Oracle® Communications
Network Charging and Control
Provisioning Interface User's and Technical Guide
Release 5.0.1

June 2013
Copyright

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

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, the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. 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 Xeon 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, Opteron, the AMD logo, and the AMD Opteron 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 on 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. 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.

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

Intel and Intel Xeon 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, Opteron, the AMD logo, and the AMD Opteron 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 on 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. 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.
# Contents

About This Document .................................................................................................................. v
Document Conventions .............................................................................................................. vi

## Chapter 1

### System Overview ............................................................................................................... 1

Overview .................................................................................................................................. 1
Introduction to the Provisioning Interface ................................................................................. 1
PI Commands ............................................................................................................................... 4

## Chapter 2

### Configuration ....................................................................................................................... 5

Overview .................................................................................................................................. 5
Configuration Overview ............................................................................................................. 5
eserv.config Configuration ......................................................................................................... 6
Oracle- CCS commands ............................................................................................................. 29
Defining the Screen Language .................................................................................................. 33
Defining the Help Screen Language .......................................................................................... 34

## Chapter 3

### PI Administration Screen .................................................................................................. 37

Overview .................................................................................................................................. 37
PI Administration Screen .......................................................................................................... 37
PI Commands ............................................................................................................................... 38
PI Hosts ...................................................................................................................................... 40
PI MAC Pairs ............................................................................................................................... 43
PI Users ....................................................................................................................................... 45
PI Ports ....................................................................................................................................... 48

## Chapter 4

### PI Tester Screen ................................................................................................................ 53

Overview .................................................................................................................................. 53
PI Tester Screen ......................................................................................................................... 53
General ...................................................................................................................................... 54
Management Tests ..................................................................................................................... 56
Connection tests ......................................................................................................................... 57

## Chapter 5

### Background Processes ....................................................................................................... 59

Overview .................................................................................................................................. 59
PImanager .................................................................................................................................... 59
PIprocess ..................................................................................................................................... 61
PIbeClient ................................................................................................................................... 61
PIbatch ....................................................................................................................................... 62
PIbatch XML ............................................................................................................................... 63
Chapter 6
PI Management Commands - Numbes Users .............................................. 67
  Overview.................................................................................................................. 67
  Debug Command ................................................................................................... 67
  Traceon Command ................................................................................................. 68
  Traceoff Command ................................................................................................. 68
  State Command ..................................................................................................... 68
  Kill Command ......................................................................................................... 69
  Sendrate Command ................................................................................................ 70
  Logstats on/off Command ...................................................................................... 71

Chapter 7
About Installation and Removal ............................................................................. 73
  Overview.................................................................................................................. 73
  Installation and Removal Overview ....................................................................... 73
  Checking the Installation......................................................................................... 74

NCC Glossary of Terms ......................................................................................... 77

Index .......................................................................................................................... 81
About This Document

Scope
The scope of this document includes all the information required to install, configure and administer the PI application. It does not include detailed design of the service.

Audience
This guide was written primarily for system administrators and persons installing, configuring and administering the PI 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 prerequisite for safely using the information contained in this technical guide. Attempting to install, remove, configure or otherwise alter the described system without the appropriate background skills, could cause damage to the system; including temporary or permanent incorrect operation, loss of service, and may render your system beyond recovery.

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

Related documents
The following documents are related to this document:

- ACS Provisioning Interface Commands
- CCS Provisioning Interface Commands
- MM Provisioning Interface Commands
- NP Provisioning Interface Commands
- VPN Provisioning Interface Commands
- CCS Technical Guide
- SMS Technical Guide
- VWS Technical Guide
Document Conventions

