

Oracle® Communications Network Charging and Control

EDR Control Agent Technical Guide



Release 15.1

April 2025

The Oracle logo, consisting of the word "ORACLE" in white, uppercase, sans-serif font, centered within a solid red square.

Copyright

Copyright © 2025, Oracle and/or its affiliates.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software" or "commercial computer software documentation" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

About This Document	v
Document Conventions	vi
Chapter 1	
System Overview	1
Overview	1
Introduction to EDR Control Agent	1
Statistics and Reports	4
Chapter 2	
Configuration	7
Overview	7
Configuration Overview	7
eserv.config Configuration	7
Chapter 3	
Background Processes	11
Overview	11
edrControlAgent	11
Chapter 4	
Troubleshooting	23
Overview	23
Possible Problems	23
Chapter 5	
About Installation and Removal	25
Overview	25
Installation and Removal Overview	25
Checking the Installation	25
Post-installation Configuration	26

About This Document

Scope

The scope of this document includes all the information required to install, configure, and administer the Oracle Communications Network Charging and Control EDR Control Agent.

Audience

This guide was written primarily for system administrators and persons installing, configuring, and administering the EDR Control Agent application. However, sections of the document may be useful to anyone requiring an introduction to the application.

Prerequisites

A solid understanding of UNIX and a familiarity with IN concepts are an essential pre-requisite 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.

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

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:

- *Service Logic Execution Environment Technical Guide*
- *Advanced Control Services Technical Guide*
- *Service Management System User's Guide*
- *Service Management System 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. Names of database tables and fields.
<i>Italics</i>	Name of a document, chapter, topic or other publication. Emphasis within text.
Button	The name of a button to click or a key to press. 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. Example: Ctrl+P or Alt+F4 .
Monospace	Examples of code or standard output.
Monospace Bold	Text that you must enter.
<i>variable</i>	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. 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.

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.

Introduction to EDR Control Agent	1
Statistics and Reports	4

Introduction to EDR Control Agent

Purpose

The Oracle Communications Network Charging and Control EDR Control Agent (ECA) is a SLEE interface that takes EDRs and translates them into InitialDPs (IDPs). These IDPs can be used to trigger control plans on an SLC. This enables a user to use additional functionality provided by call processing (such as control plans) to process information collected in EDRs from switches or other sources.

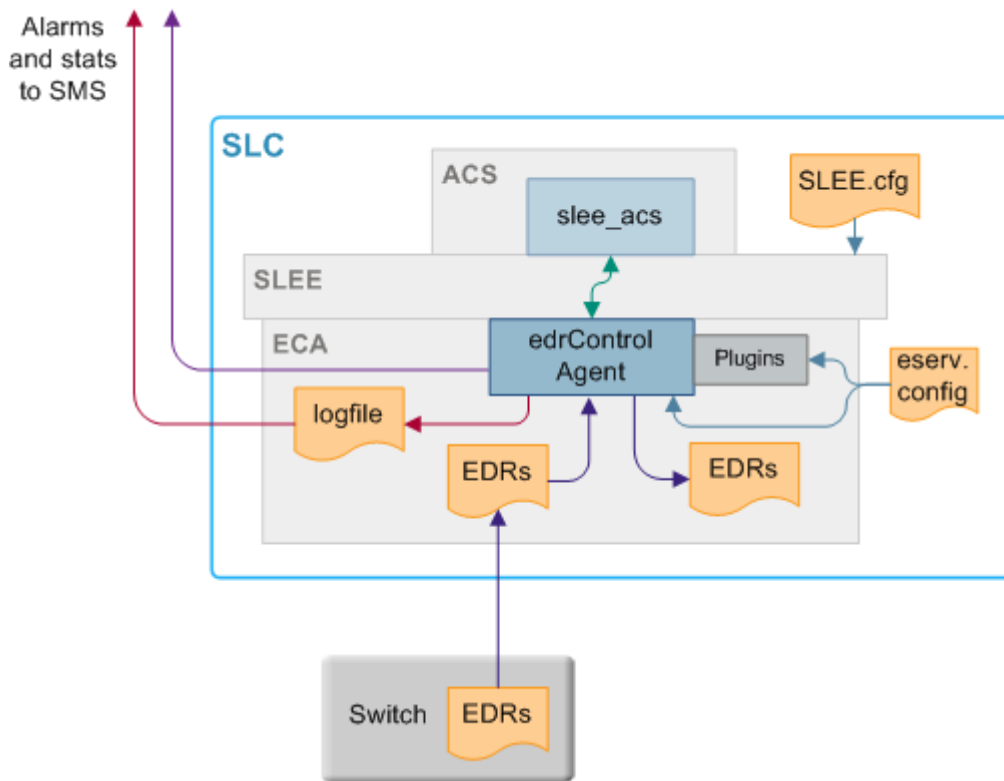
Example solution

One possible use of the ECA is to check if a subscriber is using his or her SIM card with its original handset. This can be achieved by sending the IMEI to an ACS control plan.

