# Oracle® Communications Network Charging and Control

## Number Portability Service Pack Technical Guide

Release 15.2


January 2026

# Copyright

# Contents

# About This Document

## Scope

The scope of this document includes all the information required to administer the Messaging Firewall application.

## Audience

This guide is written primarily for system administrators and other personnel who administer the Messaging Firewall application. However, the overview sections of the document are useful to anyone requiring an introduction to the application.

## Prerequisites

Although it is not a prerequisite to using this guide, familiarity with the target platform would be an advantage.

A solid understanding of Unix and a familiarity with IN concepts are an essential prerequisite for safely using the information contained in this technical guide. Attempting to install, remove, configure or otherwise alter the described system without the appropriate background skills, could cause damage to the system; including temporary or permanent incorrect operation, loss of service, and may render your system beyond recovery.

This manual describes system tasks that should only be carried out by suitably trained operators.

## Related documents

The following documents are related to this document:

- *NP Service Pack User's Guide*
- *CCS User's Guide*
- *CCS Technical Guide*
- *CCS Feature Node User's Guide*
- *CPE User's Guide*
- *ACS User's Guide*
- *ACS Technical Guide*

# Document Conventions

## Typographical Conventions

The following terms and typographical conventions are used in the Oracle Communications Network Charging and Control (NCC) documentation.

| Formatting Convention | Type of Information |
|---|---|
| **Special Bold** | Items you must select, such as names of tabs.<br>Names of database tables and fields. |
| *Italics* | Name of a document, chapter, topic or other publication.<br>Emphasis within text. |
| **Button** | The name of a button to click or a key to press.<br>**Example:** To close the window, either click **Close**, or press **Esc**. |
| **Key+Key** | Key combinations for which the user must press and hold down one key and then press another.<br>Example: **Ctrl+P** or **Alt+F4**. |
| `Monospace` | Examples of code or standard output. |
| **`Monospace Bold`** | Text that you must enter. |
| *variable* | Used to indicate variables or text that should be replaced with an actual value. |
| **menu option > menu option >** | Used to indicate the cascading menu option to be selected.<br>Example: **Operator Functions > Report Functions** |
| hypertext link | Used to indicate a hypertext link. |

Specialized terms and acronyms are defined in the glossary at the end of this guide.

Chapter 1

# System Overview

## Overview

### Introduction

This chapter provides a high-level overview of the application. It explains the basic functionality of the system and lists the main components.

It is not intended to advise on any specific Oracle Communications Network Charging and Control (NCC) network or service implications of the product.

### In this Chapter

This chapter contains the following topics.



## What is the NP Service Pack

### Introduction

The Number Portability Service Pack (NP Service Pack), accessed through SMS, provides flexibility and control over call routing of subscribers to the network.

### Features

Features of NP Service Pack include:

- The ability to configure ported subscriber information from the screens and/or using the NP Provisioning Interface (PI) commands, and supporting control plan nodes to make services NP/MNP aware
- The ability to configure routing information based on operator assigned number prefixes and supporting control plan nodes to allow services to route to appropriate operators
- Functionality to allow the IN platform to satisfy MNP SRF requirements through a MAP based application (MTA) that can trigger a control plan when a supported message is received. The control plan can then perform MNP and through the appropriate node, instruct the MTA on how to respond (for example, relay, ack or error)
- Simple GTT functionality provided in the MTA for performing relay actions
- Call connection using least cost routing based on predefined carrier selection rule sets
- Home routing for calls within the network
- EDR generation for all calls processed by the network. The EDRs can be used for billing and reporting purposes.

### NP Feature Nodes

The following feature nodes are available for the NP Service within the ACS Control Plan Editor:

- NP Destination Selection

- NP Least Cost Routing
- NP Home Routing
- NP Map Trigger

See *Feature Nodes Reference Guide* for information on these nodes. See *CPE User's Guide* for information on using the ACS Control Plan Editor.

## NP PI Commands

A number of additional Provisioning Interface (PI) commands and utilities are supplied for NP. These supplement the existing PI commands available for CCS.

The NP PI commands can be used to add, modify and delete the following data through batch input:

- DN Ranges
- LCR Rule Sets and Rules
- Home Routing data

For a description of the NP PI commands, see *NP Provisioning Interface Commands*.

For a description of the CCS PI commands, see *CCS Provisioning Interface Commands.*

For more information about the PI, see *PI User's and Technical Guide*.

Chapter 2

# Configuration

## Overview

### Introduction

This chapter explains how to configure the Oracle Communications Network Charging and Control (NCC) application.

### In this chapter

This chapter contains the following topics.



## Configuration Overview

### Introduction

This topic provides a high level overview of how the NP Service Pack is configured.

There are configuration options that are added to the configuration files that are not explained in this chapter. These configuration options are required by the system and should not be changed.

### Configuration Components

Number Portability Service Pack is configured by the following components:

| Component | Locations | Description | Further Information |
|---|---|---|---|
| **acs.conf** | SMS and SLC | Configures the acsChassis which processes calls. | *acs.conf Configuration* (on page 4) |
| **cdrIF.cfg** | SLC | Configures the EDR interface. | *Configuring EDR Collection* (on page 10) |
| **mta.cfg** | SLC | Configures the MAP Trigger application. | *mfw* (on page 16) |
| **np_components.cfg** | SLC | Configures the NP specific macro nodes. | *np_components.cfg Configuration* (on page 5) |
| **SLEE.cfg** | SLC | The <SLEE _ac>interface is configured to include the NP application, the MTA and the cdrIF interface. | *SLEE.cfg Configuration* (on page 9) and *SLEE Technical Guide.* |

### Configuration file format

The NP Service Pack configuration files are located in the **/IN/service_packages/NP_SERVICE_PACK/etc** directory.

### Editing Configuration Files

To ensure that you have a working copy, before making any changes to the configuration files, backup your current configuration.

To edit configuration files, open the configuration file using a standard file editor. Do not use file editors such as Microsoft Word that attach Microsoft DOS or Windows line termination characters (for example, ^M) at the end of each row, because this causes file errors when the application tries to read the configuration file.

### Loading Configuration Changes

If you change a configuration file, then you must send a signal (SIGHUP) to the relevant process, or restart the SLEE to enable the new options to take effect.

# acs.conf Configuration

### Introduction

The **acs.conf** file must be configured on the SLC to enable NP to work.

The **acs.conf** file will be automatically updated during the installation of the npScp package.

Refer to *ACS Technical Guide* for more information on **acs.conf** configuration.

### acsChassis Configuration

Once the NP packages have been installed, you should check that the following lines have been added to the `acsChassis` section of **acs.conf**:

```
MacroNodePluginFile libNpSpecificMacroNodeLoader.so  # Inserted by npScp
ChassisPlugin libNpCpuChassisActions.so              # Inserted by npScp
```

Optionally this line can be added to the `acsChassis` section:

```
extensionNumber 0 101 inapNumber digits
```

When the MAP Trigger Application (MTA) receives a MAP message it builds an IDP using information from the received MAP message. If the above configuration is specified the first extension number in the IDP will be set to the original MSISDN (before normalization) contained in the received MAP message.

Optionally this service library can also be added to **acs.conf**:

```
acsServiceLibrary
        ExtraPORPlugin
/IN/service_packages/NP_SERVICE_PACK/lib/libnpAcsSvcExtra.so:
```

This will enable the “Append URI” functionality in the Destination Selection feature node. See *Number Portability Service Pack User's Guide* for more information.

# np_components.cfg Configuration

## Introduction

The **np_components.cfg** file must be configured to enable the NP feature nodes to work. An example **np_components.cfg** file showing the configuration required is provided in **/IN/service_packages/NP_SERVICE_PACK/etc/np_components.cfg**.

## np_components.cfg Configuration

Once the NP packages have been installed, you must edit **np_components.cfg** to ensure it contains the following lines:

```
appID:NP
DNMinimal:6
DNMaximal:15
InternalDestination:Internal_Destination
DefaultDestination:Default_Destination
```

## Example np_components.cfg Configuration File

Here is an example **np_components.cfg** file.

```
appID:NP
DNMinimal:6
DNMaximal:15
InternalDestination:Internal_Destination
DefaultDestination:Default_Destination
DNNoaPrefix:
#AddStopDigit
#UseCutAndPaste
CCSDeployment:no
SendCarrierCode:no
FormatCLI:no
#GenLCRCDR:no
ISUPTrunkInCDR:no
OPInRanges
#DoPQYZAfterRanges
#PQYZNumType:M
UsePQYZMFile:yes
```

## Parameters

This topic describes the parameters in **np_components.cfg**.

`AddStopDigit`

| | |
|---|---|
| **Syntax:** | `AddStopDigit` |
| **Description:** | Determines whether a stop digit is added to the end of the pending termination number when the carrier-available branch is activated. |
| **Allowed:** | set (exists in config), not set (not in config) |
| **Default:** | False (not set) |
| **Notes:** | This parameter is used by the NP Least Cost Routing feature node. |
| **Example:** | `AddStopDigit` |

### appID

**Syntax:** appID:*value*

**Description:** Defines the NP application.

**Default:** NP

**Example:** appID:NP

### CCSDeployment

**Syntax:** CCSDeployment:*yes|no*

**Description:** Determines whether the NP_SERVICE_PACK will be installed and used in combination with the NCC CCS application.

