaign-name></code>	Rollback campaign. See Section 5.4.2 Roll Back a Campaign on page 28.
<code>cmw-campaign-start [--disable-backup] <campaign-name></code>	<p>Start campaign. See Section 5.4.1 Execute an SMF Campaign on page 27.</p> <p>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.</p>



Table 1 Core MW Command Overview

Core MW Commands	Description
<code>cmw-campaign-status <campaign-name></code>	Show campaign status. See Section 5.4.1 Execute an SMF Campaign on page 27.
<code>cmw-campaign-stop <campaign-name></code>	Suspend campaign. See Section 5.4.1 Execute an SMF Campaign on page 27.
<code>cmw-campaign-verify <campaign-name></code>	Verify campaign. See Section 5.4.6 Initiate SMF Verification on page 30.
<code>cmw-cluster-reboot [--yes]</code>	Reboot cluster. See Section 3.1 Reboot Cluster on page 7.
<code>cmw-hostname-get <amfnode></code>	Get hostname of given AMF node. See Section 3.8 Get Hostname of Given AMF Node on page 14.
<code>cmw-node-lock <hostname> [-t <timeout> --timeout <timeout>]</code>	Lock node. See Section 3.3 Lock Node on page 8.
<code>cmw-node-unlock <hostname> [-t <timeout> --timeout <timeout>]</code>	Unlock node. See Section 3.4 Unlock Node on page 9.
<code>cmw-node-reboot [<hostname>]</code>	Reboot single node. See Section 3.2 Reboot Node on page 8.
<code>cmw-pmjob-create <job_name> <option>... [<option>...]</code>	Create an ECIM PM Job. See Section 6.1 Create ECIM PM Jobs on page 35.
<code>cmw-pmjob-start <job_name></code>	Start an ECIM PM Job. See Section 6.2 Start ECIM PM Jobs on page 39.
<code>cmw-pmjob-stop <job_name></code>	Stop an ECIM PM Job. See Section 6.3 Stop ECIM PM Jobs on page 39.
<code>cmw-pmjob-delete <job_name></code>	Delete an ECIM PM Job. See Section 6.4 Delete ECIM PM Jobs on page 40.
<code>cmw-pmjob-status [-v] <job_name></code>	Show ECIM PM Job status. See Section 6.5 Report Status of ECIM PM Jobs on page 41.
<code>cmw-pmjob-list [<option>...]</code>	List all defined ECIM PM Jobs. See Section 6.6 List ECIM PM Jobs on page 41.
<code>cmw-repository-list [--campaign] [--no de] [<hostname>...]</code>	List imported software bundles and campaigns. See Section 5.3 Read from Software Inventory on page 24.
<code>cmw-sdp-import <file> [<file>...]</code>	Import software bundles and campaigns. See Section 5.1 Import Software Bundles and Campaigns on page 23.



Table 1 Core MW Command Overview

Core MW Commands	Description
<code>cmw-sdp-remove <name> [<name>...]</code>	Delete software bundles and campaigns. See Section 5.2 Delete Software Bundles and Campaigns on page 24.
<code>cmw-status [-v] <class-name> [<class-name>...]</code>	Show the system status. Only failing items are printed unless the <code>-v</code> flag is specified. See Section 3.5 Verify System Status on page 9.
<code>cmw-swm-config-set <option> [<option>...]</code>	Set the installation-dependent attributes in the ECIM SWM object. See Section 5.5 Configure ECIM SWM on page 31.
<code>cmw-imm-policy-set <option></code>	Configure IMM access control. For more information, see Section 3.9 Configure IMM Access Control on page 14.
<code>cmw-node-alarm-timeout</code>	Set the 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 should have finished. The housekeeping is performed after a restore when scaling is enabled (=the time that the system can have a faulty state after a restore).
<code>cmw-configuration <option></code>	Enable or disable Core MW features. For more information, see Section 3.10 Enable or Disable Core MW Features on page 14.

2.1 Deprecated Command

The deprecated commands are shown in Table 2.

Table 2 Deprecated Command

Core MW Command	Description
<code>cmw-immSave</code>	This command is deprecated and cannot be used. Using this command only result in the logging of a warning message. The IMM data is, starting with Core MW R2A, persistently saved automatically.

