Oracle® Communications Network Charging and Control

CDMA U-CA-IS41 Technical Guide

Release 15.0.0

October 2023

ORACLE®

Copyright

Contents

Chapter 1

Chapter 2

Chapter 3

Chapter 4

Chapter 5

Chapter 6

Appendix A

About This Document

Scope

The scope of this document includes all the information required to install, configure and administer the U-CA-IS41 (CDMA) component.   It does not include a detailed design of the service.

Audience

This guide was written primarily for system administrators and persons installing, configuring and administering the U-CA-IS41 (CDMA).   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 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.

Although it is not a prerequisite 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:

- *Oracle Communications Network Charging and Control Service Logic Execution Environment Technical Guide*
- *Oracle Communications Network Charging and Control Service Management System Technical Guide*
- *Oracle Communications Network Charging and Control Service Management System User's Guide*
- *Oracle Communications Network Charging and Control Advanced Control Services Technical Guide*
- *Oracle Communications Network Charging and Control Advanced Control Services User's Guide*
- *Oracle Communications Network Charging and Control SIGTRAN 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.
Italics	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.
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. 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.

What is CDMA?

Introduction

The U-CA-IS41 (CDMA) enables operators to make use of the IS-41 protocol to provide telephone services to end users on CDMA networks.

CDMA exploits existing products and standard protocols in order to provide a cost-effective solution that potentially can be interfaced to a variety of service platforms.

Diagram

The diagram below illustrates the sub-system components that comprise the U-CA-IS41 (CDMA) service, including external interfaces.
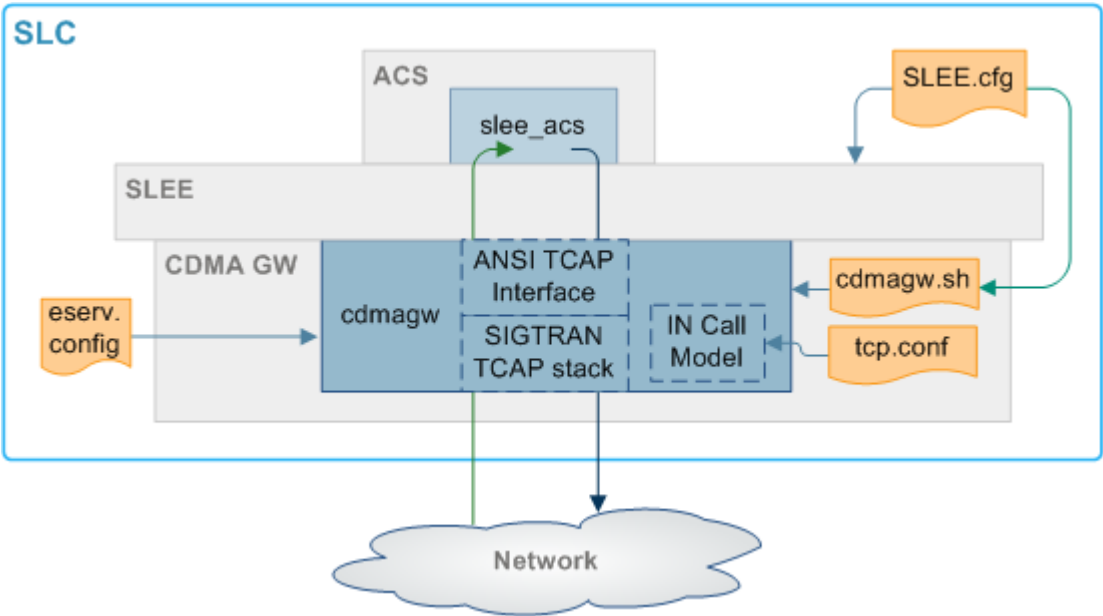
Component descriptions

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

Component	Description	Further Information
cdmagw	cdmagw is the main binary for the CDMA. It is responsible for handling CDMA interactions with MSCs and HLRs, and for triggering appropriate IN interactions to slee_acs. Supports SUA by soft linking to *cdmagw_sua* (on page 26). Supports M3UA by soft linking to *cdmagw_m3ua* (on page 26).	*cdmagw* (on page 25)
slee_acs	Provides the main call processing and service logic.	*ACS Technical Guide*

Possible processing stages

The following diagram illustrates the possible processing stages initiated by CDMA when an IS-41 message is received from the MSC, converted to INAP and forwarded to the SCF.
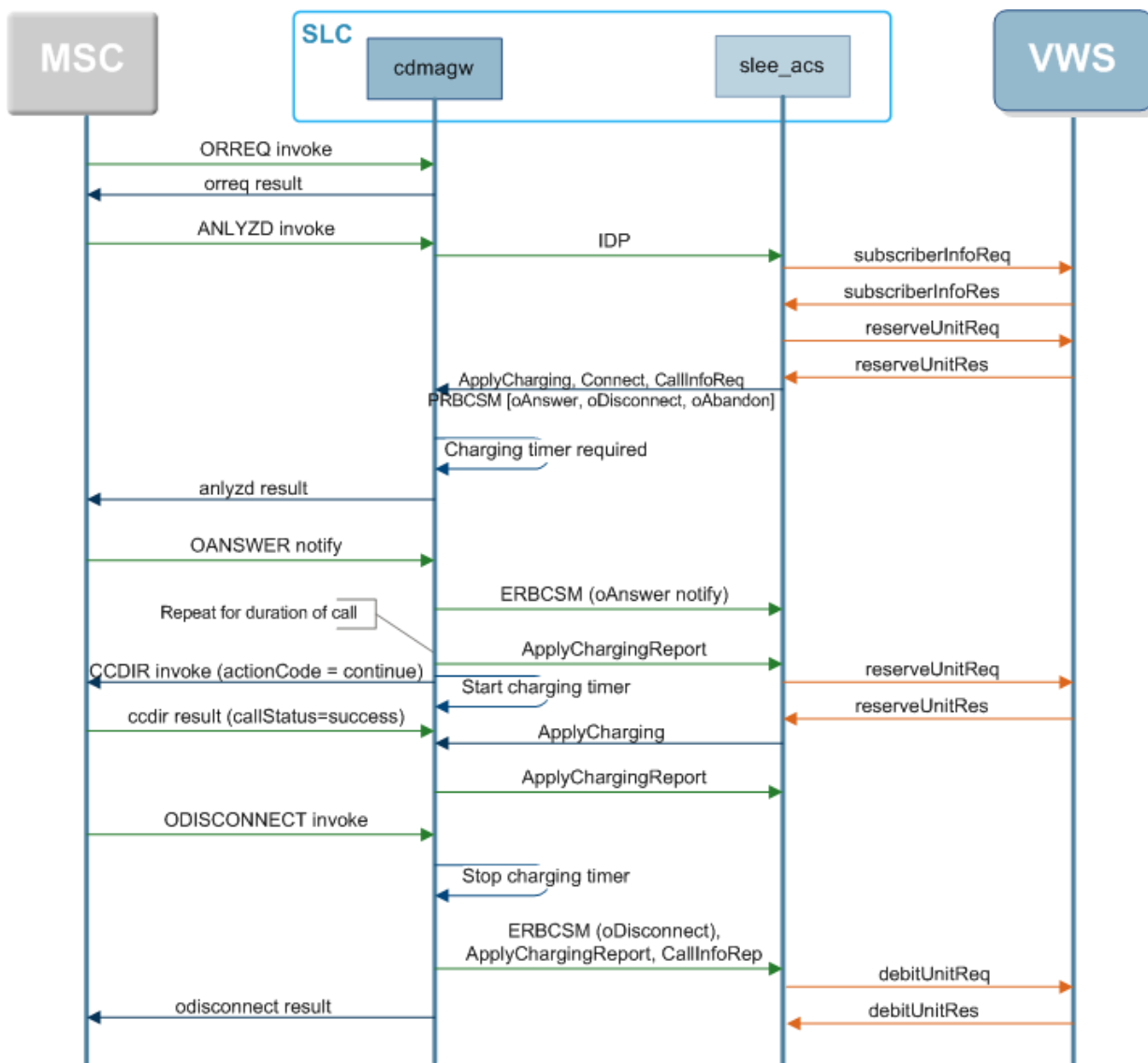
Non-WinRoaming