ECA reads the IMEI from the EDR and sends it to `slee_acs` in an InitialDP in the `tbcdExtensionstring` field. The control plan then uses a Profile Field Comparison node to compare the relevant extension call context value with a profile field holding the subscriber's original IMEI. If it matches, a discount or a bonus can be applied to the account.

Component diagram

This diagram shows the ECA components.



Component descriptions

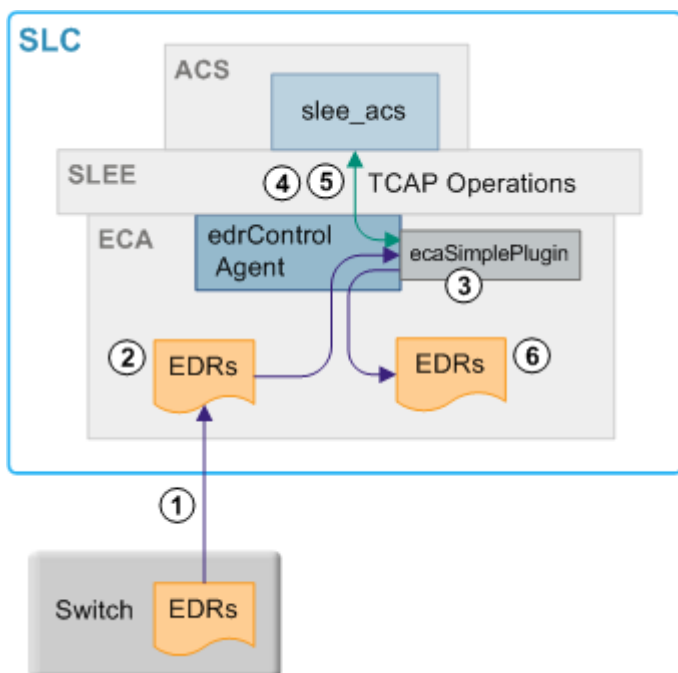
This table describes the main components involved in the ECA application.

Component	Description	Further Information
edrControlAgent	Main ECA executable, responsible for all EDR processing. Opens and parses EDRs in the input directory and moves EDRs into success or failed directories depending on plug-in processing. Uses plug-in libraries to process specific EDR types.	<i>edrControlAgent</i> (on page 11)
edrSimplePlugin.so	Basic plug-in for edrControlAgent.	<i>edrSimplePlugin.so</i> (on page 15)
SLEE	The SLEE routes calls between applications and interfaces.	<i>Service Logic Execution Environment Technical Guide</i>
SLEE.cfg	Main configuration file for SLEE. Provides some startup configuration for ECA, including number of instances.	<i>Service Logic Execution Environment Technical Guide</i>
IN Applications	This could be any call processing component, but is usually <code>slee_acs</code> .	<i>Advanced Control Services Technical Guide</i>
Switch	One of many possible sources of EDRs.	

Component	Description	Further Information
eserv.config	Main configuration file for ECA.	eserv.config <i>Configuration (on page 7)</i>

EDR processing diagram

This diagram shows ECA processing.



EDR processing

This table describes the stages involved in processing EDRs.

Note: This process uses `slee_acs`. However, other compatible call processing software could be used.

Step	Action
1	EDR files arrive in the input directory.
2	<code>edrControlAgent</code> moves the EDRs from the input directory to the processing directory.
3	<code>edrControlAgent</code> passes the EDR file to its plug-in. The plug-in parses EDRs from the EDR file.
4	For each EDR in the EDR file, the <code>edrControlAgent</code> plug-in generates a TCAP BEGIN containing an InitialDP and call duration that the <code>edrControlAgent</code> sends to <code>slee_acs</code> across the SLEE. The plug-in populates the InitialDP using specific configuration (such as <i>idpParameters</i> (on page 16)). For more information about: - SLEE configuration, see <i>NCC Service Logic Execution Environment Technical Guide</i> <code>slee_acs</code>, see <i>Advanced Control Services Technical Guide</i>
5	<code>slee_acs</code> runs the relevant control plan. This may involve further TCAP operations being passed between <code>slee_acs</code> and <code>edrControlAgent</code> .