*Table 2 Deprecated Command*

Core MW Command	Description
<code>cmw-partial-backup-create [--verbose] <label></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 19.</p>
<code>cmw-partial-backup-remove [--verbose] <label></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 20.</p>
<code>cmw-partial-backup-restore [--verbose] <label></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 20.</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 20.</p>



3 Administrative Operations

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

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

This section describes the following administrative operations:

- Rebooting cluster
- Rebooting node
- Locking node
- Unlocking node
- Verifying system status
- Saving IMM data
- Getting Application Management Framework (AMF) node given hostname
- Getting hostname given AMF node
- Configuring IMM Access Control

3.1 Reboot Cluster

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

```
cmw-cluster-reboot [--yes]
```

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

A cluster reboot with confirmation is shown in Example 1.



```

SC-1:~ # cmw-cluster-reboot
Really want to reboot the entire cluster (yes/no)? yes
Rebooting all nodes
Stopping DHCP daemon on node 2 (SC-2)
Stopping DHCP daemon ..done
Stopping DHCP daemon on node 1 (SC-1)
Stopping DHCP daemon ..done
Rebooting node 4 (PL-4)
Rebooting node 3 (PL-3)
Waiting for payload nodes to shut down
Payload nodes have shut down
Rebooting node 2 (SC-2)
Rebooting node 1 (SC-1)

Broadcast message from root (pts/0) (Mon Jul  5 13:07:36 2010):

The system is going down for reboot NOW!
SC-1:~ #

```

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 explicitly 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 impact on the system.

A node reboot is shown in Example 2.

```

SC-1:~ # cmw-node-reboot PL-4
Rebooting node 4 (PL-4)
SC-1:~ #

```

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 hosted on the node are prohibited from providing service.

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

```
-t, --timeout <sec>
```

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.

How to lock a node is shown in Example 3.

```
SC-1:~ # cmw-node-lock PL-4
SC-1:~ #
```

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 service units hosted on the node are administratively allowed to provide service.

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

```
-t, --timeout <sec>
```

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

How to unlock a node is shown in Example 4.

```
SC-1:~ # cmw-node-unlock PL-4
SC-1:~ #
```

Example 4 Unlocking Node

3.5 Verify System Status

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

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

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

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



comp	An AMF <code>comp</code> has a HA state that must be verified. No <code>comp</code> must be in <code>quiescing</code> , or <code>quiesced</code> state when the system status is “normal”. The operational state of <code>comp</code> must be enabled.
node	An executing instance of the Host OS connected to the cluster. A node can be a virtual machine.
pm	Performance Management (PM) reports overall Ericsson Common Information Model (ECIM) PM Status. If all the ECIM PM jobs are in <code>Active</code> state then it returns <code>Status OK</code> . Otherwise it provides details about ECIM PM Jobs that are in <code>Stopped</code> or <code>Faulty State</code> . The command has a short and verbose format. The short format lists only the ECIM PM job name and state for <code>Stopped</code> or <code>Faulty</code> Jobs. The verbose format lists all ECIM PM jobs, including those in <code>Active</code> states.

**sg**

A service group 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 service group are used to provide service availability. Supported redundancy models are as follows:

- 2N

In this redundancy model one SU is active for all SIs, and one SU is standby for all SIs.

- 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.

- 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.

- N-WayActive

This is a load-sharing redundancy model with only active service units.

- 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.

si

As Components are aggregated into SUs, the AMF supports the aggregation of CSIs into a logical entity called an 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.

siass

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.

**su**

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.

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.

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

```
SC-1:~ # cmw-status node comp su
Status OK
SC-1:~ #
```

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 when there is no fault in the system.

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

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

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.

```
SC-1:~ # cmw-status -v si
safSi=SC-2N,safApp=OpenSAF
AdminState=UNLOCKED(1)
AssignmentState=PARTIALLY_ASSIGNED(3)
safSi=SC-2N,safApp=ERIC-CoreMW
AdminState=UNLOCKED(1)
AssignmentState=PARTIALLY_ASSIGNED(3)
SC-1:~ #
```

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

When verifying that node status and one node is locked, the output can look as shown in Example 8.

```
SC-1:~ # cmw-status node
safAmfNode=PL-3,safAmfCluster=myAmfCluster
AdminState=LOCKED-INSTANTIATION(3)
OperState=ENABLED(1)
SC-1:~ #
```

Example 8 *System with One Node Locked*



When using verbose mode for node status, the output can look as shown in Example 9.

```
SC-1:~ # cmw-status -v node
safAmfNode=SC-2,safAmfCluster=myAmfCluster
  AdminState=UNLOCKED(1)
  OperState=ENABLED(1)
safAmfNode=SC-1,safAmfCluster=myAmfCluster
  AdminState=UNLOCKED(1)
  OperState=ENABLED(1)
safAmfNode=PL-4,safAmfCluster=myAmfCluster
  AdminState=UNLOCKED(1)
  OperState=ENABLED(1)
safAmfNode=PL-3,safAmfCluster=myAmfCluster
  AdminState=UNLOCKED(1)
  OperState=ENABLED(1)
SC-1:~ #
```

Example 9 Verifying System with Verbose Mode

When verifying PM Job status, with one job in state `Stopped`, the output can look as shown in Example 10.

```
SC-1:~ # cmw-status pm
pmJobId=TC-CMW-PM-CMD-STATUS-1, Stopped
SC-1:~ #
```