**Allowed:** yes, no

**Default:** no

**Example:** CCSDeployment:yes

### DefaultDestination

**Syntax:** DefaultDestination:*value*

**Description:** Defines the default routing destination name for NP Service Pack.

**Default:** Default_Destination

**Notes:** This parameter is used to initialize the NP Destination Selection feature node.

**Example:** DefaultDestination:Default_Destination

### DNMaximal

**Syntax:** DNMaximal:*value*

**Description:** Defines the maximum length for the dialed number.

**Allowed:** A numeric value

**Default:** 15

**Notes:** The DNMaximal value must be less than the DNMinimal value. This parameter is used to initialize the Destination Selection feature node.

**Example:** DNMaximal:15

### DNMinimal

**Syntax:** DNMinimal:*value*

**Description:** Defines the minimum length for the dialed number.

**Allowed:** A numeric value

**Default:** 6

**Notes:** The DNMinimal value must be less than the DNMaximal value. This parameter is used to initialize the Destination Selection feature node.

**Example:** DNMinimal:6

### DNNoaPrefix

**Syntax:** DNNoaPrefix:*value*

**Description:** A string that will be pre-pended to the called number NOA digit that is added to the called number by the Least Cost Routing service.

**Allowed:** Up to 3 hexadecimal digits

**Notes:** This parameter is used by the NP Least Cost Routing feature node.

**Example:** `DNNoaPrefix:`

### DoPQYZAfterRanges

**Syntax:** `DoPQYZAfterRanges`

**Description:** When set, if a matching operator cannot be found in the Dn Range table, then the PQYZ table will be searched.

**Type:** Boolean

**Optionality:** Optional

**Allowed:** set (present in config), not set (not present in config).

**Default:** False (not set)

**Notes:** This parameter is used by the NP Destination Selection node.

**Example:** `DoPQYZAfterRanges`

### FormatCLI

**Syntax:** `FormatCLI:`*`yes|no`*

**Description:** Determines whether the service will apply the preferred number formatting of the selected carrier to the CLI.

**Allowed:** yes, no

**Default:** yes

**Notes:** If set to `no`, the preferred number formatting of the selected carrier is only applied to the termination number, and the original CLI is preserved.

This parameter is used by the NP Least Cost Routing feature node.

**Example:** `FormatCLI:yes`

### GenLCRCDR

**Syntax:** `GenLCRCDR:`*`yes|no`*

**Description:** Determines whether an EDR should be generated each time the Least Cost Routing functionality is successfully applied during the processing of a running call.

**Allowed:** yes, no

**Default:** no

**Notes:** Setting this parameter to `yes` will cause multiple EDRs being generated for the same call, one for each LCR attempt.

This parameter is used by the NP Least Cost Routing feature node.

**Example:** `GenLCRCDR:yes`

### InternalDestination

**Syntax:** `InternalDestination:`*`value`*

**Description:** Defines the internal routing destination name for NP Service Pack.

**Default:** Internal_Destination

**Notes:** This parameter is used to initialize the NP Destination Selection feature node.

**Example:** `InternalDestination:Internal_Destination`

### ISUPTrunkInCDR

| | |
|---|---|
| **Syntax:** | `ISUPTrunkInCDR:`*`yes`*`|`*`no`* |
| **Description:** | Determines whether the service will include the ORIGTRUNK tag in the EDR. |
| **Allowed:** | yes, no |
| **Default:** | yes |
| **Notes:** | The value of the ORIGTRUNK tag is set to the value of the locationNumber buffer.<br>This parameter is used by the NP Least Cost Routing feature node. |
| **Example:** | `ISUPTrunkInCDR:yes` |

### OPInRanges

| | |
|---|---|
| **Syntax:** | `OPInRanges` |
| **Description:** | Determines whether operators will be stored in the dn range table |
| **Type:** | Boolean |
| **Optionality:** | Optional. |
| **Allowed:** | set (present in config), not set (not in config) |
| **Default:** | False (not set) |
| **Notes:** | This parameter is used by the NP Destination Selection node. |
| **Example:** | `OPInRanges` |

### PQYZNumType

| | |
|---|---|
| **Syntax:** | `PQYZNumType:`*`value`* |
| **Description:** | Defines whether to exit from the NP Destination Selection node along a fixed or a mobile branch. |
| **Optionality:** | Optional (default used if not set). |
| **Allowed:** | M (mobile), F (fixed) |
| **Default:** | F |
| **Notes:** | This parameter is used by the NP Destination Selection node. |
| **Example:** | `PQYZNumType:M` |

### SendCarrierCode

| | |
|---|---|
| **Syntax:** | `SendCarrierCode:`*`yes`*`|`*`no`* |
| **Description:** | Determines whether the service will include the carrier code in the DRA. |
| **Allowed:** | yes, no |
| **Default:** | yes |
| **Notes:** | This parameter is used by the NP Least Cost Routing feature node. |
| **Example:** | `SendCarrierCode:yes` |

### UseCutAndPaste

| | |
|---|---|
| **Syntyax:** | `UseCutAndPaste` |
| **Description:** | Determines whether INAP Cut and Paste is used in a Connect operation that is issued after the “Carrier available” branch of an LCR feature node has been selected. |
| **Allowed:** | set (exists in config), not set (not in config) |
| **Default:** | False (not set) |

| | |
|---|---|
| **Notes:** | If the parameter is set, then Cut and Paste will be set to the length of the original un-normalized called number, so that all digits that were initially dialed are cut. This action will ensure that any subsequent digits in an overlap sending scenario will be preserved. |
| **Example:** | `UseCutAndPaste` |

`UsePQYZMFile`

| | |
|---|---|
| **Syntax:** | `UsePQYZMFile:yes\|no` |
| **Description:** | Sets whether the NP Destination Selection feature node retrieves PQYZ entries from the NP database or from NP MFiles. For example, it may be more efficient to retrieve PQYZ data directly from the NP database if the database holds a large number of PQYZ entries and those entries change frequently, causing the PQYZ MFile to be rebuilt constantly. |
| **Type:** | Boolean |
| **Optionality:** | Optional (default used if not set) |
| **Allowed:** | • `yes` – The feature node retrieves PQYZ data from the PQYZ MFile<br>• `no` – The feature node retrieves PQYZ data directly from the NP database and the PQYZ MFile is not used |
| **Default:** | yes |
| **Notes:** | For more information about the NP Destination Selection feature node, see *Feature Nodes Reference Guide*. |
| **Example:** | `UsePQYZMFile:no` |

# SLEE.cfg Configuration

## Introduction

The **SLEE.cfg** file must be configured to enable NP to work and to enable the collection of NP- specific EDRs.

The **SLEE.cfg** file will be automatically updated during the installation of the npScp package to set up the platform to use the MTA and the NP cdrIF interface.

Refer to:

- *SLEE Technical Guide* for more information on SLEE configuration.
- *XML TCAP Interface Technical Guide* for more information on SERVICEKEY format.

## NP SLEE Configuration

Once the npScp package has been installed, check that the following lines have been added to the **SLEE.cfg** file:

```
INTERFACE=cdrIF cdrIF /IN/service_packages/NP_SERVICE_PACK/bin EVENT # Inserted by npScp
APPLICATION=mtaApplication mta.sh /IN/service_packages/NP_SERVICE_PACK/bin 1 1 # Inserted by
npScp
SERVICEKEY=INTEGER 0x10800000016 MTA # Inserted by npScp
SERVICEKEY=INTEGER 0x1080000002D MTA # Inserted by npScp
SERVICEKEY=INTEGER 0x10500000047 MTA # Inserted by npScp  (GSM ATI)
SERVICEKEY=INTEGER 0x1050000090F MTA # Inserted by npScp (IS41 Location Request)
SERVICEKEY=INTEGER 0x10500000937 MTA # Inserted by npScp (IS41 SMS Request)
SERVICE=MTA 1 mtaApplication MTA # Inserted by npScp
```

**Note:**

- It is essential for the correct operation of the CDR interface that the SLEE interface type for cdrIF is always set to EVENT.

SERVICEKEY values are generated from the subsystem number and base key specified during the npScp installation process and therefore may be different to those shown here.

# Configuring EDR Collection

## Introduction

NP can be configured to produce EDRs for use in post processing as required. All EDR configuration is done in the **cdrIF.cfg** file. The EDRs are saved to file in a location specified in **cdrIF.cfg**.

## EDR collection

Each call processed can produce a single EDR, or multiple EDRs, depending on the type and outcome of the call. As a minimum, each call invokes either an ACS or a CCS service, producing one ACS/CCS EDR for every termination attempt.

Where Least Cost Routing (LCR) is invoked, an LCR EDR is produced for every carrier selected for termination as part of the LCR service logic, in addition to theACS/CCS EDR produced for every termination attempt. This means that the number of LCR EDRs and the number of ACS/CCS EDRs produced for the call is the same.

## Format

EDRs are saved to file in tag/value pairs, separated by "|", in the following form:

```
tag1=value1|tag2=value2
```

**Note:** For ACS/CCS EDRS, the first value in the EDR is not a tag/value pair. It contains the name of the service that created the EDR (either ACS or CCS) only.

## Parameters

The parameters used to configure EDR collection in the application are contained in the following sections in **cdrIF.cfg**:

- `Defaults`
- `RecordDef`

## Defaults section

Here is an example of `Defaults` section configuration in **cdrIF.cfg**. This section defines default values for the EDR files.

```
Defaults {
    TempDirectory="/IN/service_packages/NP_SERVICE_PACK/cdr/temp"
    FileSize=4096
    FileDirectory="/IN/service_packages/NP_SERVICE_PACK/cdr/Closed"
}
```

`FileDirectory`

**Syntax:** `FileDirectory = "dir"`

**Description:** Specifies the final directory for storing the EDR files.

**Allowed:**

**Default:** "/IN/service_packages/NP_SERVICE_PACK/cdr/default/"

**Notes:** This is also the default final directory location for LCR EDRs if this parameter is not set in the `RecordDef` section.