Typographical Conventions

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

<table>
<thead>
<tr>
<th>Formatting convention</th>
<th>Type of information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Special Bold</td>
<td>Items you must select, such as names of tabs. Names of database tables and fields.</td>
</tr>
<tr>
<td>Italics</td>
<td>Name of a document, chapter, topic or other publication. Emphasis within text.</td>
</tr>
<tr>
<td><strong>Button</strong></td>
<td>The name of a button to click or a key to press. Example: To close the window, either click Close, or press Esc.</td>
</tr>
<tr>
<td><strong>Key+Key</strong></td>
<td>Key combinations for which the user must press and hold down one key and then press another. Example: Ctrl+P, or Alt+F4.</td>
</tr>
<tr>
<td><strong>Monospace</strong></td>
<td>Examples of code or standard output.</td>
</tr>
<tr>
<td><strong>Monospace Bold</strong></td>
<td>Text that you must enter.</td>
</tr>
<tr>
<td><strong>variable</strong></td>
<td>Used to indicate variables or text that should be replaced.</td>
</tr>
<tr>
<td>menu option &gt; menu option &gt;</td>
<td>Used to indicate the cascading menu option to be selected, or the location path of a file. Example: Operator Functions &gt; Report Functions. Example: /IN/html/SMS/Helptext/</td>
</tr>
<tr>
<td><strong>hypertext link</strong></td>
<td>Used to indicate a hypertext link on an HTML page.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction to the Provisioning Interface</td>
<td>1</td>
</tr>
<tr>
<td>PI Commands</td>
<td>4</td>
</tr>
</tbody>
</table>

Introduction to the Provisioning Interface

Introduction

The Provisioning Interface (PI) provides a mechanism for manipulating data in the SMF database using an API. It enables bulk or scripted operations on the SMF data to be completed, where a human operator using the Java administration screens would be inefficient or error-prone.

The PI provides a reliable, extensible, network aware interface based on interoperability standards (for example XML).
## Component diagram

This diagram shows the PI components and processes.

![Component Diagram](image)

## Component descriptions

This table describes the main components involved in the Provisioning Interface application.

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
<th>Further Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMF database</td>
<td>The main SMF database holds the configuration details which are updated by the PI and some PI configuration details. The SMF holds configuration data for: The system, Client accounts, Services.</td>
<td><em>SMS Technical Guide</em></td>
</tr>
<tr>
<td>PI Administration screens</td>
<td>Enables an administrator to interact with the SMF database. The PI screens enable you to add new PI commands, users and hosts to the system and send test messages to specific PI processes.</td>
<td><em>PI Administration Screens</em> (see &quot;PI Administration Screen&quot; on page 37)</td>
</tr>
<tr>
<td>PI Tester screens</td>
<td>Test the system by sending individual commands to PI processes.</td>
<td><em>PI Tester Screens</em> (see &quot;PI Tester Screen&quot; on page 53)</td>
</tr>
<tr>
<td>PImanager</td>
<td>Starts and stops PI processes. PImanager will start as many PI processes as specified in the PI_PORTS table on the SMF database.</td>
<td><em>PImanager</em> (on page 59)</td>
</tr>
<tr>
<td>PIprocess</td>
<td>Runs on an SMS listening at a specific port</td>
<td><em>PIprocess</em> (on page 61)</td>
</tr>
</tbody>
</table>
PIbatch

Sends multiple PI commands to the PI processes. PIbatch can take instructions from a batch file, enabling complex treatments of the data in the SMF database to take place.

PIbatch (on page 62)

PIbatch_XML

Sends multiple PI commands to the PI processes. PIbatch_XML takes instructions from XML and SOAP files.

PIbatch XML (on page 63)

eserv.config

The configuration file for PI.

Note: Not all installations require this file to be configured.

eserv.config (on page 6)

Process

This table describes the process involved in running a PI command.

Note: The security/authentication parts of this process will only happen if the security plug-in is active in your deployment.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 1    | The first message sent to the server to start a new connection is a login message.  
      | Note: All communication between the client and the SMS uses the ASCII, HTTP/1.1, or HTTPS/1.1 protocol over TCP/IP. This enables all messages to be passed on a single connection for the duration of the session. |
| 2    | On successful login, the server will optionally send a security token.  
      | If security is used, then this token should be included in future request messages to confirm authentication. |
| 3    | A client system sends PI command to the relevant network port on the SMS. |
| 4    | The server process will check the authentication state, if configured to do so, and pass the requested command to the appropriate handler. |
| 5    | The server will respond on the same network connection with a message containing the response. The response will consist of any returned data, and, if configured, the new security token for use in future messages to the PI. |
| 6    | An end session command will be sent from either the server or the client to the other party. |
| 7    | The session is closed. |

Triggering BPL tasks

This process describes how a PI command triggers a BPL task.

For more information about BPL tasks, see CCS Task Management User's Guide and CCS Technical Guide.