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

When verifying PM Job status, using verbose mode, the output can look as shown in Example 11.

```
SC-1:~ # cmw-status -v pm
Name                               Value(s)
=====
pmJobId                            TC-CMW-PM-CMD-STATUS-1
jobType                            Measurement(0x1)
granularityPeriod                   1 minute(0x3)
reportingPeriod                     1 minute(0x3)
jobPriority                         High(0x3)
requestedJobState                   Stopped(0x2)
currentJobState                     Stopped(0x2)
Overall Job State                   Stopped(0x2)
Name                               Value(s)
=====
pmJobId                            TC-CMW-PM-CMD-STATUS-2
jobType                            Threshold(0x2)
granularityPeriod                   5 minutes(0x4)
reportingPeriod                     15 minutes(0x5)
jobPriority                         Medium(0x2)
requestedJobState                   Active(0x1)
currentJobState                     Active(0x1)
Overall Job State                   Active(0x1)
SC-1:~ #
```

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

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

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

An example of the output is shown in Example 12.

```
SC-1:~ # cmw-amfnode-get SC-1  
SC-1  
SC-1:~ #
```

Example 12 Getting AMF Node 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>
```

An example of the output is shown in Example 13.

```
SC-1:~ # cmw-hostname-get SC-1  
SC-1  
SC-1:~ #
```

Example 13 Getting Hostname Given AMF Node

3.9 Configure IMM Access Control

IMM Access Control is configured by 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 need to 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

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



```
cmw-configuration <--enable / --disable / --status> <FEATURE>
[--reboot]
```

The Core MW features are the following:

- --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 the <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.

Description:

The following features are supported:

- TIPC_MULTICAST
- SC_ABSENCE_ALLOWED
- SCALING

For more details on each feature, see respective subsection.

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

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

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



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

```
SC-1:~ # cmw-configuration --enable TIPC_MULTICAST --reboot
Rebooting cluster ...
```

Example 14 Enable TIPC Multicast with Forced Reboot

```
SC-1:~ # cmw-configuration --enable TIPC_MULTICAST
Cluster reboot is required for change to take effect
```

Example 15 Enable TIPC Multicast without Forced Reboot

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

```
SC-1:~ # cmw-configuration --enable TIPC_MULTICAST --reboot
Already enable
```

Example 16 Enable TIPC Multicast When Already Enabled

The status of TIPC Multicast can be checked, as shown in Example 17.

```
SC-1:~ # cmw-configuration --status TIPC_MULTICAST
Enable
```

Example 17 TIPC Multicast Status Check

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 SC Absence in Core MW, the user must verify that the current 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 option `--reboot` parameter.

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

If SC Absence 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.

```
SC-1:~ # cmw-configuration --enable SC_ABSENCE_ALLOWED 300 --reboot
Rebooting cluster ...
```

Example 18 Enable SC Absence with Time Out of 300 Seconds and Forced Reboot



```
SC-1:~ # cmw-configuration --disable SC_ABSENCE_ALLOWED
Cluster reboot is required for change to take effect
```

Example 19 Disable SC Absence without Forced Reboot

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

```
SC-1:~ # cmw-configuration --status SC_ABSENCE_ALLOWED
WARNING: Configuration has been changed but cluster reboot isn't called
Enable
Current timeout is 300
```

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

3.10.3

SCALING

When scaling is enabled, the system can automatically be expanded with newly detected nodes, using a default scaling profile. The CRM NBI model allows the removal of scalable nodes which triggers a resize of the system. The scaling feature is enabled in run-time and does not require a reboot.

The Example 21 shows how to enable scaling.

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

Example 21 Enable SCALING

The Example 22 shows how to check the status of SCALING if it is already enabled.

```
SC-1:~ # cmw-configuration --status SCALING
Enable
```

Example 22 Check Status of SCALING

Note: It is only possible to enable and check the status of scaling feature.





4 Core MW Partial Backup and Partial Restore – Deprecated

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

This section describes different partial backup and partial restore operations.

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 be used to restore a system in most cases 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 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

Note: This section contains information that is **deprecated** since Core MW R6A

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



How to create a Core MW partial backup is shown in Example 23.

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

Example 23 Creating Core MW Partial Backup

4.2 List Core MW Partial Backups – Deprecated

Note: This section contains information that is **deprecated** since Core MW R6A

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.

How to list a Core MW partial backup label is shown in Example 24.

```
SC-1:~ # cmw-partial-backup-list
1st-CMW
mgmtug-example
SMF-BACKUP_2010-07-05T12:48:08
SMF-BACKUP_2010-07-05T12:58:56
SC-1:~ #
```

Example 24 Listing Core MW Partial Backup Labels

4.3 Delete Core MW Partial Backup – Deprecated

Note: This section contains information that is **deprecated** since Core MW R6A

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

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

How to delete a Core MW partial backup is shown in Example 25.