Step	Action
6	When the call is over (due to an appropriate TCAP operation or a timeout), the <code>edrControlAgent</code> 's plug-in closes the file. If the EDRs were successful, it moves the file to the success directory. Any EDRs that could not be parsed are written to a separate file in the failed directory.

Supported InitialDP operations

`edrControlAgent` uses CAP 3 INAP in messages to `slee_acs`. These INAP operations are supported:

- InitialDP
- ApplyChargingReport
- EventReportBCSM
- RequestReportBCSMEvent
- ApplyCharging
- Connect
- Continue
- ReleaseCall

Unsupported InitialDP parameters

All other INAP operations are not supported, including:

- ConnectToResouce
- EstablishTemporaryConnection
- PlayAnnouncement
- PromptAndCollectUserInformation
- DisconnectLeg
- CallInformationRequest

Warning: You must set up the control plans, the `acs.conf` file, and `eserv.config` file so that these and other unsupported operations are not received by `edrControlAgent` in EDRs.

Statistics and Reports

Introduction

`edrControlAgent` logs statistics to the Service Management System statistics subsystem if configured to do so in `eserv.config`. `edrControlAgent` uses Application ID ECA.

For more information about the SMS statistics subsystem, see *Service Management System Technical Guide* and *Service Management System User's Guide*.

Statistics

This table describes the statistics that can be logged by `edrControlAgent`.

Statistic	Description
INITIALDPS_SENT	Number of InitialDP operations sent.
CALL_PROCESSING_FAILED	Number of InitialDPs for which processing failed.
APPLY_CHARGING_CONNECT_SUCCESSFUL	Number of calls where AC-Connect received.

Statistic	Description
APPLY_CHARGING_CONTINUE_SUCCESSFUL	Number of calls where AC-Continue received.
APPLY_CHARGING_CONNECT_INSUFFICIENT_FUNDS	Number of calls where (AC-Connect) duration is greater than granted time.
APPLY_CHARGING_CONTINUE_INSUFFICIENT_FUNDS	Number of calls where (AC-Continue) duration is greater than granted time.
RELEASE_CALL	Number of calls where a (ReleaseCall) was received. Detail = cause.
CONNECT	Number of calls where Connect received.
CONTINUE	Number of calls where Continue received.
TIMED_OUT	Number of calls for which the tssf timer expired. For more information about the tssf timer, see <i>tssf</i> (on page 15).

Reports

ECA does not install any specific reports. However, you can report on any statistics that are recorded using the SMS Application report on the Service Management System Report Functions screen.

For more information about running SMS reports, see *Service Management System User's Guide*.

Report example

This text shows an example of the SMS Application report run for ECA for the previous 24 hours showing Total counts for each statistic.

```
Application Statistics Listing
=====
```

```
Hours since: 24
Application: ECA
Report Type: Totals
```

```
08 January 2008, 02:27:35
```

Node Name	Statistics ID	Totals
-----	-----	-----
prodscpl	CONNECT	0
prodscpl	CONTINUE	106
prodscpl	TIMED_OUT	1200
prodscpl	RELEASE_CALL	1209
prodscpl	INITIALDPS_SENT	6102
prodscpl	CALL_PROCESSING_FAILED	91
prodscpl	APPLY_CHARGING_CONNECT_SUCCESSFUL	3192
prodscpl	APPLY_CHARGING_CONTINUE_SUCCESSFUL	0
prodscpl	APPLY_CHARGING_CONNECT_INSUFFICIENT_FUNDS	79
prodscpl	APPLY_CHARGING_CONTINUE_INSUFFICIENT_FUNDS	0

```
10 rows selected.
```

```
Completed
```


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	7
eserv.config Configuration.....	7

Configuration Overview

Introduction

This topic provides a high-level overview of how you configure EDR Control Agent.

Configuration components

You configure ECA by using the following components:

Component	Locations	Description	Further Information
eserv.config	SLCs	The most important configuration file. It configures most NCC applications, including ECA processes. Use the ECA section of eserv.config to configure ECA.	eserv.config Configuration (on page 7).
SLEE.cfg	SLCs	Sets details about how the ECA runs, including number of instances.	<i>Service Logic Execution Environment Technical Guide</i>

eserv.config Configuration

Introduction

The **eserv.config** file is a shared configuration file, from which many Oracle Communications Network Charging and Control (NCC) applications read their configuration. Each NCC machine (SMS, SLC, and VWS) has its own version of this configuration file, containing configuration relevant to that machine. The **eserv.config** file contains different sections; each application reads the sections of the file that contains data relevant to it.