<table>
<thead>
<tr>
<th>Stage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A client system sends the CCSBPL command to the relevant network port on the SMS.</td>
</tr>
<tr>
<td>2</td>
<td>The PIProcess (on page 61) for that port calls the libPI_CCSBPL library and sends the request to the smsTrigDeamon to trigger the BPL task process.</td>
</tr>
</tbody>
</table>
3 When the BPL has been processed, the smsTrigDaemon returns the result of the command to PIprocess.

4 PIprocess translates the response into a PI command response for the CCSBPL command and returns it to the client system.

## PI Commands

### Introduction

The provisioning interface uses TCP/IP-based UNIX sockets to receive provisioning commands and parameters. These are translated into SQL commands that update prepaid application tables of the SMF and E2BE Oracle databases.

### PI command installation

The PI commands which are available depend on which packages were run when the PI was installed. For details about the commands which are available for your installation, see the PI commands guide for your applications.

**Example:** For the commands for CCS, see *CCS PI Commands Operations Guide*.

### Command package details

This table shows the functionality installed by each package.

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Required</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Framework to execute a PI command.</td>
<td>Mandatory</td>
<td>piSms</td>
</tr>
<tr>
<td>Core CCS commands and VWS client.</td>
<td>Mandatory</td>
<td>piCcsSms</td>
</tr>
<tr>
<td>Command definitions for a subscriber domain.</td>
<td>Optional</td>
<td>piSubscriberSms</td>
</tr>
<tr>
<td>Command definitions for a wallet domain.</td>
<td>Optional</td>
<td>piWalletSms</td>
</tr>
<tr>
<td>Command definitions for a voucher domain.</td>
<td>Optional</td>
<td>piVoucherSms</td>
</tr>
<tr>
<td>Command definitions for the Social Networking Service</td>
<td>Optional</td>
<td>piSrmSms</td>
</tr>
<tr>
<td>Template.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For more information about installing these packages, see Installation Procedure Overview.
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.

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
<th>Further information</th>
</tr>
</thead>
<tbody>
<tr>
<td>eserv.config</td>
<td>This file provides a centralized location for configuring NCC software, including PI. This file should be updated with any relevant details from the eserv.config.pi_example file added during installation.</td>
<td>eserv.config Configuration (on page 6)</td>
</tr>
<tr>
<td>PI administration screens</td>
<td>PI uses Java screens to administer user accounts, connections and commands. These screens will be populated with data entered during the installation, but may require additional configuration.</td>
<td>PI Administration Screens (see &quot;PI Administration Screen&quot; on page 37)</td>
</tr>
</tbody>
</table>
eserv.config Configuration

Introduction

The eserv.config file is a shared configuration file, from which many 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 up 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 may 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" ]
}
```

eserv.config files delivered

Most applications come with an example eserv.config configuration in a file called eserv.config.example in the root of the application directory, for example, /IN/service_packages/eserv.config.example.
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 will cause 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 will ensure you have a working copy to which you can return.

Loading eserv.config changes

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

eserv.config.pi_example

The PI comes with an example of the PI's eserv.config configuration in a file called eserv.config.pi_example in the root of the application directory. This example configuration should be copied into the main eserv.config file to provide a base for the application's configuration.

Optional sections in eserv.config

Some sections of the eserv.config file are only required if your deployment has a specific component. For example, the $PIbeClient$ section is required only if a VWS Voucher and Wallet Server is used.

eserv.config subsections

The $pi$ section of the eserv.config file has the following structure.

```
pi = {
    general = {},
    authentication = {},
    throttling = {},
    PIbeClient = {},
    ssl = {},
    soap = {}
}
```

Parameters for each subsection are described below.

General

Here is an example of the general sub section of the PI eserv.config configuration.