```
SC-1:~ # cmw-partial-backup-remove mgmtug-example
SC-1:~ #
```

Example 25 Deleting Core MW Partial Backup

4.4 Restore Using Core MW Partial Backup – Deprecated

Note: This section contains information that is **deprecated** since Core MW R6A



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 7.

How to restore using Core MW partial backup is shown in Example 26.



```
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_64.14866c4e3fbe964cda83e97d673b7447.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_64.932bfbcb668a973b6941054e99416988f.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 26 Restoring Using Core MW Partial Backup



5 Software Management

This section describes the following software management tasks:

- Importing software bundles and campaigns
- Deleting software bundles and campaigns
- Reading from Software inventory
- Using Software Management Framework (SMF) campaigns
- Configuring ECIM Software Management (SWM)

Commands are to be run as root user or prefixed with sudo and run by user belonging to `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.

An SDP import is shown in Example 27.

```
# cmw-sdp-import /home/MyApp/incoming/TestApp.sdp  
/home/MyApp/incoming/TestAppInstall.sdp  
ERIC-TestApp-CXP12345-R1A01 imported (type=Bundle)  
ERIC-InstallTestApp imported (type=Campaign)  
#
```

Example 27 Bundle SDP Import

A third party product (3PP) software import in Bundle RPM format is shown in Example 28.



```
# cmw-sdp-import MySQL-server-community-5.1.61-1.sles11.x86_64.rpm
3PP-MySQL-server-community-5.1.61-1.sles11 imported (type=Bundle)
#
```

Example 28 Bundle RPM Import

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.

How to delete an SDP is shown in Example 29.

```
# cmw-sdp-remove ERIC-TestApp-CXP12345-R1A01 ERIC-InstallTestApp
Bundle SDP removed [ERIC-TestApp-CXP12345-R1A01]
Campaign SDP removed [ERIC-InstallTestApp]
#
```

Example 29 Bundle SDP Remove

Note: Removing already removed software bundles or campaigns SDPs is not considered an error.

How to delete a Bundle RPM is shown in Example 30.

```
sc-1:~ # cmw-sdp-remove 3PP-MySQL-server-community-5.1.61-1.sles11
Bundle SDP removed [3PP-MySQL-server-community-5.1.61-1.sles11]
sc-1:~ #
```

Example 30 Bundle RPM Remove

5.3 Read from Software Inventory

To list the imported campaigns, use the following command:

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

A list of imported campaigns is shown in Example 31.



```
SC-1 #
cmw-repository-list --campaign
ERIC-InstallTestApp
SC-1 #
```

Example 31 List 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 32.

```
sc-1:~ # cmw-repository-list
ERIC-COREMW_COMMON-CXP9017566_1-PlA34 Used
ERIC-COREMW_SC-CXP9017565_1-PlA31 Used
ERIC-COREMW_OPENSAP-CXP9017656_1-PlA26 Used
3PP-MySQL-server-community-5.1.61-1.sles11 Used
sc-1:~ #
```

Example 32 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 the `hostname` is not used, the output looks like as in Example 33.

```
SC-1:~ # cmw-repository-list --node
PL-3 ERIC-COREMW_COMMON-CXP9017566_1-PlC72
PL-3 ERIC-COREMW_OPENSAP-CXP9017656_1-R1A01
PL-4 ERIC-COREMW_COMMON-CXP9017566_1-PlC72
PL-4 ERIC-COREMW_OPENSAP-CXP9017656_1-R1A01
SC-1 ERIC-COREMW_COMMON-CXP9017566_1-PlC72
SC-1 ERIC-COREMW_OPENSAP-CXP9017656_1-R1A01
SC-1 ERIC-COREMW_SC-CXP9017565_1-PlC53
SC-1 3PP-MySQL-server-community-5.1.61-1.sles11
SC-2 ERIC-COREMW_COMMON-CXP9017566_1-PlC72
SC-2 ERIC-COREMW_OPENSAP-CXP9017656_1-R1A01
SC-2 ERIC-COREMW_SC-CXP9017565_1-PlC53
SC-2 3PP-MySQL-server-community-5.1.61-1.sles11
SC-1:~ #
```

Example 33 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 the Core MW, the campaign SDP is imported using the `cmw-sdp-import` command. For more information about the `cmw-sdp-import` command, see Section 5.1 Import Software Bundles and Campaigns on page 23.

Imported campaigns can be listed using the following command:

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

For more information about the command, see Section 5.3 Read from Software Inventory on page 24.

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.