The **eserv.config** file is located in the **/IN/service_packages/** directory.

The **eserv.config** file format uses hierarchical groupings, and most applications make use of this to divide the options into logical groupings.

Configuration File Format

To organize the configuration data within the **eserv.config** file, some sections are nested within other sections. Configuration details are opened and closed using either **{ }** or **[]**.

- Groups of parameters are enclosed with curly brackets – **{ }**
- An array of parameters is enclosed in square brackets – **[]**
- Comments are prefaced with a **#** at the beginning of the line

To list things within a group or an array, elements must be separated by at least one comma or at least one line break. Any of the following formats can be used, as in this example:

```
{ name="route6", id = 3, prefixes = [ "00000148", "0000473" ] }
{ name="route7", id = 4, prefixes = [ "000001049" ] }
```

or

```
{ name="route6"
  id = 3
  prefixes = [
    "00000148"
    "0000473"
  ]
}
{ name="route7"
  id = 4
  prefixes = [
    "000001049"
  ]
}
```

or

```
{ name="route6"
  id = 3
  prefixes = [ "00000148", "0000473" ]
}
{ name="route7", id = 4
  prefixes = [ "000001049" ]
}
```

ECA **eserv.config** example section

ECA adds a cut-down **ECA** section to **eserv.config** file. It is not a full list of all parameters that are available.

The **ECA** section contains initial values that you may need to amend to suit a specific installation. After amended, **ECA** runs with no further changes to Oracle. Where additional implementation changes need to be made to Oracle, refer to the *Background Processes* (on page 11) chapters for full descriptions of all process parameters.

Editing the File

Open the configuration file on your system using a standard text editor. Do not use text editors, such as Microsoft Word, that attach control characters. These can be, for example, Microsoft DOS or Windows line termination characters (for example, ^M), which are not visible to the user, at the end of each row. This causes file errors when the application tries to read the configuration file.

Always keep a backup of your file before making any changes to it. This ensures you have a working copy to which you can return.

Loading `eserv.config` Changes

If you change the configuration file, you must restart the appropriate parts of the service to enable the new options to take effect.

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.

edrControlAgent 11

edrControlAgent

Purpose

edrControlAgent processes InitialDPs and sends them to an application across the SLEE. The edrControlAgent uses separate plugins to process different types of EDRs. It can only run one plugin per instance.

For more information about the SLEE, see *Service Logic Execution Environment Technical Guide*.

Startup

This task is started by the SLEE, by the following line in the **SLEE.cfg** configuration file:

```
INTERFACE=edrControlAgent eca.sh /IN/service_packages/ECA/bin 1 EVENT
```

Note: The above are defaults and may vary.

Configuration

To load and operate, the edrControlAgent reads the ECA section of the **eserv.config** file. The high-level structure of the ECA section is shown below.

```
ECA = [
{
    sleeInterfaceName = "uniqueID"
    inputDirectory = "dir"
    processingDirectory = "dir"
    badFileDirectory = "dir"
    fileNamePattern = "pattern"
    sleeServiceKey = key
    maxIdpsPerSecond = seconds
    statisticsEnabled = true|false
    tssf = seconds
    pluginLibrary = "lib"
```

```

    PluginSpecificConfig = {
        inapServiceKey = key
        commentChar = "str"
        separatorChar = "str"
        idpParameters = [
            "str"
            ...
        ]
        NumberRules = [
            { [remove=int][, replace=str] }
            ...
        ]
        successDirectory = "dir"
        failedDirectory = "dir"
    }
}
{
    additional_ECA_instances
}
...
]

```

Parameters

Parameters of the `edrControlAgent` are listed below.

`badFileDirectory`

Syntax:	<code>badFileDirectory = "dir"</code>
Description:	<code>edrControlAgent</code> moves EDR files that it cannot open for processing to this directory.
Type:	String
Optionality:	Required
Default:	<code>/IN/service_packages/ECA/failed</code>
Example:	<code>badFileDirectory = "/IN/service_packages/ECA/failed"</code>

`fileNamePattern`

Syntax:	<code>fileNamePattern = "pattern"</code>
Description:	The pattern that the <code>edrControlAgent</code> uses to identify EDR files in the input directory.
Type:	String
Optionality:	Required
Default:	<code>fileNamePattern = "*.edr"</code>
Notes:	Use the parameter to: • Ensure only EDR files are processed (while other files, such as README files, are left alone) • Enable a specific <code>edrControlAgent</code> instance to select specific EDR files from a common input directory
Example:	<code>fileNamePattern = "*.edr"</code>