**Example:**
```
FileDirectory =
"/IN/service_packages/NP_SERVICE_PACK/cdr/default/"
```

### FileSize

**Syntax:** `FileSize = size`

**Description:** Specifies the maximum size (in bytes) for the EDR files.

**Default:** 4096

**Example:**
```
FileSize = 4096
```

### TempDirectory

**Syntax:** `TempDirectory = "dir"`

**Description:** Specifies the temporary directory where the EDR files are stored before being moved to their final location.

**Allowed:**

**Default:** "/IN/service_packages/NP_SERVICE_PACK/cdr/temp"

**Notes:**

**Example:**
```
TempDirectory =
"/IN/service_packages/NP_SERVICE_PACK/cdr/temp"
```

## RecordDef section

Here is an example of the `RecordDef section` in **cdrIF.cfg**. This section defines the location and characteristics of the LCR EDRs.

```
RecordDef "NPLCRCDR" {
    FileDesc {
        FileDirectory="/IN/service_packages/NP_SERVICE_PACK/cdr/Closed/"
        FileName="LCR_%C(%y%M%d%h%m%s).cdr"
        FileHeader=""
        FileFooter=""
        RowHeader=""
        FileSize=1024000
        CdrFileMaxSize=1000
        CdrFileMaxAge=3600
        RowTrailer="\n"
        ColumnSeperator="|"
        RemoveNullColumns=true
    }

    ColumnDef {
        PID   "%s"                          ""
        CID   "%s"                          ""
        CUST  "%s"                          ""
        SN    "%s"                          ""
        TNNUM "%s"                          ""
        TNNOA "%s"                          ""
        CLI   "%s"                          ""
        SK    "%s"                          ""
        CPN   "%s"                          ""
        PTI   "%s"                          ""
```

```
        TIME  "%s"                              ""
        CALLINGNUM "%s"                         ""
        CALLINGNOA "%s"                         ""
        ROUTEDEST  "%s"                         ""
        CARRIERNAME "%s"                        ""
        CARRIERPOS "%s"                         ""
        ORIGTRUNK  "%s"                         ""
    }
}
```

### CdrFileMaxAge

**Syntax:** CdrFileMaxAge= *seconds*

**Description:** Specifies the maximum number of seconds that new records may be added to the EDR file before it is closed and moved to the directory location specified by the FileDirectory parameter.

**Allowed:** A numeric value.

**Default:** 3600

**Notes:** The file will already have been closed and moved, if CdrFileMaxSize is reached first.

**Example:** CdrFileMaxAge= 3600

### CdrFileMaxSize

**Syntax:** CdrFileMaxSize= *num*

**Description:** Specifies the maximum number of records in the EDR file.

**Allowed:** A numeric value.

**Default:** 1000

**Notes:**

**Example:** CdrFileMaxSize= 1000

### ColumnDef

This section specifies the LCR EDR fields (tags). For a description of each, see LCR EDRs.

### ColumnSeperator

**Syntax:** ColumnSeperator = "*char*"

**Description:** Specifies the character to use to separate the EDR fields.

**Allowed:**

**Default:** "|"

**Notes:**

**Example:** ColumnSeperator = "|"

### FileDesc

Identifies the file description section for the LCR EDRs.

### FileDirectory

**Syntax:** FileDirectory = "*dir*"

**Description:** Specifies the directory where the LCR EDR files are finally stored.

**Allowed:**

**Default:** "/IN/service_packages/NP_SERVICE_PACK/cdr/LCR/"

**Notes:**

**Example:** `FileDirectory = "/IN/service_packages/NP_SERVICE_PACK/cdr/LCR/"`

### FileFooter

**Syntax:** `FileFooter = "text"`

**Description:** Specifies the text to be appended to the end of each EDR file.

**Allowed:**

**Default:** ""

**Notes:**

**Example:** `FileFooter = ""`

### FileHeader

This parameter defines the text to be appended to the beginning of each EDR file.

**Default:** ""

**Allowed:** -

### FileName

**Syntax:** `FileName = "name"`

**Description:** Specifies the name format for the LCR EDR files.

**Allowed:**

**Default:** "LCR_%C(%y%M%d%h%m%s).cdr"

**Notes:** %C(%y%M%d%h%m%s) represents the time stamp for when the EDR file was created.

**Example file name:** `LCR_20060118105515.cdr`

**Example:** `FileName = "LCR_%C(%y%M%d%h%m%s).cdr"`

### FileSize

**Syntax:** `FileSize= size`

**Description:** Specifies the maximum size, in bytes, for the LCR EDR files.

**Allowed:**

**Default:** 1024000

**Notes:**

**Example:** `FileSize= 1024000`

### RecordDef

Identifies the LCR EDR section.

**Default:** "NPLCRCDR"

**Allowed:** -

### RemoveNullColumns

**Syntax:** `RemoveNullColumns = true|false`

**Description:** Determines whether or not to include empty fields in the EDR record.

**Allowed:** true, false

**Default:** true

**Notes:**

**Example:** `RemoveNullColumns = true`

`RowHeader`

**Syntax:** `RowHeader = "text"`

**Description:** Specifies the text to be appended to the beginning of each record in the EDR file.

**Allowed:**

**Default:** ""

**Notes:**

**Example:** `RowHeader = ""`

`RowTrailer`

**Syntax:** `RowTrailer = "\char"`

**Description:** Specifies the character to use to separate the EDR records.

**Allowed:**

**Default:** "\n"

**Notes:**

**Example:** `RowTrailer = "\n"`

Chapter 3

# Background Processes

## Overview

### Introduction

This chapter explains the processes that are started automatically by Service Logic Execution Environment (SLEE).

**Note:** This chapter also includes some plug-ins to background processes which do not run independently.

### In this chapter

This chapter contains the following topics.



## cdrIF

### Purpose

cdrIF is a SLEE interface. It manages the EDRs generated from calls processed by your network.

### Startup

This process is started automatically by the SLEE. The following line must be included in the **SLEE.cfg** to start cdrIF:

```
INTERFACE=cdrIF cdrIF /IN/service_packages/NP_SERVICE_PACK/bin EVENT
```

For more information see *SLEE.cfg Configuration* (on page 9).

**Note:** If you do not specify the NP cdrIF interface in **SLEE.cfg**, then the standard cdrIF interface for ACS and CCS will be used for processing the EDRs.

### Location

This process is located on the SLC.

### Configuration

For details on the cdrIF configuration file and the available parameters, see *Configuring EDR Collection* (on page 10).

### Failure

If cdrIF fails, no EDRs will be processed.

## mfw

### Purpose

The MAP Trigger Application (MTA) receives a MAP message from the TCAP Interface. The MAP and SCCP parameters are matched against the configured trigger rules and the first rule to match causes an IDP to be sent to the service key defined in the trigger. An appropriate alarm is raised if there is a problem processing the MAP message.

If a MAP Trigger node is defined then the action configured in the node will be performed. For example the MTA could respond with a MAP response message or relay the MAP message at the SCCP level.

### Startup

The mta is started automatically by the SLEE. The following lines must be included in the **SLEE.cfg** to start the mta:

```
APPLICATION=mtaApplication mta.sh /IN/service_packages/NP_SERVICE_PACK/bin 1 1
SERVICEKEY=INTEGER 0x10800000016 MTA
SERVICEKEY=INTEGER 0x1080000002D MTA
SERVICEKEY=INTEGER 0x10500000047 MTA
SERVICEKEY=INTEGER 0x1050000000F MTA
SERVICEKEY=INTEGER 0x10500000037 MTA
SERVICE=MTA 1 mtaApplication MTA
```

**Note:** The service key details are generated automatically during installation and therefore may be different from this example.

For more information see *SLEE.cfg Configuration* (on page 9).

### Location

The `mta` is located on the SLC.

### mta.cfg configuration file

The **mta.cfg** configuration file defines the list of triggers for the MAP messages received by the mta .