```
general = {
    # debug = 'N'
    # oraUser = "/
    # synstamp = 'Y'
    # timeout = 30
    # logLevel = 0
    # securityPlugin = ""
    # correlationRequestTagName = "CORRELATE"
    # correlationResponseTagName = "CORRELATE"
}
```
The parameters are described in detail below.

**correlationRequestTagName**

- **Syntax:** `correlationRequestTagName = "name"`
- **Description:** The expected correlation tag in the XML message.
- **Type:** String
- **Optionality:** Optional (default used if not set).
- **Allowed:**
  - Default: "CORRELATE"
- **Notes:**
  - Example: `correlationRequestTagName = "CORRELATE"

**correlationResponseTagName**

- **Syntax:** `correlationResponseTagName = "name"`
- **Description:** The expected correlation tag in the XML response.
- **Type:** String
- **Optionality:** Optional (default used if not set).
- **Allowed:**
  - Default: 'correlationRequestTagName' value
- **Notes:**
  - If not set it will be the value on 'correlationRequestTagName'.
- **Example:** `correlationResponseTagName = "CORRELATE"

**debug**

- **Syntax:** `debug = Y|N`
- **Description:** Turn debug on or off.
- **Type:** String
- **Optionality:** Optional (default used if not set).
- **Allowed:**
  - Default: N
- **Notes:**
  - Only turn on in extreme circumstances as it turns on debug for all parts of PI.
  - Command line equivalent: "-D" (Y), missing (N).
- **Example:** `debug = 'N'

**loglevel**

Switches the Plprocess trace on or off.

- **Default:** 0
- **Allowed:**
  - 1 switches the Plprocess trace on.
  - 0 switches the Plprocess trace off

**oraUser**

Defines the Oracle user name and password.

- **Default:** "/"
- **Note:**
  - This parameter is not usually required as the default "/" is correct.
  - Command line equivalent: "-u /"
securityPlugin
The name of any security plug-in used.
Default: ""
Allowed: string

synstamp
Turns the synstamp on or off.
Default: Y
Allowed: Y turns synstamp on
        N turns synstamp off
Note: Command line equivalent of "-S Y".

timeout
The maximum allowed time, in seconds, for a PI command. If, for example, you set timeout = 30, timeout will occur after 30 seconds.
Default: -1
Allowed: -1 No timeout.
        positive integer Seconds before timeout.
Note: The command line equivalent would be "-t 30".

authentication
The authentication subsection of the PI eserv.config configuration supports these parameters.

    authentication = {
        timeout = 0
    }

The parameter is described in detail below.

timeout
Syntax: timeout = seconds
Description: Authentication token timeout.
Type: Integer
Optionality: Optional (default used if not set).
Allowed:
Default: 0 (no timeout)
Notes:
Example: timeout = 0

Throttling
The throttling subsection of the PI eserv.config configuration supports these parameters.

    throttling = {
        sendRate = 0<int>
    }

The parameter is described in detail below.
sendRate
The maximum number of PI commands per second.
Default: 0
Allowed: 0 no limit
positive integer commands per second.

PlbeClient
The PlbeClient section of the eserv.config file configures the PlbeClient (on page 61) process. Here is the structure of the section.

```plaintext
PlbeClient = {
    namedEventCanSendDebitBalanceNegative = 'n|y'
    oracleLogin = "usr/pwD"
    beLocationPlugin = "lib"
    clientName = "name"
    heartbeatPeriod = microsecs
    messageTimeoutSeconds = seconds
    maxOutstandingMessages = int
    reportPeriodSeconds = seconds
    connectionRetryTime = seconds

    plugins = [
        {
            config="confStr",
            library="lib",
            function="str"
        }
    ]
    [...
}