`inputDirectory`

Syntax:	<code>inputDirectory = "dir"</code>
Description:	The full path of the directory that holds EDRs to be processed by <code>edrControlAgent</code> .
Type:	String

Optionality: Mandatory
Default: /IN/service_packages/ECA/input
Example: inputDirectory = "/IN/service_packages/ECA/input"

maxIdpsPerSecond

Syntax: maxIdpsPerSecond = *num*
Description: The maximum number of IDPs that edrControlAgent sends across the SLEE within one second.
Type: Integer
Optionality: Optional, default used if not set
Allowed: Positive integers.
 If set to 0 (zero), no throttling is performed.
Default: 0
Example: maxIdpsPerSecond = 250

pluginLibrary

Syntax: pluginLibrary = "*lib*"
Description: The ECA plug-in that processes EDRs of a specific type.
Type: String
Optionality: Optional, default used if not set
Default: ECASimplePlugin.so
Notes: You can only have one plug-in per instance. To run two plug-ins, you must have two edrControlAgents configured in the ECA section of **eserv.config**.
Example: pluginLibrary = "ECASimplePlugin.so"

PluginSpecificConfig

Syntax: PluginSpecificConfig = { *config* }
Description: Configuration for the library specified in pluginLibrary (on page 13).
Type: Array
Optionality: Required
Notes: The detail of this array depends on the plug-in.
Example:

```

PluginSpecificConfig = {
  inapServiceKey = 900
  commentChar = "#"
  separatorChar = "|"
  idpParameters = [
    "calledPartyNumber"
    "skip"
    "callingPartyNumber"
    "locationNumber"
    "tbcdExtension 1234"
    "asn1Extension 1234"
    "callDurationDeciseconds"
  ]
  NumberRules = [
    { prefix = "6449", min = 5, remove = 2,
      prepend = "0", resultNoa = 2 }
  ]
}
```

```

    successDirectory =
"/IN/service_packages/ECA/success"
    failedDirectory =
"/IN/service_packages/ECA/failed"
}

```

processingDirectory

Syntax: `processingDirectory = "dir"`

Description: Before processing the EDR files, `edrControlAgent` moves the files to the specified directory.

Type: String

Optionality: Required

Default: `/IN/service_packages/ECA/processing`

Notes: You can use this parameter to enable different instances of `edrControlAgent` to run on the same machine, while using the same input directory.
If there are multiple instances of ECA, then processing directory should be unique for each ECA instance to prevent contention.

Example: `processingDirectory = "/IN/service_packages/ECA/processing"`
`processingDirectory = "/IN/service_packages/ECA/processing2"`

sleeInterfaceName

Syntax: `sleeInterfaceName = "name"`

Description: The unique identifier of this instance of the `edrControlAgent` interface in **SLEE.cfg**.

Type: Integer

Optionality: Required

Default: `edrControlAgent`

Notes: Must match the unique identifier of `edrControlAgent` in the **SLEE.cfg** file.
For more information about **SLEE.cfg**, see *Service Logic Execution Environment Technical Guide*.

Example: `sleeInterfaceName = "edrControlAgent"`

sleeServiceKey

Syntax: `sleeServiceKey = skey`

Description: The service key to use in outgoing InitialDPs.

Type: Integer

Optionality: Required

Default: 900

Notes: Must match the SERVICEKEY entry for the `edrControlAgent` (`slee_acs` on install) in **SLEE.cfg**. For more information about **SLEE.cfg**, see *Service Logic Execution Environment Technical Guide*.
Set to 900 at installation.

Example: `sleeServiceKey = 900`

statisticsEnabled

Syntax: `statisticsEnabled = true|false`

Description: Defines whether `edrControlAgent` logs statistics about its processing to the SMS statistics sub-system.

Type: Integer

Optionality: Optional, default used if not set.
Allowed: true, false
Default: true
Notes:
Example: `statisticsEnabled = false`

`tssf`

Syntax: `tssf = seconds`
Description: The number of seconds `edrControlAgent` waits for a response before abandoning the IDP and closing the dialog.
Type: Integer
Optionality: Optional, default used if not set.
Allowed:
Default: 30
Notes: If the plug-in requires it, `edrControlAgent` increments a statistic count when a message times out.
Example: `tssf = 10`

edrSimplePlugin.so

The `PluginSpecificConfig` section of the ECA `eserv.config` configuration supports these parameters for `edrSimplePlugin.so`.

```

PluginSpecificConfig = {
    inapServiceKey = key
    commentChar = "str"
    separatorChar = "str"
    idpParameters = [
        "str"
        ...
    ]
    NumberRules = [
        { [prefix="str", ][min=int, ][max=int, ] remove=int[, replace=str][,
        prepend="str"][, resultNoa=int] }
        ...
    ]
    successDirectory = "dir"
    failedDirectory = "dir"
}

```

The parameters are described in detail below.