Here is an example **mta.cfg** file.

```
mta = {
    triggers = [
        { name="OLONoNP", servicekey=20 }
        { name="VoiceNP", servicekey=100}
        { name="ACS", servicekey=111 }
    ]
    triggerRules = [
        { msg="MAP_SRI", msisdn="99952044,99952,99904452,88", cdpn="99952,88",
        cdpnoa=4, cgpn="88852,77", cgpnoa=4,
        msisdnSuffixMinLength=2, msisdnSuffixMaxLength=10, trigger="VoiceNP" }
        { msg="MAP_SRI_SM", trigger="OLONoNP" }
        { msg="MAP_SRI,MAP_SRI_SM", trigger="ACS" }
        { msg="MAP_ATI", imsi="333, 444", trigger="ACS" }
        { msg="LOCREQ", dgtsdial="18852044, 19052",
          dgtsDialSuffixMinLength=2, dgtsDialSuffixMaxLength=10,
          trigger="ACS" }
        { msg="SMSREQ", min="788044, 6752", trigger="ACS" }
```

```
        { msg="SMSREQ", imsi="55555", trigger="ACS" }
        { msg="SMSREQ", mdn="123456", mdnSuffixMinLength=2,
          mdnSuffixMaxLength=10, trigger="ACS" }
    ]

    pc_format = "ITU-TS"
    local_pc = "7-243-3"
    local_ssn = 8
    local_gt_digits = "441473123456"
    local_gt_noa = 2
    local_numplan = 255
    local_ri = "GT"

    gttRules = [
        { prefix="44", pc = "7-243-3", ssn = 6,
        ni = true, numPlan = 255, transType = 255,
        minLength=2, maxLength=10,
        remove_chars = 1, add_chars = "44" }
        { prefix="55", pc = "1-255-2", ssn = 2 }
        { prefix="*",  pc = "7-243-3", ssn = 8 }
    ]


    map_response_imsi = [
        { prefix="44", imsi="441473123456" }
        { prefix="*", imsi="441473000000" }
    ]

    normalisationRules =   [
        { prefix = "88", noa = 4, remove_chars = 1, add_chars = "44" }
        { prefix = "44", remove_chars = 2, add_chars = "0" }
        { remove_chars = 2 , add_chars = "0" }
    ]

    in_timeout = 10
    sri_sm_dra_location = "NETWORKNODE_NUMBER"
    map1_sri_sm_dra_location= "MSC_NUMBER"

    abort_code_mapping = [  { code = "0-10", message = "A Critical Error Occurred",
    severity = "CRITICAL" }
         { code = "11", message = "An error occurred", severity = "ERROR" }
         { message = "A minor error occurred", severity = "WARNING" }
    ]
    hop_count = 0
    sccp_loop_compare_digits_only = true
    sccp_loop_compare_pc_only = false
    map_version = 1
    msisdn_sri_noa_override = 1
    msisdn_srism_noa_override = 1
    msisdn_ati_noa_override = 1
    digitsdialled_locreq_noa_override = 1
    mdn_smsreq_noa_override = 1
    msisdn_sri_plan_override = 1
    msisdn_srism_plan_override = 1
    msisdn_ati_plan_override = 1
    prefer_imsi = true
}
```

## Parameters

The mta accepts the following parameters.

### abort_code_mapping

**Syntax:** `abort_code_mapping = [ { code = "alarm_code", message = "alarm_message", severity = "severity_level" } ]`

**Description:** List of abort code number mappings (returned by the Map Trigger Node) to alarm messages and severity levels.

**Type:** Parameter array.

**Optionality:** Optional.

**Allowed:**

**Default:**

**Notes:**

- `code` - The alarm code to match. If the code is not defined, then the rule will match any alarm. A range of values can also be specified.
- `message` (mandatory) - The message to print in the alarm.
- `severity` (mandatory) - The severity of the alarm. Valid values are: NOTICE, WARNING, ERROR, and CRITICAL.

**Example:**

```
abort_code_mapping = [
{ code = "0-10", message = "A critical error occurred",
severity = "CRITICAL" }
]
```

### digitsdialled_locreq_noa_override

**Syntax:** `digitsdialled_locreq_noa_override = noa`

**Description:** Set the nature of address value to use for the digits dialed when forwarding an ANSI-41 LocationRequest with an SCCP relay.

**Type:** Integer

**Optionality:** Optional

**Allowed:** 0 - 255

**Default:**

**Notes:**

**Example:** `digitsdialled_locreq_noa_override = 4`

### gttRules

**Syntax:** `gttRules = [ { prefix = "number", pc = "DRA_pointcode", ssn = number, remove_chars = number, add_chars = "number" } ]`

**Description:** The global title translation rules. At least one GTT rule must be defined. GTT is performed on the digits returned in the destination routing address contained in the CONNECT message from ACS (sent in response to the IDP from the MFW).

**Type:** Parameter group

**Optionality:** Mandatory.

**Allowed:**

**Default:**

**Notes:** The following parameters from the group are mandatory:

- `prefix` - the rule containing the longest prefix match will be fired
- `pc` - the point code of the DRA will be updated with the defined point code (in pc_format), and
- `ssn` - defines the subsystem number for the DRA.

The following parameters are optional:

- `remove_chars` - defines the number of characters to remove from the start of the DRA, and
- `add_chars` - list of characters to add to the start of the DRA after remove_chars has been applied.
- `ni` - set or unset the national indicator in the outgoing message.
- `ri` - set the routing indicator on the outgoing message (this will override the routing indicator value set by the call plan).
- `numPlan` - set the numbering plan of the global title in the outgoing message.
- `trans_Type` - set the translation type of the global title in the outgoing message.

**Example:**

```
gttRules = [
{ prefix="44", pc = "7-243-3", ssn = 6, ni = true, numPlan =
255, transType = 255, remove_chars = 1, add_chars = "44" }
]
```

### hop_count

**Syntax:** `hop_count = hop_count`

**Description:** The SUA hop counter will be set to this value if it is not already defined.

**Type:** Integer

**Optionality:** Optional (default used if not set).

**Default:** 0

**Notes:** If the value of this parameter is zero (default) then a hop counter will not be added.

**Example:**

```
hop_count = 70
```

### in_timeout

**Syntax:** `in_timeout = seconds`

**Description:** The number of seconds to wait for a response from ACS after sending an IDP.

**Type:** Integer.

**Optionality:** Optional (default used if not set).

**Default:** 10

**Example:**

```
in_timeout = 10
```

### local_gt_digits

**Syntax:** `local_gt_digits = "global_title_digits"`

**Description:** The global title digits of the NCC platform.

**Type:** String

**Optionality:** Mandatory unless `local_pc` and `local_ssn` is defined.

**Default:** None

**Notes:** Either both `local_pc and local_ssn`, or `local_gt_digits` must be defined. If required you can define all three parameters.

**Example:**

```
local_gt_digits = "441473123456"
```

### local_gt_noa

**Syntax:** `local_gt_noa = nature_of_address`
**Description:** The nature of address of the global title.
**Type:** Integer
**Optionality:** Optional (uses default if not defined).
**Allowed:**
**Default:** 4
**Notes:**
**Example:** `local_gt_noa = 4`

### local_ni

**Syntax:** `local_ni = true|false`
**Description:** The national indicator of the NCC platform.
**Type:** Boolean.
**Optionality:** Optional (default used if not set).
**Allowed:** true, false
**Default:** true (national indicator bit is set)
**Example:** `local_ni = true`

### local_numplan

**Syntax:** `local_numplan = numbering_plan`
**Description:** The numbering plan of the global title.
**Type:** Integer.
**Optionality:** Optional.
**Default:** None
**Example:** `local_numplan = 10`

### local_pc

**Syntax:** `local_pc = "local_point_code"`
**Description:** The local point code (that is, the PC value of the NCC platform).
**Type:** String
**Optionality:** Mandatory unless either `local_ssn` or `local_gt_digits` is defined. If `local_pc` is defined, then `local_snn` must also be defined.
**Default:** None.
**Notes:** The format used will depend on the `pc_format` configuration option.

Either both `local_pc` and `local_ssn`, or `local_gt_digits` must be defined. If required you can define all three parameters.

**Example:** `local_pc = "7-243-3"`

### local_ri

**Syntax:** `local_ri = "routing_indicator"`
**Description:** The routing indicator of the NCC platform.
**Type:** String.
**Optionality:** Optional (default used if not set).

**Allowed:** PC or GT
**Default:** GT
**Example:** `local_ri = "PC"`

### local_ssn

**Syntax:** `local_ssn = ssn_number`
**Description:** The subsystem number of the NCC platform.
**Type:** Integer
**Optionality:** Mandatory unless `local_gt_digits` is defined. If `local_ssn` is defined, then `local_pc` must also be defined.
**Default:** None.
**Notes:** Either both `local_pc and local_ssn`, or `local_gt_digits` must be defined. If required you can define all three parameters.
**Example:** `local_ssn = 8`

### local_transtype

**Syntax:** `local_transtype = translation_type`
**Description:** The translation type of the global title.
**Type:** Integer.
**Optionality:** Optional.
**Default:** None.
**Example:** `local_transtype = 10`

### map_response_imsi

**Syntax:** `map_response_imsi = [ { prefix="prefix_number", imsi="imsi_number" } ]`
**Description:** List of IMSI values for MAP responses. Maps the SCCP calling party number in the original MAP message (if present) to an IMSI value that is present in any MAP responses sent by mta.
**Type:** Parameter array
**Optionality:** Mandatory.
**Default:** None
**Notes:**
- `prefix` - Triggers the rule that contains the longest prefix match. Prefix matching is performed on the SCCP calling party number in the original MAP message (if present). A default prefix of "*" will match any SCCP CgPN and will also match a message that does not have an SCCP CgPN defined. Only one default prefix can be defined.
- `imsi` - The imsi value to use in any generated MAP responses.

**Example:** `map_response_imsi = [ { prefix="44", imsi="441473123456" } ]`

### map_version

**Syntax:** `map_version = map_version_number`
**Description:** Sets the MAP version to be used if a message is received with no application context.
**Type:** Integer
**Optionality:** Optional (default used if not set).