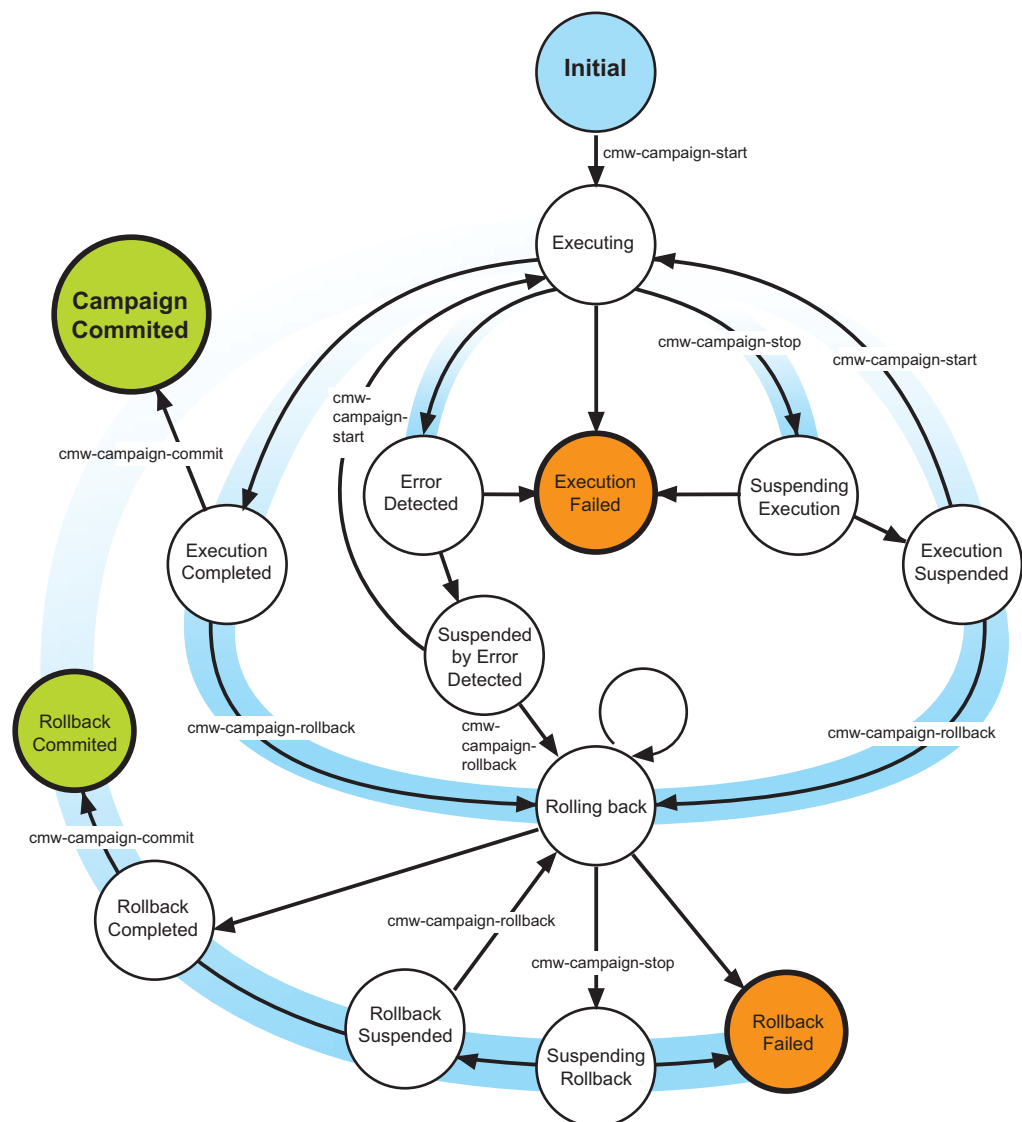


Figure 1 State Diagram



5.4.1 Execute an SMF Campaign

Make sure 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 errors that occur 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 is already executing. 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
```

For example:

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

Depending on the state, different actions are to be taken, 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	Either start or rollback.
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 20.

4. Verify the status of the cluster, see Section 3.5 Verify System Status on page 9.

If the status is OK, then proceed with Step 5, otherwise repeat the status check.

5. Commit the campaign, if it executed successfully:

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

Note: Execute the `cmw-campaign-commit` command. Then execute the `cmw-campaign-status` command until the campaign is in status `COMMITTED` (it can take some time before the campaign 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 to Step 4.

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 20.



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

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

5. Verify that the status is `ROLLBACK_COMMITTED`:

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

If the status still is `ROLLBACK_COMPLETED`, retry the `cmw-campaign-commit` command.

Note: Execute the `cmw-campaign-rollback` command. Then execute the `cmw-campaign-status` command until the campaign is in status `ROLLBACK_COMMITTED` (it can take some time before the campaign 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 27.
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-filena  
me>
```

3. Execute the upgrade software campaign, see Section 5.4.1 Execute an SMF Campaign on page 27.
4. Delete the campaign SDP and the old application software bundles:

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

5.4.5 Delete Software

Software is deleted using a campaign. The removal campaign is contained in an SDP. The removal campaign is 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 27.