`commentChar`

Syntax: `commentChar = "char"`
Description: The character that signifies comments, which are not processed.
Type: String
Optionality: Optional, default will be used if not set.
Allowed: ASCII characters
Default: `"#"`
Notes:
Example: `commentChar = "#"`

failedDirectory

Syntax:	<code>failedDirectory = "dir"</code>
Description:	When an EDR fails parsing, <code>edrControlAgent</code> writes the EDR to the specified directory.
Type:	String
Optionality:	Mandatory
Allowed:	
Default:	None
Notes:	<code>edrControlAgent</code> writes EDRs that are parsed successfully to the directory specified by the <i>successDirectory</i> (on page 17) parameter.
Example:	<code>failedDirectory = "/var/edr/failed"</code>

idpParameters

Syntax:	<code>idpParameters = ["parameters"]</code>
Description:	The order of fields in the EDR.
Type:	Array of strings
Optionality:	Mandatory
Allowed:	For information about the fields you can use, see <i>idpParameter details</i> (on page 16). If fields other than those specified are used, <code>edrControlAgent</code> does not startup.
Default:	None
Notes:	The separator in the EDR file is specified by the <i>separatorChar</i> (on page 17) parameter.
Example:	<pre>idpParameters = ["calledPartyNumber" "skip" "callingPartyNumber" "locationNumber" "tbcdExtension 1234" "callDurationSeconds"]</pre>

This configuration correctly parses the following EDR:

```
123456789|skip|987654321|123456|490154203237518|123
```

idpParameter details

You can specify the following strings for `idpParameters`:

String	Description
<code>skip</code>	The field in this position is irrelevant. Skip it.
<code>callDurationSeconds</code>	Call duration in seconds
<code>callDurationDeciseconds</code>	Call duration in deciseconds
<code>tbcdExtension(<int extension type>)</code>	An IDP extension with the given type, as an octet string. The octet string contains a coded up MAP TBCD-STRING.
<code>calledPartyNumber</code>	IDP.calledPartyNumber
<code>callingPartyNumber</code>	IDP.callingPartyNumber
<code>callingPartysCategory</code>	IDP.callingPartysCategory
<code>locationNumber</code>	IDP.locationNumber
<code>additionalCallingPartyNumber</code>	IDP.additionalCallingPartyNumber

String	Description
redirectingPartyID	IDP.redirectingPartyID
iMSI	IDP.iMSI
mscAddress	IDP.mscAddress
calledPartyBCDNumber	IDP.calledPartyBCDNumber

inapServiceKey

Syntax:	<code>inapServiceKey = <i>skey</i></code>
Description:	The service key for all IDPs generated by this plug-in.
Type:	Integer
Optionality:	Mandatory
Allowed:	Valid IDPs as specified in the relevant standard.
Default:	None
Notes:	To run some EDRs on a different service key, you must run more than one instance of <code>edrControlAgent</code> and split the EDRs for each process.
Example:	<code>inapServiceKey = 111</code>

separatorChar

Syntax:	<code>separatorChar = "<i>char</i>"</code>
Description:	The character that separates the fields in the EDR.
Type:	String
Optionality:	Optional, default will be used if not set.
Allowed:	ASCII
Default:	
Notes:	
Example:	<code>separatorChar = " "</code>

successDirectory

Syntax:	<code>successDirectory = "<i>dir</i>"</code>
Description:	<code>edrControlAgent</code> moves EDRs that are processed successfully to the specified directory.
Type:	String
Optionality:	Mandatory
Allowed:	
Default:	None
Notes:	
Example:	<code>successDirectory = "/var/edr/success"</code>

NumberRules

The `NumberRules` subsection of the ECA `eserv.config` configuration supports these parameters. If the `NumberRules` section is not present, all numbers are assumed to be in international format. This section applies to denormalization only.

```
NumberRules = [
```

```

    { [prefix="str", ][min=int, ][max=int, ][remove=int[, replace=str][,
    prepend="str"][, resultNoa=int] }
    ...
]

```

The parameters are described in detail below.