non-WinRoaming traffic

CDMA supports non-WinRoaming traffic, based on relaying pre-IS-826 calls (that is, IS-771 or earlier) from the HLR by informing the requester to retrigger to a local IS-826 capable MSC.   The local MSC will then perform IS-826 interactions with the SLC, to permit real-time charging to occur.

CDMA supports real-time control of MO voice calls in a non-WIN capable VPLMN (foreign network). CDMA supports this by causing the calls in the non-WIN capable VPLMN (foreign network) to be re-routed back to the WIN-capable HPLMN (home network) for re-origination of an IS-826 controlled call.

Network element requirements

This table describes the standards compliance requirements for the network elements needed to support this solution.

Network element	Requirement specification
Foreign MSC	Must support ANSI-41 as per: • TIA/EIA ANSI-41D, Cellular Radiotelecommunications Intersystem Operations, December 1997 • 6.4.2.30 OriginationRequest (ORREQ) • 6.5.2.90 OriginationTriggers • OriginationTriggers contains a value for All Origination)
Home HLR	Must support the IS-771 ORREQ relay capability as per: • TIA/EIA IS-771, Wireless Intelligent Network, July 1999, Section 4.19.Y Successful Relay of OriginationRequest
Home MSC	Must be IS-826 WIN capable as per: • TIA/EIA IS-826, Wireless Intelligent Network Capabilities for Pre-paid Charging, August 2000.

Non-WinRoaming processing

This process describes how non-WinRoaming calls are handled.

Stage	Description
1	A new IS-771 call arrives at *cdmagw* (on page 25). If OriginationTriggers (ORIGTRIG) Value = All Origination, cdmagw attempts to apply the non-WinRoaming treatment. Otherwise cdmagw continues using the normal message flow (that is, it sends an empty (successful) ORREQ reply).
2	cdmagw determines the GMSC to use, based on the MSID in the call via the MSID prefix to GMSC name mapping in the file specified by *msidGmscFile* (on page 17). The *gmscName* (on page 21) is later used to determine which TSAN range to use. If no msidGmscFile is specified, this lookup is not used. If no GMSC is found: • The call will be assigned the default range specified in *defaultTsanPool* (on page 13). • If no default TSANs are available, AccessDeniedReason is set to *noTsanDeniedReason* (on page 18).
3	cdmagw allocates a TSAN for storing call data. cdmagw stores the IS-771 call data required for charging purposes by the following IS-826 call, indexed on TSAN. This data is globally available to all following calls, so it can be retrieved when the corresponding IS-826 call is retriggered. (Use *nonWinRoamDataTimeout* (on page 18) to timeout call data if no retriggered call arrives.) cdmagw uses the next available TSAN from one of the ranges of TSANs which corresponds to the GMSC. If no free TSAN can be allocated, it drops the call by sending back an orreq with AccessDeniedReason set to *noTsanDeniedReason* (on page 18).
4	cdmagw extracts relevant telephony data and sends back a modified orreq message containing the MSCID, and the TSAN in PSTNTermination->DestinationDigits.
5	cdmagw waits for a IS-826 ANLYZD originated by the IS-771 call. When an IS-826 ANLYZD arrives, cdmagw determines if data associated with a corresponding IS-771 call has been stored.

Stage	Description
6	If the IS-826 has TRIGTYPE=Specific_Called_Party_Digit_String, cdmagw extracts the TSAN from the Mobile Directory Number (MDN) field of the first ANLYZD message. cdmagw uses the TSAN as a key to check the call data for previously encountered IS-771 call data. This data is stored for triggering to slee_acs.
7	A second ANLYZD message may be received containing data (specifically the Original Calling Party number) which was not received in the first. If *waitForCallingParty* (on page 22) = true, cdmagw waits for the Original Calling Party number to be received in CallingPartyNumberDigits1 before the call progress. The second ANLYZD message doesn't include a TSAN, so cdmagw uses the BillingID to map to the call data instead. If a second ANLYZD is received and the call has already been authorized, an empty anlyzd response is sent back.
8	cdmagw sends Analyzed Response (anlyzd) message with: • The original dialed number in the Digits (dialed) parameter • The full originating TriggerAddressList parameter

Supported ANLYZD parameters

The following parameters are supported in ANLYZD messages:
- BillingId
- Digits (Dialed)
- TriggerType
- CallingPartyNumberDigits1
- DestinationDigits
- MSCID
- MobileDirectoryNumber
- MSID (MIN or IMSI)

Alarms, Statistics, Reports and EDRs

Alarms

CDMA uses the SMS integrated alarms collection, viewing and forwarding system.   The alarms generated by all components of CDMA are consolidated on the SMS and stored in a centralised alarm database.

The operator can:

- View the alarms through the alarm viewer built into the SMS screens
- Forward all alarms to an integrated external fault management system using SNMP v1 or v3.

Alarms can be automatically deleted from the SMF alarm database after a configurable period.

For more information about the:

- Specific alarms generated by CDMA, see *CDMA Alarms Guide*.
- SMS alarms subsystem, see *SMS User's Guide*.

Statistics

The following table shows the statistics that will be generated by the cdmagw service if statistics are enabled.   For more information about how cdmagw generates statistics, see *Configuration* (on page 7).