3. Delete the removal campaign SDP and the software bundle:

```
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 27.

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 and `exit code 1` on any failure.

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, then the installation-dependent attributes in the ECIM SWM object must be set.

To set these attributes, use the following command:

`cmw-swm-config-set`

The command takes the options shown in Table 3.

Table 3 Command Options

Options	Description
<code>-l, --localFileStorePath</code>	<p><code>localFileStorePath</code> is the path to the locally stored upgrade packages.</p> <p>Since no default setting exists, this attribute must be assigned for the ECIM SWM functionality to work.</p>
<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", "mergedSystem". The attribute is used to select the correct campaign to apply from an Upgrade Package.</p> <p>There is no default setting for this attribute.</p>
<code>-b, --automaticBackup</code>	<p><code>automaticBackup</code> is set to 1 (<code>true</code>) if ECIM SWM is to take an automatic backup at activation.</p>
<code>-r, --automaticRestore</code>	<p><code>automaticRestore</code> is set to 1 (<code>true</code>) in order for ECIM SWM to perform automatic restore at failure.</p>



Options	Description
-f, --fallbackTimer	fallbackTimer specifies the maximum number of seconds ECIM SWM waits for user action before fallback. The default setting is 1200.
-a, --alarmBeforeTimeout	alarmBeforeTimeout specifies the time in seconds that an alarm is sent out before a fallback occurs. The default setting is 180.

Configuration of ECIM SWM is shown in Example 34.

```
cmw-swm-config-set --automaticBackup 1 -s smallCluster -l /home/my/package/repository
```

Example 34 Configuring ECIM SWM

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

Note: Expectations on the file store are as follows:

- The directory structure must be created before setting cmw-swm-config-set.
- The directory must be accessible by both controllers.
- The directory must be writable to delete the packages as part of the removeUpgradePackage() operation in the ECIM SWM MOM.

5.6 Configure ISP Events for Cluster Restarts

ISP events for cluster restarts are disabled by default.

This feature can be enabled or disabled during Automatic Installation Tool (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 within a campaign by setting the ispClusterRebootLogEnabled attribute of object with class type CmwMonitorlsp to a value of 1.

An example of a campaign Init section which enables ISP events for cluster restarts by setting the ispClusterRebootLogEnabled attribute to 1 is shown in Page 32.



```
<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 35 ISP Events for Cluster Restarts

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 it can 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` must specify them in the `<singleStepUpgrade>` section, as shown in Example 36.

```
<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 36 Campaign Defining Affected Nodes for Single-Step Upgrade Procedures

The `smfSSAffectedNodesEnable` attribute of the SMF configuration controls the nodewise reboot feature and is set to a default value of 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 37.

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

Example 37 Campaign Disabling Nodewise Reboot Feature





6 Performance Management

PM jobs can be created, 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

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

6.1 Create ECIM PM Jobs

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

The command is used as follows:

```
cmw-pmjob-create <job_name>
                  (-u <pmgroup_id>
                   [-M <meas_reader>]
                   [-R <variation_rate>]
                   [-d <thresholdDirection>]
                   [(-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 to be interpreted as follows:

- `job_name` is the `pmJobId` attribute in the `EcimMoClass PmJob` table.
- `pmgroup_id` is the Managed Object 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 (MR) and Measurement Specifications (MS).

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

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

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

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

The optional parameters for threshold jobs with default values in bold, are as follows:

```
-T <threshold_id>
    -H <threshold_high>
    -L <threshold_low>
    -S <severity>
        3 - CRITICAL
        4 - MAJOR
        5 - MINOR
        6 - WARNING
-R <variation_rate>
    0 - PER_SECOND
    1 - PER_GP
    Note: it only applies for CC collection method.
-d <thresholdDirection>
    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 will be referenced.
    It supports any number of instance parameters.
    The format is as follows:
    -i instance1 -i instance2 -i instance3
```



```

-t <job_type>
  1 - Measurement Job
  2 - Threshold Job

-r <reporting_period>
  3 - 1 minute
  4 - 5 minutes
  5 - 15 minutes
  6 - 30 minutes
  7 - 1 hour
  8 - 12 hours
  9 - 24 hours

-g <granularity_period>
  3 - 1 minute
  4 - 5 minutes
  5 - 15 minutes
  6 - 30 minutes
  7 - 1 hour
  8 - 12 hours
  9 - 24 hours

-p <job_priority>
  1 - low
  2 - medium
  3 - high

-q <requested_state>
  1 - Active
  2 - Stopped

-c <job_control>
  0 - Full
  1 - Start stop
  2 - View only

-G <job_group>

-x <compression_type>
  0 - GZIP

```

The creation of an ECIM PM job named `Job3`, which directly references MT `rxOctets` in `moClass mocRx`, is shown in Example 38.

```

SC-1:~ # cmw-pmjob-create job3 -u mocRx -m rxOctets
PM job job3 created
SC-1:~ #

```

Example 38 Create an ECIM PM Job

The command returns exit code 0 on success and exit code 1 on any failure to create the job.



