

Prepare Upgrade Package

OPERATING INSTRUCTIONS

Copyright

© Ericsson AB 2016, 2017. All rights reserved. No part of this document may be reproduced in any form without the written permission of the copyright owner.

Disclaimer

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

Trademark List

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



Contents

| | | |
|----------|-------------------------|----------|
| 1 | Description | 1 |
| 2 | Procedure | 2 |
| 2.1 | Prepare Upgrade Package | 2 |





1 Description

This instruction describes the preparation phase of a software upgrade.

A software upgrade has a limited time window during which some planned service impact is allowed. The preparation phase has no service impact. It is therefore recommended to perform it outside the upgrade time window during normal “traffic hours”. Use the upgrade time window only for the execution phase of the software upgrade. The execution phase is described in *Activate Upgrade*.

This preparation phase consists of the following:

- Creates the upgrade package, by making the upgrade package visible in the Managed Object Model.
- Prepares the upgrade package, by performing the relevant software upgrade package extraction activities or integrity checks. The integrity check performs checksum checks or equivalent (oblivious hashing, check and guard system, active or passive tamper resistance) and is needed to secure that the correct upgrade package has been fetched.
- Verifies the Management Element (ME) ability to activate the upgrade package.
- Allows the user to configure the upgrade package execution phase behavior, that is, upgrade execution method and activation behavior regarding breakpoints and steps.
- Checks the Managed Element (ME) and uses the model compiler and merge tool inside the UP package.

The procedure in this document shows how to configure the upgrade package execution method by the *UpgradePackage* and *RoleUpgrade* managed object attributes. Default execution method is configured using system-level *SwM* and *RoleUpgrade* managed object attributes. The procedure step to create the *UpgradePackage* managed object populates the upgrade package execution method with default, that is system-level, execution method values.

The procedure in this document is illustrated by an example where a software version with product name *ERIC-COREMW_RUNTIME*, product number *CXP9020355_1*, and product revision *R7E01* is running in the system. A software upgrade package *ERIC_UP-CXP9020355_1-R7F01*, which is designed to upgrade this software version to product revision *R7F01*, is going through the preparation phase.



2 Procedure

2.1 Prepare Upgrade Package

Prerequisites

- No documents are required.
- No tools are required.
- The following conditions must apply:
 - The ME has passed a health check routine.
 - The uncompressed software upgrade package file is present under a known directory on a remote repository. In this document, `/node/software/upgrade/R7F01` is used as a directory example.
 - The required SSH File Transfer Protocol (SFTP) user and password are known.
 - There are no black patches on the site.
 - A manual backup has been done before the upgrade.
 - An Ericsson Command-Line Interface (ECLI) session in Exec mode is in progress.

Steps

1. Navigate to the *SwM* managed object, for example:

```
>dn ManagedElement=NODE06ST,SystemFunctions=1,SwM=1
```

2. Create an upgrade package managed object by importing the software upgrade package file from a remote repository, for example:

```
(SwM=1)>createUpgradePackage --uri sftp://user@192.0.2.10:/node/software/upgrade/R7F01 --password 26538813
```

The system returns `invocation Id (actionId)`, which is 2 in this example. This indicates a successful operation. If the operation is unsuccessful, the system returns 0. The returned `actionId` is also presented in the associated `reportProgress` structure as an attribute.

```
2
```

3. Verify the result for a successfully triggered operation:

```
(SwM=1)>show -v
```



The following example output shows the final result. If `state=RUNNING`, the creation is not yet completed. Result `result=FAILURE` means that the creation failed.

```
SwM=1
[...]
  reportProgress <read-only>
    actionId=2 <read-only>
[...]
  progressPercentage=100 <read-only>
  result=SUCCESS <read-only>
  state=FINISHED <read-only>
[...]
```

4. Navigate to the *UpgradePackage* managed object:

```
(SwM=1) >UpgradePackage=ERIC_UP-CXP9020355_1-R7F01
```

5. Prepare the upgrade package:

```
(UpgradePackage=ERIC_UP-CXP9020355_1-R7F01) >prepare
```

The system returns output `true` or `false`.

6. Verify the preparation:

```
(UpgradePackage=ERIC_UP-CXP9020355_1-R7F01) >show -v
```

The following is an example output, which shows the final result. If a transient result is shown, where `state=RUNNING`, the operation is not yet completed.

```
UpgradePackage=ERIC_UP-CXP9020355_1-R7F01
  activationFallbackTimer=600 <read-only>
[...]
  state=PREPARE_COMPLETED <read-only>
[...]
  reportProgress <read-only>
    actionId=5 <read-only>
    actionName="Prepare" <read-only>
    additionalInfo[]
      "" <read-only>
    progressInfo="Prepare UpgradePackage" <read-only>
    progressPercentage=100 <read-only>
    result=SUCCESS <read-only>
    resultInfo="" <read-only>
    state=FINISHED <read-only>
    step=1 <read-only>
    stepProgressPercentage=0 <read-only>
    timeActionCompleted="2013-12-18T04:43:08" <read-only>
    timeActionStarted="2013-12-18T04:43:03" <read-only>
    timeOfLastStatusUpdate="2013-12-18T04:43:08" <read-only>
```

7. Verify the upgrade package:

```
(UpgradePackage=ERIC_UP-CXP9020355_1-R7F01) >verify
```

The system returns output `true` or `false`.