confStr = {
    plug-in_configuration
}

notEndActions = [
    {type="str", action="[ACK |NACK]"
    }]
}
```

The parameters are described in detail below.

**clientName**

**Syntax:**
`clientName = "name"`

**Description:**
The unique client name of the process.

**Type:**
String

**Optionality:**
Mandatory

**Allowed:**
Must be unique.

**Default:**
The hostname of the local machine.

**Notes:**
The server generates clientId from a hash of `name`.
If more than one client attempts to connect with the same name, then some connections will be lost.
This parameter is used by libBeClientIF.

**Example:**
`clientName = "PlbeClient"`
connectionRetryTime

Syntax: \( \text{connectionRetryTime} = \text{seconds} \)

Description: The maximum number of seconds the client process will wait for a connection to succeed before attempting a new connection.

Type: Integer
Optionality: Required
Allowed: Default: 5
Notes: This parameter is used by libBeClientIF.
Example: \( \text{connectionRetryTime} = 2 \)

heartbeatPeriod

Syntax: \( \text{heartbeatPeriod} = \text{microsecs} \)

Description: The number of microseconds during which a Voucher and Wallet Server heartbeat message must be detected, or the BeClient process will switch to the other VWS in the pair.

Type: Integer
Optionality: Required
Allowed: Default: 3000000
Notes: 1 000 000 microseconds = 1 second.
If no heartbeat message is detected during the specified time, client process switches to the other Voucher and Wallet Server in the pair.
This parameter is used by libBeClientIF.
Example: \( \text{heartbeatPeriod} = 1000000 \)

maxOutstandingMessages

Syntax: \( \text{maxOutstandingMessages} = \text{num} \)

Description: The maximum number of messages allowed to be waiting for a response from the Voucher and Wallet Server.

Type: Integer
Optionality: Required
Allowed: Default: If this parameter is not set, the maximum is unlimited.
Notes: If more than this number of messages are waiting for a response from the Voucher and Wallet Server, the client process assumes the Voucher and Wallet Server is overloaded. In this event, the client process refuses to start new calls but continues to service existing calls.
The messages are queued until the Voucher and Wallet Server has reduced its outstanding load.
This parameter is used by libBeClientIF.
Example: \( \text{maxOutstandingMessages} = 100 \)
messageTimeoutSeconds

Syntax: 
messageTimeoutSeconds = seconds

Description: The time that the client process will wait for the server to respond to a request.

Type: Integer

Units: Seconds

Optionality: Required

Allowed: 1-604800 Number of seconds to wait.

0 Do not time out.

Default: 2

Notes: After the specified number of seconds, the client process will generate an exception and discard the message associated with the request.

This parameter is used by libBeClientIF.

Example: messageTimeoutSeconds = 2

namedEventCanSendDebitBalanceNegative

Determines whether a named billable event charge can be sent a negative debit balance.

Default: n

Allowed: y negatives will be allowed

Y negatives will be allowed

n only positives will be allowed

donEndActions

Syntax: notEndActions = [
    {type="str", action="[ACK|NACK]"}
    [...]]

Description: The notEndActions parameter array is used to define the messages associated with dialogs that should not have their dialog closes, because the dialog is closed by default. This facilitates failover.

Type: Parameter array.

Optionality: Required

Allowed: 

Default: 

Notes: If the incoming dialog for a call closes and the last response received was of the notEndActions type, the client process sends an ABRT message. The ABRT message allows the VWS to remove the reservation. An example of this situation would be where slee_acs has stopped working.

This parameter is used by libBeClientIF.

For more information about slee_acs, see ACS Technical Guide.

Example: notEndActions = [
    {type="IR", action="ACK"}
    {type="SR", action="ACK"}
    {type="SR", action="NACK"}
    {type="INER", action="ACK"}
    {type="SNER", action="ACK"}
    {type="SNER", action="NACK"}]
### action

**Syntax:**

**Description:** Action to take with a message.

**Type:**

**Optionality:**

**Allowed:**
- "NACK"
- "ACK"

**Default:**

**Notes:**

**Example:**

```
```

### type

The type of message.

### oracleLogin

Identifies the PIbeClient when it logs on to the database.

**Default:** "/" 

### plugins

**Syntax:**

```plaintext
plugins = [
    {
        config="
        library="lib"
        function="str"
    },
    ...
]
```

**Description:** Defines any client process plug-ins to run. Also defines the string which maps to their configuration section.

**Type:** Parameter array

**Optionality:** Optional (as plug-ins will not be loaded if they are not configured here, this parameter must include any plug-ins which are needed to supply application functions; for more information about which plug-ins to load, see the BeClient section for the application which provides the BeClient plug-ins).

**Allowed:** Empty (that is, do not load any plug-ins).

**Default:**

**Notes:**

The libclientBcast plug-in must be placed last in the plug-ins configuration list.

For more information about the libclientBcast plug-in, see libclientBcast.

This parameter is used by libBeClientIF.

**Example:**