Attempting to create an ECIM PM job without first defining the MT or moClass, results in failure, as shown in Example 39.

```
SC-1:~ # cmw-pmjob-create job4 -u mocTx -m txOctets
Creating job job4...
Failed to apply object creations 21
CMW: ERROR (cmw-pmjob-create): Job creation from command line failed.
SC-1:~ #
```

Example 39 Failed Attempt to Create an ECIM PM Job

How to create an ECIM PM Job that references an MT indirectly by the moClass mocRxRate (by not specifying the -m option), is shown in Example 40.

```
SC-1:~ # cmw-pmjob-create job4 -u mocRxRate
PM job job4 created.
SC-1:~ #
```

Example 40 Create an ECIM PM Job That Indirectly References an MT

Multiple MT references can be applied to the same job, as shown in Example 41.

```
SC-1:~ # cmw-pmjob-create job5 -u mocRx
-m rxDiscFr
-T rxDiscAlm
-H 300 -L 250 -S 3
-u mocRxRate
-m rxRate
-T rxRateOversub
-H 600 -L 550 -S 3
-T rxRateHigh
-H 400 -L 350 -S 4
-T rxRateBusy
-H 200 -L 150 -S 6
-t 2
PM job job5 created.
```

Example 41 Create an ECIM PM Job with Multiple MTs

Job5 is defined as a Threshold type job (-t flag set to 2), as shown in Example 41.

It contains two MR references which can be described as follows:

- The first MR directly references an MT named rxDiscFr in moClass mocRx. 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 mocRxRate. It defines three pmThresholdMonitoring thresholds (rxRateOversub, rxRateHigh, and rxRateBusy), each which has an assigned severity and threshold levels.

The job_control attribute is optional. The default value is Full, which means that the job can be started, stopped, or deleted. The PMJob2 is defined as a measurement job, which has job_control as Start stop, as shown in Example 42.



```
SC-2-1:~ # cmw-pmjob-create PMJob2 -u PMGroup-1
-t 1 -r 3 -g 3
-p 3 -q 2 -c 1
```

Example 42 Create an ECIM PM Job with Start Stop Control

6.2 Start ECIM PM Jobs

To start an ECIM PM job, use the **cmw-pmjob-start** command.

Note: The precondition is that `job_control` is Full or Start stop.

The command is used as follows:

cmw-pmjob-start <job_name>

The syntax is to be interpreted as follows:

`job_name` is the `pmJobId` attribute in the `EcimMoClass PmJob` table.

A request to start an ECIM PM job is shown in Example 43.

```
SC-1:~ # cmw-pmjob-start TC-CMW-PM-JOB-CMD-START
PM Job TC-CMW-PM-JOB-CMD-START Started.
SC-1:~ #
```

Example 43 Start an ECIM PM Job

The command returns exit code 0 on success and exit code 1 on any failure to start the job.

A failure to start an ECIM PM Job, whose `job_name` does not exist, is shown in Example 44.

```
SC-1:~ # cmw-pmjob-start TC-CMW-PM-JOB-CMD-START_non-exist
CMW: ERROR (cmw-pmjob-start): PM Job =>
TC-CMW-PM-JOB-CMD-START_non-exist does not exist
SC-1:~ #
```

Example 44 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.

6.3 Stop ECIM PM Jobs

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

Note: The precondition is that `job_control` is Full or Start stop.

The command is used as follows:

cmw-pmjob-stop <job_name>



The syntax is to be interpreted as follows:

`job_name` is the `pmJobId` attribute in the `EcimMoClass PmJob` table.

A request to stop an ECIM PM job is shown in Example 45.

```
SC-1:~ # cmw-pmjob-stop TC-CMW-PM-JOB-CMD-STOP
PM job TC-CMW-PM-JOB-CMD-STOP stopped.
SC-1:~ #
```

Example 45 Stop an ECIM PM Job

The command returns exit code 0 on success and exit code 1 on any failure to stop the job.

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

Warning: PM Job `<job_name>` is already Stopped.

6.4 Delete ECIM PM Jobs

To delete a stopped ECIM PM job, use the `cmw-pmjob-delete` command.

Note: The precondition is that `job_control` is Full.

The command is used as follows:

```
cmw-pmjob-delete <job_name>
```

The syntax is to be interpreted as follows: `job_name` is the `pmJobId` attribute in the `EcimMoClass PmJob` table.

A request to delete an ECIM PM job is shown in Example 46.

```
SC-1:~ # cmw-pmjob-delete TC-CMW-PM-JOB-CMD-CC-ECIM-001
PM job TC-CMW-PM-JOB-CMD-CC-ECIM-001 deleted.
SC-1:~ #
```

Example 46 Delete an ECIM PM Job

The command returns exit code 0 on success and exit code 1 on any failure to delete the job.