8. Verify the result:

```
(UpgradePackage=ERIC_UP-CXP9020355_1-R7F01) >show -v
```



The following is an example output:

```
UpgradePackage=ERIC_UP-CXP9020355_1-R7F01
[...] state=PREPARE_COMPLETED
[...] reportProgress
[...]   actionName="Verify"
[...]   result=SUCCESS
[...]   state=FINISHED
[...]
```

This output shows the final result. If `state=RUNNING` is shown, the operation is not yet completed. Result `result=FAILURE` means that the creation failed.

9. Verify the upgrade execution method:

```
(UpgradePackage=ERIC_UP-CXP9020355_1-R7F01) >show  
execMethod
```

The following is an example output:

```
execMethod=ROLLING
```

10. Is the displayed execution method appropriate for the upgrade package?

Yes: Proceed with Step 18.

No: Continue with the next step.

11. Enter Config mode:

```
(UpgradePackage=ERIC_UP-CXP9020355_1-R7F01) >configure
```

12. Perform one of the following options:

- Set Rolling Upgrade execution method:

```
(config-UpgradePackage=ERIC_UP-CXP9020355_1-R7F01)  
>execMethod=ROLLING
```

Proceed with Step 17.

- Set Single-Step Upgrade execution method:

```
(config-UpgradePackage=ERIC_UP-CXP9020355_1-R7F01) >e  
xecMethod=ONE_STEP
```

Proceed with Step 17.

- Set Balanced In-Service Upgrade (BISU) execution method:



```
(config-UpgradePackage=ERIC_UP-CXP9020355_1-R7F01) > execMethod=BALANCED
```

Configuration of BISU parameters is done in Step 13 through Step 16.

13. Create a *RoleUpgrade* managed object, for example:

```
(config-UpgradePackage=ERIC_UP-CXP9020355_1-R7F01) > RoleUpgrade=BISU-service-config
```

14. Set BISU parameters by setting the relevant attributes, for example:

- (config-RoleUpgrade=BISU-service-config) > **roleId="ManagedElement=NODE06ST, SystemFunctions=1, SysM=1, CrM=1, Role=PLs"**

Here `roleId` is the scalable `CrM` role specifying the payload nodes to be upgraded using the BISU execution method.

- (config-RoleUpgrade=BISU-service-config) > **minSizeOfRole=10**

Here `minSizeOfRole` is the minimum number of nodes in the scalable role to do the upgrade in bigger chunks than one by one (Rolling Upgrade execution method).

Note: If fewer nodes than `minSizeOfRole` are available in the role, the Rolling Upgrade execution method is used during activation of the upgrade package.

- (config-RoleUpgrade=BISU-service-config) > **minRemainingCapacity=65**

Here `minRemainingCapacity` is the minimum remaining capacity in % of the engineered capacity during the activation of the upgrade.

Note: If the number of nodes available in the role cannot supply the minimum remaining capacity, the Rolling Upgrade execution method is used during activation of the upgrade package.

15. Navigate to the *UpgradePackage* managed object, for example:

```
(config-RoleUpgrade=BISU-service-config) > dnManagedElement=NODE06ST, SystemFunctions=1, SwM=1, UpgradePackage=ERIC_UP-CXP9020355_1-R7F01
```

16. Verify the result:

```
(config-UpgradePackage=ERIC_UP-CXP9020355_1-R7F01) > show -v -r
```

The following is an example output:



```
UpgradePackage=ERIC_UP-CXP9020355_1-R7F01
[...]
  execMethod=BALANCED
[...]
  RoleUpgrade=BISU-service-config
    minRemainingCapacity=65
    minSizeOfRole=10
    numberOfSteps=[] <empty>
    roleId="ManagedElement=NODE06ST,SystemFunctions=1,SysM=1,CrM=1,⇒
    Role=Pls",
    roleUpgradeId="BISU-service-config"
[...]
```

Note: Attribute `numberOfSteps` in the `RoleUpgrade` managed object is calculated during activation of the upgrade package.

17. Commit the changes:

```
(config-UpgradePackage=ERIC_UP-CXP9020355_1-R7F01) >commit
```

18. Verify the activation step behavior:

```
(UpgradePackage=ERIC_UP-CXP9020355_1-R7F01) >show ignoreBreakPoints
```

The following output is shown if the upgrade package is configured for one-step activation:

```
ignoreBreakPoints=true
```

19. Is the displayed activation behavior appropriate for the upgrade package?

Yes: Job is completed.

No: Continue with the next step.

20. Enter Config mode:

```
(UpgradePackage=ERIC_UP-CXP9020355_1-R7F01) >configure
```

21. Perform one of the following options:

- Set one-step activation:

```
(config-UpgradePackage=ERIC_UP-CXP9020355_1-R7F01) >ignoreBreakPoints=true
```

- Set step-by-step activation:

```
(config-UpgradePackage=ERIC_UP-CXP9020355_1-R7F01) >ignoreBreakPoints=false
```

22. Commit the setting:

```
(config-UpgradePackage=ERIC_UP-CXP9020355_1-R7F01) >commit
```



23. Verify the result:

```
(UpgradePackage=ERIC_UP-CXP9020355_1-R7F01) >ignoreBreakPoints
```

The following is an example output:

```
ignoreBreakPoints=true
```