**Allowed:** 1, 2, or 3
**Default:** 1
**Notes:**
**Example:** `map_version = 1`

### map1_sri_sm_dra_location

**Syntax:** `map1_sri_sm_dra_location = "DRA_fields"`
**Description:** Defines the field in the v1 MAP_SRI_SM response in which to place the destination routing address from the CONNECT message received.
**Type:** String
**Optionality:** Optional (default used if not set).
**Allowed:**
- "ROAMING_NUMBER"
- "MSC_NUMBER"
- "FORWARDED_TO_NUMBER"

**Default:** MSC_NUMBER
**Notes:**
**Example:** `map1_sri_sm_dra_location = "MSC_NUMBER"`

### mdn_smsreq_noa_override

**Syntax:** `mdn_smsreq_noa_override = noa`
**Description:** Set the nature of address value to use for the Mobile Directory Number when forwarding an ANSI-41 SMSRequest with an SCCP relay.
**Type:** Integer
**Optionality:** Optional
**Allowed:** 0 - 255
**Default:**
**Notes:**
**Example:** `mdn_smsreq_noa_override = 4`

### msisdn_ati_noa_override

**Syntax:** `msisdn_ati_noa_override = noa`
**Description:** Set the nature of address value to use for the MSISDN when forwarding a MAP ATI with an SCCP relay.
**Type:** Integer
**Optionality:** Optional
**Allowed:** 0 - 255
**Default:**
**Notes:**
**Example:** `msisdn_ati_noa_override = 4`

### msisdn_ati_plan_override

**Syntax:** `msisdn_ati_plan_override = noa`
**Description:** Set the numbering plan value to use for the MSISDN when forwarding a MAP ATI with an SCCP relay.
**Type:** Integer
**Optionality:** Optional

**Allowed:** 0 - 255
**Default:**
**Notes:**
**Example:** `msisdn_ati_plan_override = 4`

### msisdn_sri_noa_override

**Syntax:** `msisdn_sri_noa_override = noa`
**Description:** Set the nature of address value to use for the MSISDN when forwarding a MAP SRI with an SCCP relay.
**Type:** Integer
**Optionality:** Optional
**Allowed:** 0 - 255
**Default:**
**Notes:**
**Example:** `msisdn_sri_noa_override = 4`

### msisdn_sri_plan_override

**Syntax:** `msisdn_sri_plan_override = noa`
**Description:** Set the numbering plan value to use for the MSISDN when forwarding a MAP SRI with an SCCP relay.
**Type:** Integer
**Optionality:** Optional
**Allowed:** 0 - 255
**Default:**
**Notes:**
**Example:** `msisdn_sri_plan_override = 4`

### msisdn_srism_plan_override

**Syntax:** `msisdn_srism_plan_override = noa`
**Description:** Set the numbering plan value to use for the MSISDN when forwarding a MAP SRI SM with an SCCP relay.
**Type:** Integer
**Optionality:** Optional
**Allowed:** 0 - 255
**Default:**
**Notes:**
**Example:** `msisdn_srism_plan_override = 4`

### msisdn_srism_noa_override

**Syntax:** `msisdn_srism_noa_override = noa`
**Description:** Set the nature of address value to use for the MSISDN when forwarding a MAP SRI SM with an SCCP relay.
**Type:** Integer
**Optionality:** Optional
**Allowed:** 0 - 255

**Default:**

**Notes:**

**Example:** `msisdn_srism_noa_override = 4`

### normalisationRules

**Syntax:** `normalisationRules = [ { prefix = "`*`number`*`", noa = `*`nature_of_address`*`, minLength = `*`number`*`, maxLength = `*`number`*`, remove_chars = `*`number`*`, add_chars = "`*`number`*`" } ]`

**Description:** The normalization rules to apply to the MSISDN. The normalized MSISDN will then be sent in an IDP to the service key defined in the `triggers` parameter. If normalization rules are used then at least one character must be removed or added.

**Type:** Parameter array.

**Optionality:** Optional.

**Allowed:**

**Default:**

**Notes:**

- `prefix` - normalization will be applied to any any MSISDN DigitsDialled or MDN matching the specified prefix. The first prefix to match will be the rule that is applied. The order in which the rules are defined is therefore important.
- `noa` - if an NoA is defined, then the rule will only match if the NoA of the number matches the defined NoA. If no prefix and no NoA are defined then the rule will match any number.
- `minLength` – the rule will only match if the length of the number is equal to or greater than this value.
- `maxLength` – the rule will only match if the length of the number is less than this value.
- `remove_chars` - defines the number of characters to remove from the start of the DRA.
- `add_chars` - list of characters to add to the start of the DRA after remove_chars has been applied.
- either `remove_chars` or `add_chars` or both must be defined.

**Example:**

```
gttRules = [
{ prefix="88", noa = 4, minLength=2, maxLength=10,
remove_chars = 1, add_chars = "44" }
]
```

### pc_format

**Syntax:** `pc_format = "`*`Point_Code_standard`*`"`

**Description:** The point code standard used in the configuration

**Type:** String

**Optionality:** Optional (default used if not set).

**Allowed:** ITU-TS or ANSI

**Default:** ITU-TS

**Notes:** ITU-TS point codes are encoded as a 14-bit structure consisting of a:

- 3-bit zone identification, and 8-bit area/network identification
- 3-bit signaling point identification.

ANSI point codes are encoded in a 24-bit structure consisting of an:

- 8-bit network identification
- 8-bit cluster identification, and
- 8-bit member identification.

**Example:** `pc_format = "ITU-TS"`

### prefer_imsi

**Syntax:** `prefer_imsi = true|false`

**Description:** If this parameter is set to true, a message received containing both an IMSI and a MIN, will have the IMSI stored in the IMSI field of the IDP and the IMSI field of the response.

Else, the MIN will be present in the IMSI field of the IDP and the MIN field of the response.

**Type:** Boolean

**Optionality:** Optional

**Allowed:** true, false

**Default:** true

**Notes:**

**Example:** `msisdn_ati_plan_override = 4`

### sccp_loop_compare_digits_only

**Syntax:** `sccp_loop_compare_digits_only = true|false`

**Description:** When performing an SCCP relay, the MTA checks that the outgoing message has enough differences from the incoming message to make a possible loop unlikely. If this item is set to true and both the incoming and outgoing destination routing indicators are set to global title, then the global title digits will be checked to make sure they are different. When set to false, the whole global title (including encoding scheme and NoA) will be checked.

**Type:** Boolean

**Optionality:** Optional (default used if not set).

**Allowed:** true, false

**Default:** true

**Notes:**

**Example:** `sccp_loop_compare_digits_only = true`

### sccp_loop_compare_pc_only

**Syntax:** `sccp_loop_compare_pc_only = true|false`

**Description:** Used when performing an SCCP relay loop-check. If this item is set to false and the routing indicator on both the incoming and outgoing message is set to point code, then the point code and subsystem number will be checked to ensure that at least one of them has changed. If it is set to true then only the point code will be checked.

**Type:** Boolean

**Optionality:** Optional (default used if not set).

**Allowed:** true, false

**Default:** false

**Example:** `sccp_loop_compare_pc_only = true`

### sri_sm_dra_location

**Syntax:** `sri_sm_dra_location = "DRA_field"`

**Description:** Defines the field in the v2/3 MAP_SRI_SM response in which to place the Destination Routing Address from the CONNECT message received.

**Type:** String

**Optionality:** Optional (default used if not set).

**Allowed:**
- "NETWORKNODE_NUMBER"
- "MSC_NUMBER"
- "BOTH"

**Default:** "NETWORKNODE_NUMBER"

**Notes:**

**Example:**
```
sri_sm_dra_location = "NETWORKNODE_NUMBER"
```

### triggerRules

**Syntax:**
```
triggerRules = [ { msg="map_message_list", msisdn="msisdn",
msisdnSuffixMinLength=minLength, msisdnSuffixMaxLength =
maxLength, dgtsdial="dgtsdial",
dgtsDialSuffixMinLength=minLength,
dgtsDialSuffixMaxLength=maxLength, mdn="mdn",
mdnSuffixMinLength=minLength, mdnSuffixMaxLength=maxLength,
min="min", imsi="imsi", cdpn="list_of_SCCP_cdpns",
cdpnoa="list_of_SCCP_cdpn_noas", cgpn="list_of_SCCP_cdpns>",
cgpnoa="list_of_SCCP_cdpn_noas", trigger="name"}
]
```

**Description:** List of trigger rules against which the MAP and SCCP parameters from the MAP message will be matched. The first rule to match a message will be fired, therefore the order in which the rules are defined is important.

**Type:** Parameter array

**Optionality:** Mandatory.

**Notes:** Within the parameter group:
- `msg` defines a list of map messages that will match this rule.
- `trigger` defines the trigger that will fire if the rule is matched.

The following parameters within the group are optional:
- `msisdn`
- `msisdnSuffixMinLength`
- `msisdnSuffixMaxLength`
- `dgtsdial`
- `dgtsDialSuffixMixLength`
- `dgtsDialSuffixMaxLength`
- `mdn`
- `mdnSuffixMinLength`
- `mdnSuffixMaxLength`
- `min`
- `imsi`
- `cdpn`
- `cgpn`
- `cdpnoa`

- cgpnoa

**Example:**

```
triggerRules = [
{ msg="MAP_SRI", msisdn="99952044,99952,99904452,88",
cdpn="99952,88", cdpnoa=4, cgpn="88852,77", cgpnoa=4,
msisdnSuffixMinLength=2, msisdnSuffixMaxLength=10,
trigger="VoiceNP" }
{ msg="MAP_SRI,MAP_SRI_SM", trigger="ACS" }
{ msg="MAP_ATI", imsi="333, 444", trigger="ACS" }
{ msg="LOCREQ", dgtsdial="18852044, 19052",
  dgtsDialSuffixMinLength=2, dgtsDialSuffixMaxLength=10,
  trigger="ACS" }
{ msg="SMSREQ", min="788044, 6752", trigger="ACS" }
{ msg="SMSREQ", imsi="55555", trigger="ACS" }
{ msg="SMSREQ", mdn="123456", mdnSuffixMinLength=2,
  mdnSuffixMaxLength=10, trigger="ACS" }
]
```

`triggers`

| | |
|---|---|
| **Syntax:** | `triggers = [{ name = "`*`name`*`", servicekey = `*`sknumber`*`}]` |
| **Description:** | List of triggers for MAP messages and the service keys to send the translated IDPs to. |
| **Type:** | Parameter array |
| **Optionality:** | Mandatory. |
| **Notes:** | The value defined for the `name` variable maps to the `trigger` variable in the `triggerRules` parameter array. |
| **Example:** | `triggers = [{ name = "ACS", servicekey = 111}]` |

### Failure

If the mta fails, then alarms will be raised to the syslog.