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 an exit code 1 with the following message:

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



6.5 Report Status of ECIM PM Jobs

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

The command is used as follows:

```
cmw-pmjob-status [-v] <job_name>
```

The syntax is to be interpreted as follows:

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

The output using the short format, generated when omitting the `-v` flag, is shown in Example 47.

```
SC-1:~ # cmw-pmjob-status TC-CMW-PM-JOB-CMD-STATUS-2
PM job TC-CMW-PM-JOB-CMD-STATUS-2 is active
SC-1:~ #
```

Example 47 Reporting Status of an ECIM PM Job

Using the `-v` flag, the long format, for the same job is shown in Example 48.

```
SC-1:~ # cmw-pmjob-status -v TC-CMW-PM-JOB-CMD-STATUS-2
Name Value(s)
=====
pmJobId TC-CMW-PM-JOB-CMD-STATUS-2
jobType Measurement (0x1)
granularityPeriod 1 minute (0x3)
reportingPeriod 1 minute (0x3)
jobPriority High (0x3)
requestedJobState Active (0x1)
currentJobState Active (0x1)
Overall Job State Active (0x1)
SC-1:~ #
```

Example 48 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.

6.6 List ECIM PM Jobs

To list ECIM PM jobs, use the `cmw-pmjob-list [options]` command.

The command is used as follows:

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



The possible options are as follows:

```
-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 <N>
-t <N>  List only jobs of this job type <N>
-p <N>  List only jobs with this job priority <N>
-g <N>  List only jobs with this granularity period <N>
-q <N>  List only jobs with this requested job state <N>
-s <N>  List only jobs with this current job state <N>
where <N> is a numeral.
```

Using the command without any options results in a short listing, showing each Ecim PM Job name, requested job state, and current job state. A short listing is shown in Example 49.

```
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 49 Listing ECIM PM Jobs

The command contains a -v flag for Verbose or long format. The output contains detailed information about each ECIM PM Job. A verbose listing is shown in Example 50.

```
SC-1:~ # cmw-pmjob-list -v
Name                               Value(s)
=====
pmJobId                           Job1
jobType                           Measurement(0x1)
granularityPeriod                  15 minutes(0x5)
reportingPeriod                    5 minutes(0x4)
jobPriority                         Medium(0x2)
requestedJobState                  Active(0x1)
currentJobState                   Active(0x1)
Overall Job State                 Active(0x1)
Name                               Value(s)
=====
pmJobId                           Job2
jobType                           Threshold(0x2)
granularityPeriod                  15 minutes(0x5)
reportingPeriod                    1 minute(0x3)
jobPriority                         Medium(0x2)
requestedJobState                  Active(0x1)
currentJobState                   Active(0x1)
Overall Job State                 Active(0x1)
Name                               Value(s)
=====
pmJobId                           Job3
jobType                           Threshold(0x2)
granularityPeriod                  15 minutes(0x5)
reportingPeriod                    1 hour(0x7)
jobPriority                         Medium(0x2)
requestedJobState                  Active(0x1)
currentJobState                   Active(0x1)
Overall Job State                 Active(0x1)
SC-1:~ #
```

Example 50 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 50, jobs that have `reportingPeriod` equal to 1 minute and `jobType` equal to Measurement can be shown. A filtered list, using multiple options, is shown in Example 51.

```
SC-1:~ # cmw-pmjob-list -v -r 3 -t 1
Name                               Value(s)
=====
pmJobId                             Job1
jobType                             Measurement (0x1)
granularityPeriod                   15 minutes (0x5)
reportingPeriod                     5 minutes (0x4)
jobPriority                         Medium (0x2)
requestedJobState                   Active (0x1)
currentJobState                     Active (0x1)
Overall Job State                   Active (0x1)
Name                               Value(s)
=====
pmJobId                             Job2
jobType                             Threshold (0x2)
granularityPeriod                   15 minutes (0x5)
reportingPeriod                     1 minute (0x3)
jobPriority                         Medium (0x2)
requestedJobState                   Active (0x1)
currentJobState                     Active (0x1)
Overall Job State                   Active (0x1)
SC-1:~ #
```

Example 51 *Listing ECIM PM Jobs Using Multiple Options*

Multiple options can be used in short format (without the `-v` flag), as shown in Example 52.

```
SC-1:~ # cmw-pmjob-list -r 3 -t 1 -g 5
pmJobId=Job2, requestedJobState=Active(0x1), =>
    currentJobState=Active(0x1), reportingPeriod=3
pmJobId=Job1, requestedJobState=Active(0x1), =>
    currentJobState=Active(0x1), jobType=1
pmJobId=Job1, requestedJobState=Active(0x1), =>
    currentJobState=Active(0x1), granularityPeriod=5
pmJobId=Job2, requestedJobState=Active(0x1), =>
    currentJobState=Active(0x1), granularityPeriod=5
pmJobId=Job3, requestedJobState=Active(0x1), =>
    currentJobState=Active(0x1), granularityPeriod=5
SC-1:~ #
```

Example 52 *Listing ECIM PM Jobs in Short Format with Multiple Flags*