Statistic ID	Description
NUM_ORREQ_INVOKE	Total origination requests.
NUM_ORREQ_RESPONSE	Total origination responses.
NUM_ANLYZD_INVOKE	Total analyzed information requests.
NUM_ANLYZD_RESPONSE	Total analyzed information responses.
NUM_OANSWER_INVOKE	Total origination answer requests.
NUM_TANSWER_INVOKE	Total termination answer requests.
NUM_ODISCONNECT_INVOKE	Total origination disconnect requests.
NUM_ODISCONNECT_RESPONSE	Total origination disconnect responses.
NUM_TDISCONNECT_INVOKE	Total termination disconnect requests.
NUM_TDISCONNECT_RESPONSE	Total termination disconnect responses.
NUM_CCDIR_INVOKE	Total call control directive requests.
NUM_CCDIR_RESPONSE	Total call control responses.
NUM_SUCCESSFUL_CALLS	Total successfully completed call attempts.
NUM_FAILED_CALLS	Total failed call attempts.
NUM_CONNRES_INVOKE	Total connect to resource requests.
NUM_CONNRES_SUCCESS	Total connect to resource successful responses.
NUM_CONNRES_FAILED	Total connect to resource failed responses.
NUM_BULKDISCONN_INVOKE	Total bulk disconnect requests.
NUM_BULKDISCONN_RESPONSE	Total bulk disconnect responses.

Reports

CDMA does not install any specific SMS reports.   However, you can report on any statistics which are recorded using the SMS Application report on the SMS Report Functions screen.   For more information about running SMS reports, see *SMS User's Guide*.

EDRs

CDMA does not write any EDRs itself.   However messages sent from cdmagw to slee_acs will result in an EDR being written by slee_acs.   For more information about what EDRs are written by slee_acs, see *Event Detail Record Reference Guide*.

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 CDMA component is configured.

Configuration components

CDMA is configured by the following components:

Component	Locations	Description	Further Information
cdmagw.sh	All SLC machines	This shell script sets the command line parameters for the main *cdmagw* (on page 25) binary.	*cdmagw.sh Configuration* (on page 23)
eserv.config	All SLC machines	The CDMA is configured in the `cdmagw` section of the **eserv.config** file.	**eserv.config** *Configuration* (on page 9)
msidGmscFile	All SLCs	This file holds the MSID to GMSC mapping used to locate an available TSAN when supporting non-WinRoaming.	*msidGmscFile* (on page 8)
tdp.conf	All SLC machines	The **tdp.conf** file configures the IN Call Model in cdmagw.	*IN Call Model configuration* (on page 8)
SLEE.cfg	All SLC machines	The SLEE configuration must be altered to include the CDMA.	*SLEE Technical Guide*

Component	Locations	Description	Further Information
		The system is configured so that the CDMA and associated interfaces all start together. This is performed by the SLEE and is defined using **SLEE.cfg**.	
SMF database	SMS	Statistics and EFM alarms configuration for CDMA. Configured automatically when cdmaSms is installed.	*Installation Guide*

IN Call Model configuration

Configuration for the IN Call Model element of the U-CA-IS41 (CDMA) is contained in the **/IN/service_packages/SLEE/etc/tdp.conf** file.   This text shows an example of the file.

```
# mandatory params:
CAMEL
ADDITIONALNUMS
IMSI QUAL=129

# optional params: (comment out if not required)
ACH WARN PERIOD=10
ACH TONE=20,0
ACH ANNOUNCE=156

# the trigger/service key definition
3 1 3 request all all
```

For more information about how to configure the **tdp.conf** file, see Configuring IN Call Model Triggers.

msidGmscFile

The **msidGmscFile** file is only used for non-WIN roaming redirection to pick a GMSC TSAN pool to use.

It contains multiple entries (one entry per line) in the form:

```
MSID_Prefix,GMSC_Name
```

Notes:

- Each MSID Prefix must be less than *maxMsidForGmscSelect* (on page 17) characters long.

- The GMSC name must exist in the **eserv.config** under *tsanPools* (on page 20).

- On configuration reload, the file specified in *msidGmscFile* (on page 17) is read.   This can be triggered by a SIGHUP.

- Lines that begin with hashes (#), are ignored by *cdmagw* (on page 25).

Example: This text shows an example of a line from msidGmscFile.

```
189883,MSC1
```

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" ]
}
```

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.

Example cdmagw Configuration Section

Here is an example of the `cdmagw` section in the **eserv.config** file, showing the CDMA configuration.

```
cdmagw = {
    ImsiIdpQualifier=129
    CallStatusTimeout=30
    ResetSrfTimeout=30
    DisconnectTimeout=30
    BusyStatusTimeout=10
    CcdirTimeout=30
    CallFailedTimeout=1
    DisconnectWarningType='T'
    DisconnectWarningID=24
    LowCreditWarningType='T'
    LowCreditWarningID=24
    TestInterfaceEnabled=false
    TestInterfacePort=14875
    SoakTestInterfaceEnabled=false
    SoakTestInterfacePort=14876
    EnableDisConnRes=true
    ANLYZDActionCodeRequired=false
    ThrottlingCallRate=0
    EnableStats=false
    AnnouncementsUseCCDIR=true
    MaxStatusCheckFailures=1
    LocationIdIncludesMSCID=false
    SoakTestCalledNumber="1234567890"
    SoakTestCallingNumber="1234567890"
    CompareDestinationDigits=false
    AllowInitialTermination=false
    Dynamic_DMHServiceID=false
    DMHServiceID_ANLYZD_OnRelease=false
    DMHServiceID_ANLYZD=""
    DMHServiceID_ORREQ=""
    DMHServiceID_ODISCONNECT=""
    DMHServiceID_TDISCONNECT=""
    DMHServiceID_DynamicPrefix="1.1"
    waitForCallingParty = false
    abortOnCallFail = false
    forceCorrelationID = false
    dummyMDNForCallLookup="0000"
    msidGmscFile=""
    maxMsidForGmscSelect=6
    msidRequiredDeniedReason= 0
    cfnaOnRedirectionIndicators = [3,17]
    nonWinRoamDataTimeout = 30
    noTsanDeniedReason = 3
    keepTsanLocal = true
    dummyMsid = "989123"
    defaultTsanPool = "MSC1"
    tsanAnlyzdField = "Digits"
    tsanPools = [
        {
            gmscName="MSC1"
            tsanNature = 4
            ranges = [
                {
                    rangeStart = "8049096367"
                    rangeEnd = "8049096417"
                }
```

```
        {
            rangeStart = "9876540850"
            rangeEnd = "9876540890"
        }
    ]
}


{
    gmscName="MSC2"
    ranges = [
        {
            rangeStart = "0987650990"
            rangeEnd = "0987650150"
        }
        {
            rangeStart = "5402946949"
            rangeEnd = "5402946999"
        }
    ]
}
    ]