### Output

The mta writes error messages to the system messages file and writes additional output to **/IN/service_packages/NP_SERVICE_PACK/tmp/mta.log**.

# npMfileCarrierDaemon

### Purpose

This process ensures the carrier data stored in memory is synchronized with the Carrier database table. The daemon ensures that the optimal performance, when querying the database and storing records, is maintained.

### Startup

This process is started by entry scb1 in the inittab, via the shell script:

```
/IN/service_packages/NP_SERVICE_PACK/bin/startCarrierDaemon.sh
```

You can check if the process is running by using the Unix ps command.

To check the process, type:

ps -ef | grep npMfileCarrierDaemon

**Result:** The listed process is the complier process.

**Note:** Placing the npMfileCarrierDaemon startup script in the inittab file ensures that if npMfileCarrierDaemon should die, it will be automatically restarted by the operating system within a few seconds.

### Shutdown

To terminate this process, use the Unix ps command to identify the process number and then kill it manually.

### Location

This process is located on the SLC.

### Parameters

There are no command line parameters for the npMfileRuleDaemon process.

### Failure

If this process fails, alarms will be raised to the syslog and updates to carrier records will not be available in the production system.

### Output

The npMfileCarrierDaemon writes error messages to the system messages file and writes additional output to:

```
/IN/service_packages/NP_SERVICE_PACK/npMfileCarrierDaemon.log
```

## npMfilePQYZDaemon

### Purpose

This process ensures the PQYZ data stored in memory is synchronized with PQYZ data in the database tables. The daemon ensures that the optimal performance, when querying the database and storing records, is maintained.

### Startup

This process is started by entry scb2 in the inittab, via the shell script:

```
/IN/service_packages/NP_SERVICE_PACK/bin/startPQYZDaemon.sh
```

You can check if the process is running by using the Unix ps command.

To check the process, type:

ps -ef | grep npMfilePQYZDaemon

**Result:** The listed process is the complier process.

**Note:** Placing the npMfilePQYZDaemon startup script in the inittab file ensures that if npMfilePQYZDaemon should die, it will be automatically restarted by the operating system within a few seconds.

### Shutdown

To terminate this process, use the Unix ps command to identify the process number and then kill it manually.

### Location

This process is located on the SLC.

### Parameters

There are no command line parameters for the npMfileRuleDaemon process.

### Failure

If this process fails, alarms will be raised to the syslog and updates to PQYZ records will not be available in the production system.

### Output

The npMfilePQYZDaemon writes error messages to the system messages file and writes additional output to:

```
/IN/service_packages/NP_SERVICE_PACK/npMfilePQYZDaemon.log
```

## npMfileRoutingDestinationDaemon

### Purpose

This process ensures the routing destination data stored in memory is synchronized with routing destination data in the database tables. The daemon ensures that the optimal performance, when querying the database and storing records, is maintained.

### Startup

Two instances of this process are started by the inittab. Entry scb3 (the routing destination index entry) and entry scb4 (the routing destinations entry) are started via the following shell scripts:

```
/IN/service_packages/NP_SERVICE_PACK/bin/startRoutingDestinationDaemonIndex.sh
/IN/service_packages/NP_SERVICE_PACK/bin/startRoutingDestinationDaemonDest.sh
```

You can check if the processes are running by using the Unix ps command.

To check the processes, type:

```
ps -ef | grep npMfileRoutingDestinationDaemon
```

**Result:** The listed processes are the complier processes.

**Note:** Placing the npMfileRoutingDestinationDaemon startup scripts in the inittab file ensures that if the npMfileRoutingDestinationDaemon processes should die, they will be automatically restarted by the operating system within a few seconds.

### Shutdown

To terminate this process, use the Unix ps command to identify the process number and then kill it manually.

### Location

This process is located on the SLC.

### Parameters

npMfileRoutingDestinationDaemon accepts the following command line parameters.

| Parameter | Default | Description |
|---|---|---|
| `-oracleuser` *`oracle_user/password`* | - | Specifies the oracle user and password on the SLC.<br>**Example:** `-oracleuser scp/scp` |
| `-gpna` *`mfile_type`* | - | Specifies the type of routing destination mfile. Valid values are:<br>• destination<br>• index |

### Failure

If this process fails, alarms will be raised to the syslog and updates to routing destination records will not be available in the production system.

### Output

npMfileRoutingDestinationDaemon writes error messages to the system messages file and writes additional output to:

```
/IN/service_packages/NP_SERVICE_PACK/npMfileRoutingDestinationDaemon.log
```

## npMfileRuleDaemon

### Purpose

This process ensures the rule data held in memory is synchronized with rule data stored in the database tables. The daemon ensures that the optimal performance, when querying the database and storing records, is maintained.

### Startup

This process is started by entry scb5 in the inittab via the shell script:

```
/IN/service_packages/NP_SERVICE_PACK/bin/startRuleDaemon.sh
```

You can check if the process is running using the Unix ps command.

To check the process, type:

```
ps -ef | grep npMfileRuleDaemon
```

**Result:** The listed process is the complier process.

**Note:** Placing the npMfileRuleDaemon startup script in the inittab file ensures that if the npMfileRuleDaemon processes should die, it will be automatically restarted by the operating system within a few seconds.

## Shutdown

To terminate this process, use the Unix ps command to identify the process number and then kill it manually.

## Location

This process is located on the SLC.

## Parameters

There are no command line parameters for the npMfileRuleDaemon process.

## Failure

If this process fails, alarms will be raised to the syslog and updates to rule records will not be available in the production system.

## Output

npMfileRuleDaemon writes error messages to the system messages file and writes additional output to:

```
/IN/service_packages/NP_SERVICE_PACK/npMfileRuleDaemon.log
```

Chapter 4

# Tools and Utilities

## Overview

### Introduction

This chapter explains the tools and utilities that are available.

### In this chapter

This chapter contains the following topics.



## NP EDRs

### Introduction

The NP Service Pack produces ACS and LCR EDRs, on the SLC, for use in post processing as required.

### EDR collection

Each call processed can produce a single EDR, or multiple EDRs, depending on the type and outcome of the call. As a minimum, each call invokes either an ACS or a CCS service, producing one ACS/CCS EDR for every termination attempt.

Where Least Cost Routing (LCR) is invoked, an LCR EDR is produced for every carrier selected for termination as part of the LCR service logic, in addition to theACS/CCS EDR produced for every termination attempt. This means that the number of LCR EDRs and the number of ACS/CCS EDRs produced for the call is the same.

### NP EDR files

The EDRs are saved to file in a location specified in the **cdrIF.cfg** configuration file. For details, see *Configuring EDR Collection* (on page 10).

EDR files have the following names, depending on the EDR type.

| EDR Type | File Name |
|---|---|
| ACS | ACS_*yyyymmddhhss*_*PID*.cdr |
| LCR | LCR_*yyyymmddhhss*_.cdr |

Where:

- *yyyymmddhhss* = the date and time when the file was opened
- *PID* = the Unix process ID of the service instance that created the EDR file

**Note:** For the LCR EDRs, the file name is configured in the **cdrIF.cfg** and may be different to the format described in this topic.

Later the files are moved by the cmnPushFiles process from each SLC to a configurable location on the SMS. This location is specified in the **cmnPushFilesStartup.sh** script located in the **NP_SERVICE_PACK/bin** directory of each SLC.

The cdr files moved from the SLCs are prefixed with the name of their corresponding SLC, that is, **LCR**_*YYYYMMDDHHMMSS*.**cdr** from 'SLC1' will be renamed to **SLC1_LCR**_*YYYYMMDDHHMMSS*.**cdr** in the SMS.

**Note:** If the location directories for the EDR files are changed manually to something different from the package defaults, the new location directories will have to be manually created in the system and the process that create and move the EDR files (cmnPushFiles and cdrIF) will have to be manually restarted.

## EDR fields

EDRs are saved to file in tag/value pairs, separated by "|", each record separated by a Unix newline character, in the following form:

```
APP|tag1=value1|tag2=value2|...
```

**Note:** The first field in the EDR is not a tag/value pair. It contains the name of the service (either ACS or CCS) that created the EDR. For more information about the format of SLC generated EDRs, see SLC Generated EDRs.

For LDR EDRs, the row trailer (newline) and column separator can be configured in the **cdrIF.cfg** and may be different to the default described in this topic.

## ACS tags

The following ACS tags are generated in the EDR.