max

Syntax:	<code>max = <i>maxNoLength</i></code>
Description:	The maximum number of digits that a number can contain. If the number contains digits less than or equal to this value, the max part of the number rule is met.
Type:	Integer
Optionality:	Optional, (if not set, default it used).
Allowed:	
Default:	999
Notes:	This parameter is an element of the <code>NumberRules</code> parameter array.
Example:	<code>max = 9</code>

min

Syntax:	<code>min = <i>minNoLength</i></code>
Description:	The minimum number of digits that a number can contain. If the number contains digits that is greater than or equal to this value, the min part of the number rule is met.
Type:	Integer
Optionality:	Optional, if not set default is used.
Allowed:	$0 \leq \text{minNoLength}$
Default:	0
Notes:	- The <code>remove</code> parameter affects the <code>min</code> parameter. If <code>remove</code> is equal to <code>noOfRemovedDigits</code> and <code>noOfRemovedDigits</code> is greater than 0, then <code>minNoLength</code> must be set so <code>minNoLength</code> is greater than or equal to <code>noOfRemovedDigits</code>. - This parameter is an element of the <code>NumberRules</code> parameter array.
Example:	<code>min = 5</code>

prefix

Syntax:	<code>prefix = "prefix"</code>
Description:	Contains a digit or digits. Rule attempts to match the first digit or digits of a number with this value. If the digit or digits match, the prefix part of the number rule is met.
Type:	String
Optionality:	Optional
Allowed:	One or more decimal digits.
Default:	
Notes:	This parameter is an element of the <code>NumberRules</code> parameter array.
Example:	<code>prefix = "25"</code>

prepend

Syntax:	<code>prepend = "firstDigits"</code>
Description:	Digits added to the beginning of a number.

Type:	String
Optionality:	Optional
Allowed:	Can be: - Any combination of decimal digits - A null string ("")
Default:	
Notes:	- If the <code>remove</code> and <code>prepend</code> parameters are both used in the same number rule, <i>firstDigits</i> is added to the beginning of the number after the number has been modified by the <code>remove</code> parameter. - This parameter is an element of the <code>NumberRules</code> parameter array.
Example:	<code>prepend = "0"</code>
<code>remove</code>	
Syntax:	<code>remove = noOfRemovedDigits</code>
Description:	The number of digits stripped from the beginning of a number.
Type:	Integer
Optionality:	Mandatory
Allowed:	
Default:	
Notes:	- The <code>remove</code> parameter affects the <code>min</code> parameter. If <code>min</code> is equal to <i>minNoLength</i> and if <i>noOfRemovedDigits</i> is greater than 0, then <i>minNoLength</i> must be set to <i>minNoLength</i> is greater than or equal to <i>noOfRemovedDigits</i>. See <i>min</i> (on page 18). - The <code>remove</code> parameter is an element of the <code>NumberRules</code> parameter array.
Example:	<code>remove = 2</code>
<code>replace</code>	
Syntax:	<code>replace = "str"</code>
Description:	Characters that the number rule substitutes for a number.
Type:	String
Optionality:	Optional
Allowed:	
Default:	
Notes:	This parameter is an element of the <code>NumberRules</code> parameter array.
Example:	<code>replace = "111"</code>
<code>resultNoa</code>	
Syntax:	<code>resultNoa = NOA</code>
Description:	The nature of address (NOA) sent to the network after denormalization.
Type:	Integer
Optionality:	Optional
Allowed:	
Default:	

- Notes:**
- A value for `NOA` is typically specified in denormalization rules.
 - This parameter is an element of the `NumberRules` parameter array.

Example: `resultNoa = 4`

Example number denormalisation

Example 1

```
{ prefix="027", min=9, remove=1, resultNoa=3 }
```

This denormalization rule:

- Matches numbers that:
 - Start with the digits 027
 - Have a minimum of 9 digits
- Removes the first digit.
- Sets `NOA = 3`.

For example, the outgoing message 027nnnnnnnn is changed to 27nnnnnnnn.

Example 2

```
{ prefix="00", min=5, remove=2, prepend="", resultNoa=4 }
```

This denormalization rule:

- Matches numbers that:
 - Start with the digits 00
 - Contain a minimum of 5 digits
- Removes the first two digits.
- Sets `NOA = 4`.

For example, the outgoing message 00nnnnnnnnnnnn is changed to nnnnnnnnnnnn.