    }
}
```

Parameters

The following parameters are supported.

`abortOnCallFail`

Syntax:	`abortOnCallFail = true|false`
Description:	Indicates if an abort message should be sent to slee_acs on call failure. That is, CCDIR failure response.
Type:	Boolean
Optionality:	Optional (default used if not set).
Allowed:	true or false
Default:	`false`
Notes:	
Example:	`abortOnCallFail = false`

`AllowInitialTermination`

Syntax:	`AllowInitialTermination = false|true`	
Description:	Flag to control whether to allow InitialTermination triggers to create a call.	
Type:	Boolean	
Optionality:	Optional (default used if not set).	
Allowed:	false	Wait for the CalledRoutingAddressAvailable trigger in the second ANLYZD message.
	true	Create a call instance when a InitialTermination trigger arrives.
Default:	false	
Notes:	This trigger does not contain the Called Number (DestinationDigits), so the IDP CalledPartyNumber field contains the Dialed Digits.	

For more information about message flow, see *Possible processing stages* (on page 3).

Example:

ANLYZDActionCodeRequired

Flag to enable sending an ActionCode in an ANLYZD result (success or failure).

Default:	true
Allowed:	true or false

AnnouncementsUseCCDIR

Flag to control whether switch-based announcements use CCDIR or ANLYZD/ODISCONNECT responses.

Default:	false
Allowed:	true or false

BusyStatusTimeout

A periodic timer that sets the number of seconds to wait before checking whether the called party was busy before a call has been answered.

Default:	30 seconds
Allowed:	Integer

CallFailedTimeout

The amount of time between verifying a call failure and releasing a call.

Default:	1 second
Allowed:	Integer, minimum value 1 second.
Note:	Cannot be disabled.

CallStatusTimeout

A periodic timer that checks the status of the call to make sure no calls have been dropped by the MSC without notifying the Cdma Gateway.

Default:	30 seconds
Allowed:	Integer

CcdirTimeout

A periodic timer that sets the number of seconds to wait before assuming that the CCDIR invoke operation failed.

Default:	30 seconds
Allowed:	Integer

cfnaOnRedirectionIndicators

Syntax:	cfnaOnRedirectionIndicators = [*array*]
Description:	The value specified indicates the *DMH_RedirectionIndicator* that will be used when handling Call Forward No Answer (CFNA) scenarios.
Type:	Array
Optionality:	Optional (default used if not set).
Allowed:	

Default:	[3]
Notes:	Specifying `0` means the indicator will not be checked. It overrides all values.
Example:	`cfnaOnRedirectionIndicators = [3,17]`

`CompareDestinationDigits`

Syntax:	`CompareDestinationDigits = false|true`	
Description:	Which digits to use to determine whether the Called Party number has changed on receipt of a Connect operation from slee_acs.	
Type:	Boolean	
Optionality:	Optional (default used if not set).	
Allowed:	true	Compare destination digits.
	false	Compare dialed digits.
Default:	false	
Notes:		
Example:		

`ConnresTimeout`

A periodic timer that sets the number of seconds to wait before assuming that the CONNRES was successful.

Default:	10 seconds
Allowed:	Integer
Note:	The timer can be disabled if the MSC is configured to send TC_CANCEL (0 = disabled)

`defaultTsanPool`

Syntax:	`defaultTsanPool = "name"`
Description:	The name of the default GMSC TSAN pool to use if a MSID isn't available or it's not found in *msidGmscFile* (on page 8).
Type:	String
Optionality:	Optional (default used if not set).
	Mandatory for IS-826 retriggering.
Allowed:	This must exist in the tsanPools section.
Default:	none (No default pool)
Notes:	For more information about TSANs, see *Non-WinRoaming processing* (on page 4).
Example:	`defaultTsanPool = "MSC1"`

`DisconnectTimeout`

A periodic timer that sets the number of seconds to wait before assuming that the forced release (using a CCDIR) has failed.

Default:	30 seconds
Allowed:	Integer

`DisconnectWarningID`

The ID of the tone to be played when a call is disconnected using CCDIR.

Default: PPCDisconnectTone (24)

Allowed: Integer

Note: This is ignored if the type is set to 'N'.

`DisconnectWarningType`

Playas a tone, or announcement, or nothing when about to disconnect.

Default: T

Allowed: T = Tone, A = Announcement, N = None

`DMHServiceID_ANLYZD`

The DMH_ServiceID to send to the MSC as part of an ANLYZD result.   If this is set to an empty value no DMH_ServiceID is sent in the result.:

Default: ""

Allowed: String

Note: The format is "marketID.marketSegmentId.ServiceIdValue"

`DMHServiceID_ANLYZD_OnRelease`

Flag to enable sending a DMH_ServiceID in an ANLYZD response that releases a call.

Default: true

Allowed: true or false

`DMHServiceID_DynamicPrefix`

The prefix to add to the dynamic DMH_ServiceID.   Only used when dynamic dmh_service ids are enabled, this should be set to an empty value.   If this is set to an empty value no DMH_ServiceID is sent in the ANLYZD result.

Default: 1.1

Allowed: String

Note: The format is "marketID.marketSegmentId"

`DMHServiceID_ODISCONNECT`

The DMH_ServiceID to send to the MSC as part of an ODISCONNECT result.   If this is set to an empty value no DMH_ServiceID is sent in the result.

Default: ""

Allowed: String

Note: The format is "marketID.marketSegmentId.ServiceIdValue"

`DMHServiceID_ORREQ`

The DMH_ServiceID to send to the MSC as part of an ORREQ result.   If this is set to an empty value no DMH_ServiceID is sent in the result.

Default: ""

Allowed: String

Note: The format is "marketID.marketSegmentId.ServiceIdValue"

DMHServiceID_TDISCONNECT

The DMH_ServiceID to send to the MSC as part of an TDISCONNECT result.   If this is set to an empty value no DMH_ServiceID is sent in the result.

Default:	""
Allowed:	String
Note:	The format is "marketID.marketSegmentId.ServiceIdValue"

dummyMDNForCallLookup

Syntax:	dummyMDNForCallLookup = "number"
Description:	The dummy MDN number to use as part of look up key for call lookup.
Type:	String
Optionality:	Optional (default used if not set).
Allowed:	
Default:	none (no default pool)
Notes:	This is used because the key for call lookup for a second ANLYZD message, consists of both billing ID and MDN.
	Setting this effectively means that MDN is not used.
	For more information about how this is used, see *Non-WinRoaming processing* (on page 4).
Example:	dummyMDNForCallLookup = "0000"

dummyMsid

Syntax:	dummyMsid = "*number*"
Description:	This is used to set the MSID internally when no MIN is received in the ANLYZD message.
Type:	String
Optionality:	Optional (default used if not set).
Allowed:	
Default:	"0"
Notes:	However, if a MIN is received in the ANLYZD, that will be used to set the MSID internally instead.
Example:	dummyMsid = "989123"

Dynamic_DMHServiceID

Flag to enable dynamic DMH_ServiceIDs, set by the SCP using FCI.   Enabling this overrides the static ANLYZD DMH_ServiceID.

Default:	false
Allowed:	true or false

EnableDisConnRes

Flag to enable or disable whether the CdmaGateway should send a DISCONNRES before sending an ANLYZD or ODISCONNECT result to an MSC following a CONNRES announcement.

Default:	true
Allowed:	true or false

EnableStats

Flag to enable the reporting of statistics.

Default:	false
Allowed:	true or false

forceCorrelationID

Syntax:	forceCorrelationID = true|false
Description:	Indicates whether to force the correlation ID to be set for the following operations:
	• Connnect to Resource
	• ResetTimer
	• Disconnect Resource
Type:	Boolean
Optionality:	Optional (default used if not set).
Allowed:	true or false
Default:	false
Notes:	
Example:	forceCorrelationID = false

ImsiIdpQualifier

The identifier that is used when sending the IMSI in an IDP.

Default:	129 (Binary: 10000001)
Allowed:	Integer
Note:	This value MUST be the same as specified in the 'tdp.conf' file.

keepTsanLocal

Syntax:	keepTsanLocal = true|false
Description:	If this parameter is set to true, the gateway checks if the Digits or DestinationDigits in a received ANLYZD message match a non-WIN roaming TSAN record.
	If yes, it will replace these fields with stored values from a previous origination request, preventing the TSAN from being sent to the service.
Type:	Boolean
Optionality:	Optional (default used if not set).
Allowed:	true or false
Default:	true
Notes:	
Example:	keepTsanLocal = true

LocationIdIncludesMSCID

Flag to enable including the MSCID in the Location Area ID (MSCID is prepended to the ServingCellID).

Default:	false
Allowed:	true or false

LowCreditWarningID

The ID of the tone to be played when a call has reached its low credit theshold.

Default:	PPCDisconnectTone (24)
Allowed:	Integer
Note:	This is ignored if the type is set to 'N'.

LowCreditWarningType

Plays a tone, announcement or nothing when credit balance is low.

Default:	T
Allowed:	T = Tone, A = Announcement, N = None

maxMsidForGmscSelect

Syntax:	maxMsidForGmscSelect = *len*
Description:	The maximum length of an acceptable MSID prefix within the file specified by *msidGmscFile* (on page 17) for GMSC lookup.
Type:	Integer
Optionality:	Optional (default used if not set).
Allowed:	
Default:	6
Notes:	If an entry in the msidGmscFile is more than this limit, a WARNING alarm is logged and the row is ignored.
Example:	maxMsidForGmscSelect = 6

MaxStatusCheckFailures

How may CCDIR failures to allow before assuming that the call has failed.

Default:	1
Allowed:	Integer
Note:	Setting this to 1 means that the first failure will cause the call to fail.

msidGmscFile

Syntax:	msidGmscFile = "*path/file*"
Description:	The name and location of file which defines the mapping of MSIDs with to GMSCs.
Type:	String
Optionality:	Optional (default used if not set).
Allowed:	
Default:	none (that is, no mapping file)
Notes:	For more information about the file, see *msidGmscFile* (on page 8).
	On configuration reload, the file specified in *msidGmscFile* (on page 17) is read. This can be triggered by a SIGHUP.
	For more information about how this mapping is used, see *Non-WinRoaming processing* (on page 4).
Example:	msidGmscFile = "/IN/service_packages/IS41/etc/msidtogmsc.mapping"

msidRequiredDeniedReason

Syntax:	`msidRequiredDeniedReason = reason`
Description:	The AccessDeniedReason returned if no MSID is available in ORREQ for roaming calls.
Type:	Integer
Optionality:	Optional (default used if not set).
Allowed:	
Default:	0 (MSID not required, TSAN will be allocated from the default TSAN pool)
Notes:	For more information about how roaming calls are handled, see *Non-WinRoaming processing* (on page 4).
Example:	`msidRequiredDeniedReason = 0`

nonWinRoamDataTimeout

Syntax:	`nonWinRoamDataTimeout = seconds`
Description:	The number of seconds to retain call data for non-WinRoaming calls.
Type:	Integer
Optionality:	Optional (default used if not set).
Allowed:	
Default:	30
Notes:	For more information about how roaming calls are handled, see *Non-WinRoaming processing* (on page 4).
Example:	`nonWinRoamDataTimeout = 30`

noTsanDeniedReason

Syntax:	`noTsanDeniedReason = reason`	
Description:	The AccessDeniedReason to send back if all TSANs are in use.	
Type:	Integer	
Optionality:	Optional (default used if not set).	
Allowed:	0	Not used.
	1	Unassigned directory number (the MS is not served by the accessed system).
	2	Inactive (the MS is not active in the accessed system and the HLR pointer to the MSs VLR should be maintained).
	3	Busy (the MS is busy in the accessed system and cannot accept additional calls).
	4	Termination Denied (terminations to this MS are not allowed).
	5	No Page Response (the MS was paged by the accessed system but did not respond).
	6	Unavailable (the MS is currently not available and the HLR pointer to the MSs VLR should be maintained and the MS shall remain in the same state).
Default:	3	
Notes:	For more information about TSANs, see *Non-WinRoaming processing* (on page 4).	

Example: `noTsanDeniedReason = 3`

`ResetSrfTimeout`

A periodic timer that resets the 'SSFT' MSC timer.

Default:	30 seconds
Allowed:	Integer
Note:	This must be sent during an IP interaction, initiated by a CONNRES.

`SoakTestCalledNumber`

The default value for called numbers when using the   SoakTestInterface.

Used for DEST and CALLED numbers.

Default:	"1234567890"
Allowed:	String

`SoakTestCallingNumber`

The default value for calling numbers when using the   SoakTestInterface.

Used for MDN and CALLING numbers.

Default:	"1234567890"
Allowed:	String

`SoakTestInterfaceEnabled`

Flag to enable or disable the soak test interface.   In a production system this can be disabled by default but enabled so that a soak test can be run on the system.

Default:	false
Allowed:	true or false
Note:	Enabling this interface overrides the normal test interface.

`SoakTestInterfacePort`

The tcp network port that the soak test interface will listen on (if it is enabled).

Default:	14876
Allowed:	Integer

`TestInterfaceEnabled`

Flag to enable or disable the test interface.   In a production system this can be disabled by default but enabled so that a test can be run on the system.

Default:	false
Allowed:	true or false
Note:	Enabling the soak test interface overrides this interface.

`TestInterfacePort`

The tcp network port that the test interface will listen on (if it is enabled).

Default:	14875
Allowed:	Integer

ThrottlingCallRate

Limits (throttles) how much traffic the cdmagw will process per second.

Description:	Limits (throttles) the traffic the cdmagw processes per second.
Type:	Integer
Default:	0 (disabled)
Allowed:	Any Integer
Notes:	If suaCdmaGateway or m3uaCdmaGateway fail to create a new SLEE dialog due to overload then new calls are rejected until the end of the monitor period. That is, until the end of the current second.

tsanAnlyzdField

Syntax:	`tsanAnlyzdField = "field"`
Description:	The number field containing the TSAN for nonWinRoaming.
Type:	String
Optionality:	Optional (default used if not set).
Allowed:	• DestinationDigits • Digits • MDN
Default:	"Digits"
Notes:	
Example:	`tsanAnlyzdField = "Digits"`

tsanPools

Syntax:	```tsanPools = [
	    {
	        gmscName="name"
	        ranges = [
	            {
	                rangeStart = "tsan"
	                rangeEnd = "tsan"
	            }
	            ...
	        ]
	    }
	    ...
	]```

Description:	The list of permissible TSANs per GMSC.
Type:	Array
Optionality:	Optional (not used if not set) Mandatory for IS-826 retriggering
Allowed:	
Default:	
Notes:	For more information about IS-826 and TSANs, see *Non-WinRoaming processing* (on page 4).

Example:	```tsanPools = [
	    {
	        gmscName="MSC1"
	        ranges = [
	            {
	                rangeStart = "8049096367"
	                rangeEnd = "8049096417"```