- AIDL
- CA
- CAET
- CBAT
- CCET
- CCTS
- CGNA
- CGNN
- CID
- CLI
- CPC
- CPN
- CPNI
- CPNN
- CPPI
- CS
- CUST
- FATS
- HTS
- LAC
- LGID

- LPN (not applicable for NP)
- NOAT
- OA
- OCPI
- OTI
- PCNA (not applicable for NP)
- PTNA (not applicable for NP)
- RELC
- SK
- SN
- TCE
- TCS
- TFN
- TGNA (not applicable for NP)
- TN
- TPNI (not applicable for NP)

**Note:** These are standard tags, as described in the ACS EDR tags topic in *Event Detail Record Reference Guide*.

## Example ACS EDRs

### Example 1

This example shows the output produced for a successful termination attempt EDR.

```
ACS|CID=61080|OA=0|OTI=0|CUST=1|SN=2125551212|TN=2125551212|
CGN=93933301|CLI=3135551212|SK=111|TCS=20051026133312|
TCE=20051026133317|LPN=|LAC=|CS=4|CPC=10|CC=|CPNI=0|PCNA=|TPNI=0|
PTNA=|CGNA=|TGNA=|TFN=ST-1,DDS-5,ATTP-6,ATTP-8,END-3|LGID=0|
CPN=atp|CAET=5|CCET=0.1|CA=2125551212|RELC=16|OCPI=|CPNN=1|
CGNN=4|CPPI=1|NOAT=2|CBAT=1|FATS=0|CCTS=20051026133312|
HTS=20051026133312|AIDL=
```

### Example 2

This example shows the output produced for a failed termination attempt EDR.

```
ACS|CID=61080|OA=0|OTI=0|CUST=1|SN=2125551212|TN=2125551212|
CGN=93933301|CLI=3135551212|SK=111|TCS=20051026133312|
TCE=20051026133312|LPN=|LAC=|CS=0|CPC=10|CC=|CPNI=0|PCNA=|
TPNI=0|PTNA=|CGNA=|TGNA=|TFN=ST-1,DDS-5,ATTP-6|LGID=0|CPN=atp|
CAET=0|CCET=0.0|CA=2125551212|RELC=25|OCPI=|CPNN=1|CGNN=4|
CPPI=1|NOAT=1|CBAT=0|FATS=0|CCTS=20051026133312|
HTS=20051026133312|AIDL=
```

## LCR EDR tags

The following standard ACS tags are generated in the LCR EDR.

- CID
- CLI
- CPN
- CUST
- SK

- SN

**Note:** These are standard tags, as described in the ACS EDR tags topic in *Event Detail Record Reference Guide*.

LCR EDRs do not contain any information on whether the termination attempt was successful or not – this data is stored in the ACS/CCS EDRs.

The LCR EDRs also contain the same CID field that can be used for correlation purposes with ACS/CCS EDRs and with other LCR EDRs.

The following LCR tags are unique to NP.

`CALLINGNUM (lcr set calling number)`

| | |
|---|---|
| **Description:** | The calling number set by the LCR service.<br>This is the number that the service uses as the calling number when an attempt is made to connect the call. |
| **Format:** | Integer. May be up to 32 digits long. |
| **Version:** | NP 2.4.1.1 |
| **Notes:** | |
| **Example:** | `CALLINGNUM=33111111111111` |

`CALLINGNOA (noa of callingnum)`

| | |
|---|---|
| **Description:** | The nature of address of the CALLINGNUM. |
| **Format:** | A single digit |
| **Version:** | NP 2.4.1.1 |
| **Notes:** | |
| **Example:** | `CALLINGNOA=4` |

`CARRIERNAME (carrier name)`

| | |
|---|---|
| **Description:** | The name of the selected carrier. |
| **Format:** | String. May be up to 30 characters long. |
| **Version:** | NP 2.4.1.1 |
| **Notes:** | |
| **Example:** | `CARRIERNAME=Test` |

`CARRIERPOS (position of carrier name in hunt list)`

| | |
|---|---|
| **Description:** | The position of the selected carrier in the hunt list. |
| **Format:** | Integer from 1 to 8. |
| **Version:** | NP 2.4.1.1 |
| **Notes:** | |
| **Example:** | `CARRIERPOS=2` |

`ORIGTRUNK (idp location number content)`

| | |
|---|---|
| **Description:** | Contains contents of location number field from the IDP. |
| **Format:** | integer |
| **Version:** | NP 2.4.1.1 |
| **Notes:** | |
| **Example:** | `ORIGTRUNK=441473` |

### PID (unix process ID)

**Description:** The Unix process ID of the service instance.
**Format:** Integer
**Version:** NP 2.4.1.1
**Notes:**
**Example:** `PID=4355`

### PTI (product type ID)

**Description:** The product type ID for the CCS account type of the calling subscriber.
**Format:** Integer
**Version:** NP 2.4.1.1
**Notes:**
**Example:** `PTI=2`

### ROUTEDEST (routing destination for call)

**Description:** The routing destination for the call.
**Format:** String. May be up to 64 characters long.
**Version:** NP 2.4.1.1
**Notes:**
**Example:** `ROUTEDEST=Destination_3`

### TIME (creation timestamp of lcr edr)

**Description:** The timestamp for when the LCR EDR was created.
**Format:** Date
**Version:** NP 2.4.1.1
**Notes:**
**Example:** `TIME=20051020154857`

### TNNUM (lcr terminating number)

**Description:** The terminating number set by the LCR service (the number that the service attempts to connect to).
**Format:** Number. May be up to 32 digits long.
**Version:** NP 2.4.1.1
**Notes:** The first digit of the number provides an indication of the NOA.
**Example:** `TNNUM=4ABCD12AB987654321`

### TNNOA (noa of terminating number)

**Description:** The nature of address of the terminating number.
**Format:** Integer. A single digit.
**Version:**
**Notes:** This should correspond to the first digit of the TNNUM field.
**Example:** `TNNOA=4`

## Example LCR EDRs

### Example 1

This example shows the output produced for a successful termination attempt EDR.

```
PID=4355|CID=142163|CUST=1|SN=987654321|TNNUM=4ABCD12AB987654321|TNNOA=4|CLI=3311111
1111111|SK=111|CPN=Rob_HR_LCR_plan|PTI=2|TIME=20051020154857|CALLINGNUM=331111111111
11|CALLINGNOA=4|ROUTEDEST=Destination_3|CARRIERNAME=Test Carrier
2|CARRIERPOS=2|ORIGTRUNK=441473
```

### Example 2

This example shows the output produced for a failed termination attempt EDR.

```
PID=4355|CID=142163|CUST=1|SN=987654321|TNNUM=4123412AB987654321|TNNOA=4|CLI=3311111
1111111|SK=111|CPN=Rob_HR_LCR_plan|PTI=2|TIME=20051020154857|CALLINGNUM=331111111111
11|CALLINGNOA=4|ROUTEDEST=Destination_3|CARRIERNAME=Test Carrier
1|CARRIERPOS=1|ORIGTRUNK=441473
```

# prunePortedNumbers.sh

## Purpose

This script runs an SQL procedure to delete obsolete records from the DN range table. It looks for groups of records that have matching Dn start and Dn end dates and purges them according to these rules:

1. All records within the group that have an Activation Date in the past will be deleted except the currently active record (this is the one that is most recently in the past).
2. If the routing number for the currently active record is set to DEAD, then this record will also be deleted.

**Note:** For more information on the DEAD routing number, refer to the *NP Service Pack User's Guide*.

## Location

prunePortedNumbers.sh is located in the **/IN/service_packages/NP_SERVICE_PACK/bin** directory.

## Startup

prunePortedNumbers.sh can be started from the command line by using the command:
`/IN/service_packages/NP_SERVICE_PACK/bin/prunePortedNumbers.sh`

It can also be started by the cron daemon via an entry in a selected user's (such as smf_oper) crontab.

**Tip:** For details on setting up a crontab entry, see *Adding a prunePortedNumbers.sh crontab entry* (on page 38).

## Adding a prunePortedNumbers.sh crontab entry

Follow these steps to add prunePortedNumbers.sh to a crontab entry for the smf_oper user, to automate the purging process.

| Step | Action |
|---|---|
| 1 | Log on to the SMS as smf_oper. |
| 2 | Type in the command: `crontab -e` |
| 3 | Add these lines to the crontab:<br>`0 3 * * * ( . /IN/service_packges/SMS/.profile ; .` |

| Step | Action |
|---|---|
| | `/IN/service_packages/SMS/.profile-sms ;`<br>`/IN/service_packages/NP_SERVICE_PACK/bin/prunePortedNumbers.sh ) >>`<br>`/IN/service_packages/NP_SERVICE_PACK/tmp/prunePortedNumbers.log 2>&1` |
| 4 | Save the changes.<br>**Result:** The purge process will run every day at 3am.<br>**Note:** The actual time used can be changed to suit operational requirements by adjusting the values used for the first two fields of the crontab line. |

### Failure

If this script fails to run then the DN range table will not be purged and obsolete records will not be deleted from the database.

### Output

prunePortedNumbers.sh writes error messages to the system messages file.

# Statistics

### Introduction

NP collects statistics using the standard SMS statistic mechanism and stores them to the SMF database. Please refer to *SMS Technical Guide* for details of how the statistics are collected.