Example configuration

An example of the `ECA` section of a `eserv.config` file is listed below. Comments have been removed.

```
ECA = [  
  {  
    sleeInterfaceName = "EDRControlAgent"  
    inputDirectory = "/var/edr/input"  
    processingDirectory = "/var/edr/processing"  
    badFileDirectory = "/var/edr/unprocessable"  
    fileNamePattern = "*.cdr"  
    sleeServiceKey = 300  
    maxIdpsPerSecond = 250  
    statisticsEnabled = true  
    tssf = 10  
    pluginLibrary = "ECASimplePlugin.so"  
    PluginSpecificConfig = {  
      inapServiceKey = 111  
      commentChar = "#"  
      separatorChar = "|"   
      idpParameters = [  
        "calledPartyNumber"  
        "skip"  
        "callingPartyNumber"  
        "locationNumber"  
        "tbcdExtension 1234"  
        "callDurationSeconds"  
      ]  
    }  
  }  
]
```

```

        NumberRules = [
            { prefix = "6449", min = 5, remove = 2, prepend = "0", resultNoa = 2
            }
        ]
        successDirectory = "/var/edr/success"
        failedDirectory = "/var/edr/failed"
    }
}
]

```

Failure

If `edrControlAgent` stops while processing an EDR file, it writes the following information to **ecaProgress.txt**:

- The names of all the files being processed
- The name of the EDR file currently being processed
- The number of IDPs sent for the current file
- If the plug-in requires it, the file names and EDR number for all the 'calls' in progress

When it restarts, `edrControlAgent` uses **ecaProgress.txt** to identify where to start again.

If `edrControlAgent` stops without writing to the **ecaProgress.txt** file, it uses the plug-in to move any files out of the progress directory to the success or failed directory.

Input

`edrControlAgent` takes EDRs from the input directory configured by the *inputDirectory* (on page 12) parameter in **eserv.config**.

Output

`edrControlAgent` moves processed EDRs to the success or failed directory depending on the result of the EDR processing. These directory locations are configurable in **eserv.config**.

`edrControlAgent` also writes error messages to the system messages file, and writes additional output to **/IN/service_packages/ECA/tmp/edrControlAgent.log**

Note: The above are defaults and may vary.

Troubleshooting

Overview

Introduction

This chapter explains the important processes on each of the server components in NCC, and describes a number of example troubleshooting methods that can help aid the troubleshooting process before you raise a support ticket.

In this chapter

This chapter contains the following topics.

Possible Problems..... 23

Possible Problems

Introduction

This topic lists common problems and actions you can take to investigate or solve them. This list enables you to check for alarms based on the overall behavior you are experiencing.

Flooding

edrControlAgent may produce more IDPs than the processing applications can cope with. In this circumstance, edrControlAgent waits until the next second to re-send the IDP.

You can throttle the number of IDPs edrControlAgent sends by specifying one of the following:

- A lower *maxIdpsPerSecond* (on page 13)
- A lower MAX_DIALOGS for the edrControlAgent INTERFACE entry in **SLEE.cfg**.

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	25
Checking the Installation	25
Post-installation Configuration.....	26

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

ECA packages

An installation of Oracle Communications Network Charging and Control EDR Control Agent includes the following packages, on the:

- SMS:
 - ecaSms
- SLC:
 - ecaScp

Checking the Installation

Introduction

Refer to these checklists to ensure that ECA installed correctly.

Checklist - SMS

Follow the steps in this checklist to ensure ECA installed correctly on an SMS machine.

Step	Action
1	Log in to SMS machine as root.
2	Check that the following directory structure exists, with subdirectories: /IN/service_packages/ECA
3	Check that directories contain subdirectories and that all are owned by: smf_oper user (group esg)

Checklist - SLC

Follow the steps in this checklist to ensure that ACS installed correctly on an SLC machine.

Step	Action
1	Log in to SLC machine as root.
2	Check that the following directory structure exists, with subdirectories: /IN/service_packages/ECA
3	Check that the directory contains subdirectories and that all are owned by: smf_oper user (group esg)
4	Check that the processes listed in the process lists are running. For a list of the processes that should be running, see <i>Process list - SLC</i> (on page 26).

Process list - SLC

If the application is running correctly, the following process should be running on each SLC:

- Started during SLEE startup:
 - edrControlAgent

Post-installation Configuration

Configuration process overview

This table describes the steps involved in configuring ECA for the first time.

Stage	Description
1	Uncomment (and if necessary update) the default SLEE configuration for the ECA in the SLEE.cfg file. For more information, see <i>Service Logic Execution Environment Technical Guide</i> .
2	The eserv.config file must be configured for ECA. The installation script adds an ECA section to eserv.config . Any required configuration should be updated. For more information, see eserv.config Configuration (on page 7).