```plaintext
plugins = [
    {
        config="broadcastOptions"
        library="libclientBcast.so"
        function="makeBroadcastPlugin"
    }
]
```
config

Syntax: \( \text{config} = \text{name} \)
Description: The name of the configuration section for this plug-in. This corresponds to a configuration section within the `plugins` section in the `eserv.config` file.
Type: String
Optionality: Mandatory (must be present to load the plug-in).
Allowed:
Default: No default
Notes:
Example: `config = \text{voucherRechargeOptions}`

function

Syntax: \( \text{function} = \text{str} \)
Description: The function the plug-in should perform.
Type: String
Optionality: Mandatory (must be present to load the plug-in).
Allowed:
Default: No default
Notes:
Example: `function = \text{makeVoucherRechargePlugin}`

library

Syntax: \( \text{library} = \text{lib} \)
Description: The filename of the plug-in library.
Type: String
Optionality: Mandatory (must be present to load the plug-in).
Allowed:
Default: No default
Notes:
Example: `library = \text{libccsClientPlugins.so}`

Voucher and wallet plugins

There are four plug-ins which provide functionality for the P1beClient:

1. Voucher recharge (VRW)
2. Voucher type recharge (VTR)
3. Merge wallets (MGW)
4. Broadcast (on page 20)

Note: The broadcast plug-in configuration must be placed last in the `plugins` configuration section.

Each plug-in can have a configuration section. The name of this subsection will match the string provided for the config parameter in the `plugins` subsection.

Example: The Voucher Recharge plug-in has config set to `voucherRechargeOptions`. So the configuration section for this plug-in is:

```
voucherRechargeOptions = {
    ...
}
```
reportPeriodSeconds

Syntax: \[ reportPeriodSeconds = \text{seconds} \]

Description: The number of seconds separating reports of failed messages.

Type: Integer

Units: Seconds

Optionality: Required

Allowed: 10

Default: BeClient issues a failed message report:

- For timed-out messages
- For unrequested responses
- For new calls rejected because of congestion
- For messages with invalid Voucher and Wallet Server identifiers
- If new and subsequent requests fail because both Voucher and Wallet Servers have stopped working

VWS heartbeat detection must be enabled for the parameter to work. Set reportPeriodSeconds to more than heartbeatPeriod.

Notes: This parameter is used by libBeClientIF.

Example: \[ reportPeriodSeconds = 10 \]

Voucher Recharge plug-in

The Voucher Recharge BeClient plug-in executes voucher recharges.

The plugins section must include the following configuration to load this plug-in.

```json
{  
  config="voucherRechargeOptions",  
  library="libccsClientPlugins.so",  
  function="makeVoucherRechargePlugin"  
}
```

Note: The VRW plug-in requires the broadcast plug-in.

The voucher recharge plug-in supports the following configuration.

```python
voucherRechargeOptions = {  
  srasActivatesPreuseAccount=true|false  
  voucherServerCacheLifetime = \text{seconds}  
  voucherServerCacheCleanupInterval = \text{seconds}  
  sendBadPin = true|false  
  voucherRechargeTriggers = [  
    "str"  
  ]  
}
```

The parameters are described in detail below.

sendBadPin

Syntax: \[ sendBadPin = \text{true}|false \]

Description: Whether or not to increment the Bad PIN count for a failed voucher redeem.

Type: Boolean

Optionality: Optional
Allowed: true Increment Bad PIN count for each failed attempt to recharge a voucher.
false Do not increment Bad PIN count for failed attempts to recharge a voucher.

Default: false
Notes: This parameter:
- applies only to an invalid voucher number or voucher PIN. It does not apply to failed wallet recharges.
- is part of the voucherRechargeOptions parameter group.

Example: sendBadPin = false

srasActivatesPreuseAccount
Syntax: srasActivatesPreuseAccount = true|false
Description: Sets whether or not alternate subscribers can activate subscriber accounts which are in a pre-use state.
Type: Boolean
Optionality: Optional

Allowed: true A scratch card alternate subscriber will be able to activate a pre-use account.
false A scratch card alternate subscriber will not be able to activate a pre-use account.

Default: true
Notes: This parameter is:
- Not used by ccsBeOrb
- Part of the voucherRechargeOptions parameter group

Example: srasActivatesPreuseAccount = false

voucherRechargeTriggers
Syntax: voucherRechargeTriggers = [ "VRW" ]
Description: This message triggers the voucher recharge plug-in.
Type: Array
Optionality: Mandatory

Allowed: VRW
Default:
Notes: This parameter array is part of the voucherRechargeOptions parameter group.

Example:

voucherServerCacheCleanupInterval
Syntax: voucherServerCacheCleanupInterval = seconds
Description: Time in seconds between purges of the voucher server id cache.
Type: Integer
Optionality: Optional

Allowed: Any positive decimal integer.
Default: 60 (seconds)
Notes:
Example: voucherServerCacheCleanupInterval = 60

voucherServerCacheLifetime
Syntax: voucherServerCacheLifetime = seconds
Description: Time in seconds to hold items in the voucher server ID cache.
Type: Integer
Optionality: Optional
Allowed: Any positive decimal integer.
Default: 600 (seconds)

Voucher Type Recharge plug-in
The Voucher Type Recharge PIbeClient plug-in executes voucher type recharges.
The plugins section must include the following configuration to load this plug-in.
{
    config="voucherTypeRechargeOptions",
    library="libccsClientPlugins.so",
    function="makeVoucherTypeRechargePlugin"
}

Note: The VTR plug-in requires the broadcast plug-in.

The voucher recharge plug-in supports the following configuration.
     voucherTypeRechargeOptions = {
     srasActivatesPreuseAccount=true|false
     voucherTypeRechargeTriggers = [
         "VTR "
     ]
     }

The parameters are described in detail below.

srasActivatesPreuseAccount
Syntax: srasActivatesPreuseAccount = true|false
Description: Sets whether or not alternate subscribers can activate subscriber accounts which are in a pre-use state.
Type: Boolean
Optionality: Optional

Allowed: true A scratch card alternate subscriber will be able to activate a pre-use account.

false A scratch card alternate subscriber will not be able to activate a pre-use account.
Default: true

Notes: This parameter is:
- Not used by ccsBeOrb
- Part of the voucherRechargeOptions parameter group

Example: srasActivatesPreuseAccount = false
voucherTypeRechargeTriggers

**Syntax:**