```
            }
            {
                rangeStart = "9876540850"
                rangeEnd = "9876540890"
            }
        ]
    }

    {
        gmscName="MSC2"
        ranges = [
            {
                rangeStart = "0987650990"
                rangeEnd = "0987650150"
            }
            {
                rangeStart = "5402946949"
                rangeEnd = "5402946999"
            }
        ]
    }
]
```

`gmscName`

Syntax:	`gmscName = "name"`
Description:	The name of the Gateway MSC for the range of TSANs defined by the corresponding range parameter.
Type:	String
Optionality:	Optional (ranges not used if not set)
	Mandatory for IS-826 retriggering
Allowed:	A string up to 50 characters long
Default:	
Notes:	For more information about TSANs, see *Non-WinRoaming processing* (on page 4).
Example:	`gmscName = "MSC1"`

`ranges`

Syntax:	`ranges = [`
	`    {}`
	`    ...`
	`]`

Description:	The TSAN pools which can be used with this Gateway MSC.
Type:	Array
Optionality:	Optional (not used if not set)
	Mandatory for IS-826 retriggering
Allowed:	
Default:	none
Notes:	
Example:	For an example of this parameter used in context, see *tsanPools* (on page 20).

rangeEnd

Syntax:	rangeEnd = "*tsan*"
Description:	The last TSAN in a range in a TSAN pool.
Type:	String
Optionality:	Mandatory if *ranges* (on page 21) is used.
Allowed:	
Default:	
Notes:	For more information about TSANs, see *Non-WinRoaming processing* (on page 4).
Example:	For an example of this parameter used in context, see *tsanPools* (on page 20).

rangeStart

Syntax:	rangeStart = "*tsan*"
Description:	The first TSAN in a range in a TSAN pool.
Type:	String
Optionality:	Mandatory if *ranges* (on page 21) is used.
Allowed:	
Default:	
Notes:	For more information about TSANs, see *Non-WinRoaming processing* (on page 4).
Example:	For an example of this parameter used in context, see *tsanPools* (on page 20).

tsanNature

Syntax:	tsanNature = *noa*	
Description:	This is the NoA (nature of address) of the TSAN which is sent in the orreq (Origination Request Result) in both the DialedDigits and the DestinationDigits fields (See Notes below).	
Type:	Integer	
Optionality:	Optional (default used if not set)	
Allowed:	3	Sets national for the INAP representation of NOA for the TSAN.
	4	Sets international for the INAP representation of the TSAN.
Default:	3 (national)	
Notes:	• These values will be converted into an IS41 Nature of Address in the OriginationRequest return result. i.e. 0 for national and 1 for international.	
	• The DestinationDigits field is derived from TerminationList -> PSTNTermination -> DestinationDigits.	
Example:	tsanNature = 4	

waitForCallingParty

Syntax:	waitForCallingParty = false|true
Description:	How long to wait for a calling party value in a message before sending an IDP.
Type:	Boolean
Optionality:	Optional (default used if not set).

Allowed:	true	If there is no calling party number in the first ANLYZD message, wait until the second ANLYZD message is received.
	false	If there is not a calling party number in the first or second ANLYZD messages use the MDN.
Default:	false	
Notes:		For more information about message flows and ANLYZD messages, see *Non-WinRoaming processing* (on page 4).
Example:		

SIGTRAN configuration

cdmagw also supports the configuration for sua_If and m3ua_If from the SIGTRAN TCAP Interface in the cdmagw section.   The SIGTRAN parameters are used as if they were inside the a sigtran section.

For more information about the available parameters, see *SIGTRAN TCAP IF Technical Guide*.

cdmagw.sh Configuration

About cdmagw.sh configuration

The CDMA startup shell script **/IN/service_packages/IS41/bin/cdmagw.sh** contains additional configuration for the ANSI TCAP Interface.

This setting states where the **tdp.conf** file is located (this file is installed during setup):

```
TDP_DEFINITONS=/IN/service_packages/SLEE/etc/tdp.conf
export TDP_DEFINITIONS
# the following settings should be configured to reflect your system
#   setup:
# -ssns  = subsystem numbers - default = 19
# -autoac = auto app context set - default = yes
# -defoutac = Specify object identifier to use for TCAP Interface Application
context
# -stps     = comma seperated list of STPs
# -monitorperiod = the period over which to monitor call attempts for throttling -
default = 1000
# -retssn   = return SSN address
# -retpc    = return PC address
# -retgt    = return GT address
# -retri    = return RI
# -asidbase = application server id base
```

Note: The start up option `-retpc` can also be expressed as either an existing single integer, or as an ANSI network-cluster-member point code representation.

For example:

```
-retpc 2193
-retpc 10-20-42
```

cdmagw.sh example

This text shows an example **cdmagw.sh**:

```
TDP_DEFINITONS=/IN/service_packages/SLEE/etc/tdp.conf
export TDP_DEFINITIONS
```

```
exec /IN/service_packages/IS41/bin/cdmagw \
        -ssns 11,146 \
        -stps 2596 \
>> /IN/service_packages/IS41/tmp/cdmagw.log 2>&1
```

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.

cdmagw

Purpose

cdmagw is the main binary for the CDMA.   It is responsible for handling CDMA interactions with MSCs and HLRs, and for triggering appropriate IN interactions to slee_acs.   Uses the IN call model.

Note: cdmagw is usually a link to one of *cdmagw_sua* (on page 26) or *cdmagw_m3ua* (on page 26). The binary linked to it defines which version of the SIGTRAN stack is being used.

Location

This binary is located on SLCs.

Startup

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