### Statistics collected

This table describes the statistics that are collected.

| Statistic | Description |
|---|---|
| DS1 | This statistic is incremented each time destination selection processing is invoked. |
| DS2 | This statistic is incremented for each internal destination (ported in) request. |
| DS3 | This statistic is incremented each time the requested destination is not found. |
| DS4 | This statistic is incremented for each invalid number. |
| DS5 | This statistic is incremented for each external destination (ported out) request. |
| DS6 | This statistic is incremented for each internal destination (native operator) request. |
| DS7 | This statistic is incremented for each external destination (native operator) request. |
| LCR1 | This statistic is incremented each time a carrier is made available for call routing. |
| LCR2 | This statistic is incremented each time the LCR macro node is invoked. |

Chapter 5

# About Installation and Removal

## Overview

### Introduction

This chapter provides information about the installed components for the Oracle Communications Network Charging and Control (NCC) application described in this guide. It also lists the files installed by the application that you can check for, to ensure that the application installed successfully.

### In this Chapter

This chapter contains the following topics.



## Installation and Removal Overview

### Introduction

For information about the following requirements and tasks, see *Installation Guide*:

- NCC system requirements
- Pre-installation tasks
- Installing and removing NCC packages

### NP Packages

An installation of NP Service Pack includes the following packages, on the:

- SMS:
  - npSms
  - npciSms
  - npPISms
- SLC:
  - npScp

## NP Table Replication

### Introduction

You must replicate the following NP tables to the SLC to complete the installation of the npScp package:

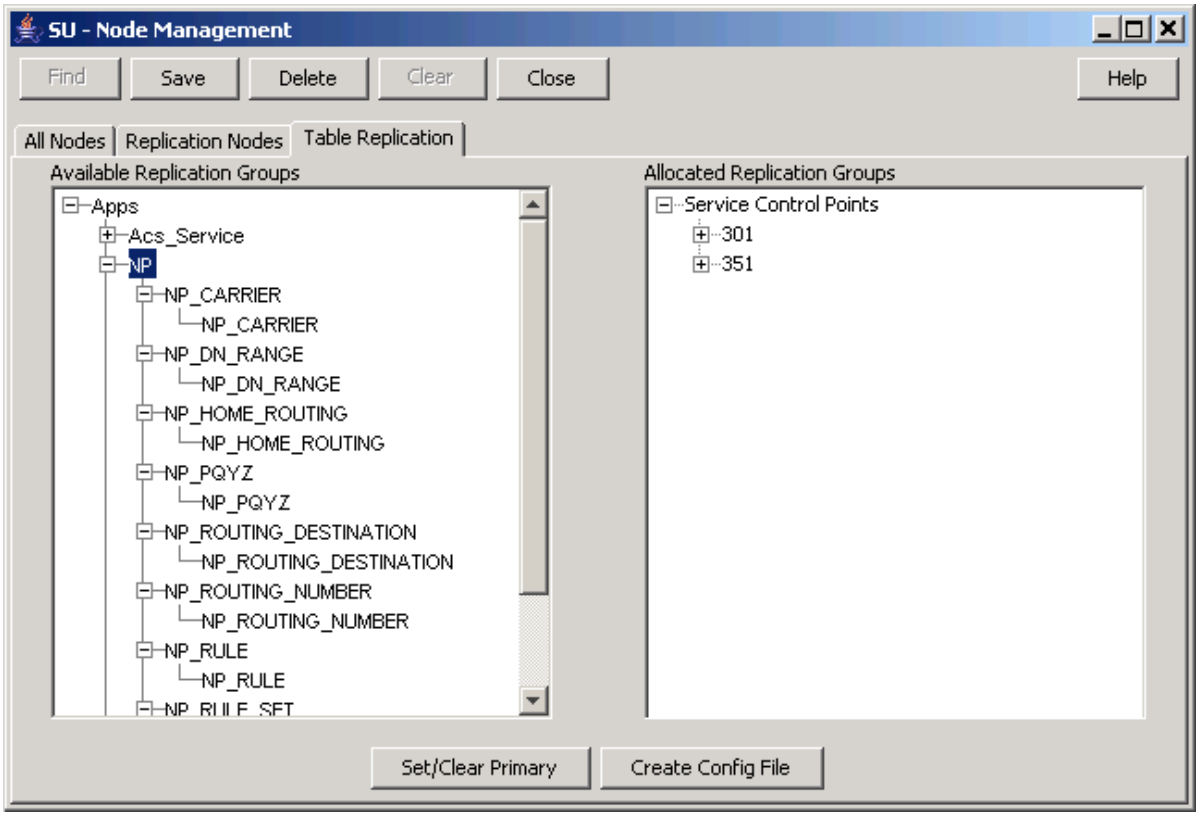
- NP_CARRIER
- NP_DN_RANGE

- NP_HOME_ROUTING
- NP_PQYZ
- NP_ROUTING_DESTINATION
- NP_ROUTING_NUMBER
- NP_RULE
- NP_RULE_SET
- SMF_APPLICATION_ALERT

## Replicating the NP Tables

Follow these steps to replicate the NP tables to the SLC.

| Step | Action |
|---|---|
| 1 | In the SMS, select **Node Management** from the **Operator Functions** menu. |
| 2 | Select the **Table Replication** tab.<br>**Result:** You see the available and allocated replication groups. |




| | |
|---|---|
| 3 | Expand the NP group in the **Available Replication Groups** list. |
| 4 | Click on the table you want to replicate, and drag it to the appropriate SLC node under Service Control Points in the **Allocated Replication Groups** list. |
| | **Note:** You can check which nodes are SLC nodes using Find and Search on the **Replication Nodes** tab. |
| 5 | Repeat step 4 for all the tables you want to replicate. |
| 6 | Click **Save**. |
| 7 | Click **Create Config File**. |

| Step | Action |
|---|---|
| 8 | Check the database on the SLC to ensure the data has been replicated. |

# Checking the Installation

## Introduction

Refer to these check lists to ensure the NP Service Pack has been installed correctly.

## NP Database Tables - SMS

The following tables should exist on the NP SMF database on the SMS:

- NP_CARRIER
- NP_DN_RANGE
- NP_HOME_ROUTING
- NP_PQYZ
- NP_ROUTING_DESTINATION
- NP_ROUTING_NUMBER
- NP_RULE
- NP_RULE_SET
- SMF_APPLICATION

## NP database tables - SCP

The following tables should exist on the SCP database on the SLC:

- NP_CARRIER
- NP_DN_RANGE
- NP_HOME_ROUTING
- NP_PQYZ
- NP_ROUTING_DESTINATION
- NP_ROUTING_NUMBER
- NP_RULE
- NP_RULE_SET
- SMF_APPLICATION

## NP directories and files

The NP installation creates the following directories:

```
/IN/service_packages/NP_SERVICE_PACK/bin
/IN/service_packages/NP_SERVICE_PACK/db/common
/IN/service_packages/NP_SERVICE_PACK/db/MacroNodes
/IN/service_packages/NP_SERVICE_PACK/db/LCR
/IN/service_packages/NP_SERVICE_PACK/db/HR
/IN/service_packages/NP_SERVICE_PACK/db/DS
/IN/service_packages/NP_SERVICE_PACK/etc
/IN/service_packages/NP_SERVICE_PACK/lib
/IN/service_packages/NP_SERVICE_PACK/tmp
```

The NP installation installs the following binaries and interfaces:

```
/IN/services_packages/NP_SERVICE_PACK/bin/cdrIF
/IN/services_packages/NP_SERVICE_PACK/bin/npMfileCarrierDaemon
/IN/services_packages/NP_SERVICE_PACK/bin/npMfileCarrierDaemon.sh
/IN/services_packages/NP_SERVICE_PACK/bin/npMfilePQYZDaemon
/IN/services_packages/NP_SERVICE_PACK/bin/npMfilePQYZDaemon.sh
/IN/services_packages/NP_SERVICE_PACK/bin/npMfileRoutingDestinationDaemon
/IN/services_packages/NP_SERVICE_PACK/bin/npMfileRoutingDestinationDaemonDest.sh
/IN/services_packages/NP_SERVICE_PACK/bin/npMfileRoutingDestinationDaemonIndex.sh
/IN/services_packages/NP_SERVICE_PACK/bin/npMfileRuleDaemon
/IN/services_packages/NP_SERVICE_PACK/bin/npMfileRuleDaemon.sh
/IN/services_packages/NP_SERVICE_PACK/bin/prunePortedNumbers.sh
```

The NP installation installs the following example configuration files:

```
/IN/services_packages/NP_SERVICE_PACK/etc/SLEE.cfg
/IN/services_packages/NP_SERVICE_PACK/etc/acs.cfg
/IN/services_packages/NP_SERVICE_PACK/etc/cdrIF.cfg
/IN/services_packages/NP_SERVICE_PACK/etc/slee_acs_NP.cfg
/IN/services_packages/NP_SERVICE_PACK/etc/mta.cfg
```

The NP installation installs the following shared libraries:

```
/IN/services_packages/NP_SERVICE_PACK/lib/libNpCpuChassisActions.so
/IN/services_packages/NP_SERVICE_PACK/lib/libNpSpecificMacroNodeLoader.so
```

# Oracle Configuration

## Procedure initSCP.ora

Follow these steps to configure the **initSCP.ora** file.

| Step | Action |
|---|---|
| 1 | Type on the command line:<br>`cd $ORACLE_HOME/dbs` |
| 2 | Type `vi initSCP.ora` to start an editing session. |
| 3 | Add or edit the following lines:<br>`CURSOR_SHARING = EXACT`<br>`QUERY_REWRITE_ENABLED = TRUE` |
| 4 | Save **initSCP.ora** and exit. |
| 5 | Restart the Oracle database for these settings to take effect. |