```
INTERFACE=cdmagw cdmagw.sh /IN/service_packages/IS41/bin EVENT
```

Notes:

- **cdmagw.sh** is a shell script which starts cdmagw.   For more information about the configuration which must be set in this file, see *cdmagw.sh Configuration* (on page 23).

- The above are defaults and may vary.

cdmagw_sua

Purpose

cdmagw_sua provides the cdmagw binary compiled against the sua_if version of the SIGTRAN stack. It provides CDMA voice over SCCP over SUA.

For more information about sua_if, see *SIGTRAN TCAP IF Technical Guide*.

Location

This binary is located on SLCs.

Startup

If this binary is linked to from cdmagw, it will be started in the way described in *Startup* (on page 25).

Configuration

cdmagw_sua is configured using **eserv.config** and **cdmagw.sh**.   For more information about the available parameters, see *Configuration* (on page 7).

cdmagw_m3ua

Purpose

cdmagw_m3ua provides the cdmagw binary compiled against the m3ua_if version of the SIGTRAN stack.   It provides CDMA voice over SCCP over M3UA.

For more information about m3ua_if, see *SIGTRAN TCAP IF Technical Guide*.

Location

This binary is located on SLCs.

Startup

If this binary is linked to from cdmagw, it will be started in the way described in *Startup* (on page 25).

Configuration

cdmagw_m3ua is configured using **eserv.config** and **cdmagw.sh**.   For more information about the available parameters, see *Configuration* (on page 7).

Administrative Tasks

Overview

Introduction

This chapter provides the procedures for administering the U-CA-IS41 (CDMA) application.

In this chapter

This chapter contains the following topics.

Starting and Stopping the U-CA-IS41 (CDMA)

Introduction

This topic explains how to start or stop the U-CA-IS41 (CDMA) application.

Starting the U-CA-IS41 (CDMA) service

To start the automated shell script, which in turn starts the U-CA-IS41 (CDMA) service, as the user `acs_oper`, enter:

`/IN/service_packages/SLEE/bin/slee.sh`

Result: This shell script starts the slee_acs and the associated interfaces cdmagw, timer IF and cdrIF.

The stdout and stderr from **slee.sh** will appear on the screen, so if this screen is closed the output will no longer be viewable. If this information is required then redirect output to a file, for example **slee.sh** to **sleeout.log**.

Startup output

When the SLEE service starts various information is presented on stdout and the syslog.

Stopping the U-CA-IS41 (CDMA) service

To stop the automated shell script, which in turn stops the U-CA-IS41 (CDMA) service, as the user `acs_oper`, enter:

`/IN/service_packages/SLEE/bin/stop.sh`

Note: It also recommended to run a **./clean** following the stop.

If the SLEE_FILE variable is being used it must be visible to the stop program. If it is not visible, the program will not be able to clear the shared memory and will exit with error 3005.

Note: If the service has stopped for any abnormal reasons a manual cleanup should be performed, that is, `ps -fu acs_oper` to find the remaining processes, then `kill` *pid* each one. You should check the shared memory using `ipcs | grep acs`, then remove acs_oper owned ones using `ipcrm` and `clean`.

Backing up the U-CA-IS41 (CDMA) Service

Introduction

The filesystem areas that should be backed up for the U-CA-IS41 (CDMA) service are **/IN/service_packages/IS41/**.

The standard UNIX restore utilities can then be used.

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.

Common Troubleshooting Procedures

Introduction

Refer to *System Administrator's Guide* for troubleshooting procedures common to all NCC components.

Debug output

Debugging output is available.   Do not enable debugging under production call loads.   Enable debugging only when advised to do so by Oracle support.

To switch the debug level, send *cdmagw* a SIGUSR1.

To turn on SCCP logging, send cdmagw a SIGUSR2.

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

CDMA (U-CA-IS41) packages

An installation of CDMA (U-CA-IS41) includes the following packages:

* cdmaSms on the SMS
* cdmagw on the SLC

Configuring IN Call Model Triggers

Overview

This introduces the generic configuration requirements of the NCC IN Call Model.

The NCC IN Call Model is not a separate product, rather it is a set of libraries that is bound into a final usable interface (such as the UCA-ISUP).

Environment variables

This table describes the UNIX shell environment variables to be configured.

Environment Variable Name	Description	Example Value
TDP_DEFINITIONS	Defines the full path name of the Trigger Detection Point definition file.	**/IN/service_packages/SLEE/etc/tdp.conf**

Trigger detection point (TDP) definition file

The **tdp.conf** file has two sections:

1   A number of configuration parameters.
2   The trigger tables used to determine when to trigger a call to the SCF.

Example: This text shows an example **tdp.conf** file:

```
# A comment
KEEP SD
ETC RULES=6 3
3 1 3 request all 123 6
4 2 4 notify all 222 keep
3 1 3 request 2:122 3:222 5 keep
```

Note: All lines starting with # are treated as comments. If no TDP definition file is defined, a default action is taken where:

- ALL calls are triggered to the SCF with a service key of 1 (one) and a trigger point of 3 (analyzedInformation.)

- None of the global configuration parameters are considered set.

Global configuration parameters

The following configuration parameters may be set once on individual lines in the TDP definition file.

Global Parameter	Description
KEEP SD	If defined ALL all stop digits (defined by the BCD digit 'F') on the end of called party numbers are kept in the called party number. By default the stop digit is stripped from ALL triggered numbers.
CAMEL	This parameter is intended for CAMEL testing purposes only and should not be defined under normal usage.

Global Parameter	Description
	If defined, the called party number is also copied into the intialDP's calledPartyBCDNumber CAMEL parameter. The NOA of the called party number becomes the BCD number type.
ADDITIONALNUMS	If defined, the IN Call Model will request all additional numbers available from the underlying protocol and insert them into the InitialDP message sent to the SLC.
	All these additional numbers are placed into a G8 extension in the InitialDP except any additional calling party number that is placed in the additionalCallingPartyNumber field.
ETC RULES= c or ETC RULES= c s	If defined then additional EstablishTemporaryConnection (ETC) rules are used.
	If the integer c is defined, the correlationID in all ETC messages from the SCF are appended on to the end of the assistingSSPIPRoutingAddress that is used, the digits are padded to a width of c digits.
	If s is also defined, then the scfID of the ETC is also appended on afterwards in the same way.
	For example:
	With "ETC RULES=6 4" and an ETC message with:
	assistingSSPIPRoutingAddress =1111, correlationID =55, scfID =0x42
	Then the actual assistingSSPIPRoutingAddress used will be "11110000550042".
USER LIB = *library*	If defined the call model will use the user written shared object *library* specified by the full pathname library when dealing with ApplyCharging operations.
AC=*a,b,c....*	Sets the TCAP application context used by the call model to the comma separated list of OIDs supplied.
ORIG_PC= pc	If defined, all InitialDPs will be sent with an SCCP calling party (origination) address that includes a Point Code defined by the integer pc.
	If not defined, and ORIG_SSN and ORIG_GT are not defined, all InitialDPs will be sent without an SCCP calling party address.
	Note: This value may be defined in hex using a prefix of 0x.
ORIG_SSN= ssn	If defined, all initialDPs will be sent with an SCCP calling party (origination) address that includes a subsystem number defined by the integer ssn.
	If not defined, and ORIG_PC and ORIG_GT are not defined, all InitialDPs will be sent without an SCCP calling party address.
ORIG_GT=1, n, addr or ORIG_GT=2, t, addr or ORIG_GT=3, t, p, addr or ORIG_GT=4, t, p, n, addr	If defined, all initialDPs will be sent with an SCCP calling party (origination) address that includes a Global Title defined by the integers n, t, p and the number string addr. The initial value (1 to 4) identifies the Global Title type: • n is the NOA • t is the translation type • p is the numbering plan • addr is the address digits (0 to 9, A to F) If not defined, and ORIG_PC and ORIG_SSN are not defined, all InitialDPs will be sent without an SCCP calling party address.
DEST_PC= pc	If defined, all initialDPs will be sent with an SCCP called party (destination) address that includes a Point Code defined by the integer pc.

Global Parameter	Description
	Note: This value may be defined in hex using a prefix of 0x.
DEST_SSN= ssn	If defined, all initialDPs will be sent with an SCCP called party (destination) address that includes a subsystem number defined by the integer ssn.
DEST_GT=1, n, addr or DEST_GT=2, t, addr or DEST_GT=3, t, p, addr or DEST_GT=4, t, p, n, addr	If defined all initialDPs will be sent with an SCCP called party (destination) address that includes a Global Title defined by the integers n, t, p and the number string addr. The initial value (1 to 4) identifies the Global Title type: • n is the NOA • t is the translation type • p is the numbering plan • addr is the address digits (0 to 9, A to F)
ACH WARN PERIOD=*period*	Sets the default ApplyCharging warning to occur *period* seconds before the end of the call.
ACH RESOURCE=*ad*	Sets the default ApplyCharging warning announcement/tone to use the resource identified by the address digits *ad*. **Note:** This is only applicable if the underlying controlled call supports the ability to play announcements or tones.
ACH ANNOUNCE=*messageId*	Causes the default ApplyCharging warning to use the announcement with message identifier *messageId*. **Note:** This is only applicable if the underlying controlled call supports the ability to play announcements or tones.
ACS TONE=*id,dur*	Causes the default ApplyCharging warning to use tone with identifier *id* for a duration of *dur* seconds. **Note:** This is only applicable if the underlying controlled call supports the ability to play announcements or tones.

Trigger detection point definitions

After any global parameters have been set, the configuration file may take one or more trigger detection point (TDP) definitions.

Each line defines a single trigger; its trigger parameter values that get sent and the conditions under which it gets sent.

Each line takes the following form:

```
tdp svcKey eventType msgType cgPn cdPn [wild] [keep]
```

The table below defines the meanings and forms of these parameters.

Global Parameter Value	Type	Description
tdp	integer	This integer value defines the point that the TDP is triggered at. Together with cgPn, *cdPn* and *wild* it defines the condition that the trigger will fire on. See the TDP event type table for a list of valid values and meanings.

Global Parameter Value	Type	Description
svcKey	integer	This parameter defines the serviceKey value that will be inserted into the initialDP message when this trigger fires.
eventType	integer	This parameter defines the eventTypeBCSM value that will be inserted into the InitialDP message when this trigger fires. See the TDP event type table for a list of valid values and meanings. Generally this will be the same value as *tdp*.
msgType	request or notify	This parameter defines whether the TDP is sent as a TDP-R (request) or TDP-N(notify). Generally request is used here.
cgPn	*num* or *nat.num* or all	This parameter defines the calling party numbers that will trigger the TDP. Together with *tdp*, *cdPn* and *wild* it defines the condition that the trigger will fire on. *num* defines the prefix of the calling party digits, numbers must begin with these digits for the trigger to fire. *nat* is optional and defines additionally a nature of address (NOA) of the calling party that must match for the trigger to fire. If not provided a nature of 2 (unknown) is assumed. If all is defined then ALL calling party numbers will match.
cdPn	*num* or *nat.num* or all	This parameter defines the called party numbers that will trigger the TDP. Together with *tdp*, *cgPn* and *wild* it defines the condition that the trigger will fire on. *num* defines the prefix of the called party digits, numbers must begin with these digits for the trigger to fire. *nat* is optional and defines additionally a nature of address (NOA) of the called party that must match for the trigger to fire. If not provided a nature of 2 (unknown) is assumed. If all is defined then ALL called party numbers will match.
wild	integer	This optional parameter defines the number of digits that must be present in the called party numbers before the TDP will trigger. Together with *tdp*, *cgPn* and *cdPn* it defines the condition that the trigger will fire on. If set the trigger will not fire until the called party number has this number of digits. **Note:** The *wild* parameter can be set to a special value of "stop". If it is set to this value, then the trigger will only fire when a stop digit is received.
keep	-	If this optional flag is defined then all numbers triggered by this TDP will keep their stop digits (if they have one).

TDP event type values

The following table defines the list of TDPs as defined by the CS-1 standard. It also defines the point at which the trigger will be instantiated by the NCC IN Call Model.

TDP	CS-1 Trigger Name	Call Model TDP Creation Point
1	origAttemptAuthorized	digitsReceived
2	collectedInfo	digitsReceived
3	analyzedInformation	digitsReceived
4	routeSelectFailure	released (cause != 16, 17, 18, 19, 21 or 31)
5	oCalledPartyBusy	released (Aparty, cause==17)
6	oNoAnswer	released (Aparty, cause==18, 19 or 21)
7	oAnswer	answered(Aparty)
8	oMidCall	not supported
9	oDisconnect	released (Aparty, cause==16 or 31)
10	oAbandon	released (Aparty, cause==16 or 31)
12	termAttemptAuthorized	digitsReceived
13	tCalledPartyBusy	released (Bparty, cause==17)
14	tNoAnswer	released (Bparty, cause==18, 19 or 21)
15	tAnswer	answered(Bparty)
16	tMidCall	not supported
17	tDisconnect	released (Bparty, cause==16 or 31)
18	tAbandon	released (Bparty, cause==16 or 31)
100	n/a	ringing (Aparty)
101	n/a	ringing (Bparty)