```python
type = str

[voucherTypeRechargeTriggers = [
    str [...]]
]
```

**Description:**
Starts the voucher type recharge plug-in.

**Type:** Array

**Optionality:** Mandatory

**Allowed:** VRW

**Default:**

**Notes:**
This parameter array is part of the voucherTypeRechargeOptions parameter group.

**Example:**

```python
voucherTypeRechargeTriggers = ["VTR "]
```

---

**Merge Wallets plug-in**

The Merge Wallets PlbeClient plug-in executes wallet merges. The `plugins` section must include the following configuration to load this plug-in.

```python
config = "mergeWalletsOptions",
library = "libccsClientPlugins.so",
function = "makeMergeWalletsPlugin"
```

**Note:** The VTR plug-in requires the broadcast plug-in.

The merge wallets plug-in supports the following configuration.

```python
mergeWalletsOptions = {
    oracleLogin = "usr/pwd"
    mergeBucketExpiryPolicy = "str"
    mergeWalletExpiryPolicy = "str"
    allowedSourceWalletStates = "str[,...]"
    mergeWalletsTriggers = [
        "str [...]]"
}
```

The parameters are described in detail below.

**allowedSourceWalletStates**

**Syntax:**

```python
allowedSourceWalletStates = "str[...]"
```

**Description:**
The states the source wallet must be in to allow it to be merged with another wallet.

**Type:** String

**Optionality:** Mandatory

**Allowed:**

- P Pre-use
- A Active
- D Dormant
- S Suspended
- F Frozen
- T Terminated

**Default:** None

**Notes:**
At least one state must be included, or all merged will be disallowed.

**Example:**

```python
allowedSourceWalletStates = "PA"
```
mergeBucketExpiryPolicy
Syntax:  mergeBucketExpiryPolicy = "str"
Description: Determines how the bucket expiry policy is treated.
Type: String
Optionality: Optional (default used if not set).
Allowed: merge policy is merged
move policy is moved
Default: merge
Notes:
Example: mergeBucketExpiryPolicy = "move"

mergeWalletExpiryPolicy
Syntax:  mergeWalletExpiryPolicy = "str"
Description: Determines the way expiry dates for merged wallets are managed.
Type: String
Optionality: Optional
Allowed: best The expiry date of the wallet with the most time left is used.
ignore The expiry date of the source wallet is ignored.
Default: best
Notes:
Example: mergeWalletExpiryPolicy = "best"

mergeWalletsTriggers
Syntax:  mergeWalletsTriggers = ["str [...]"
Description: Wallets of this type starts the merge wallets plug-in.
Type: Array of strings.
Optionality: Mandatory
Allowed: MGW
Default: None
Notes: The syntax must be typed exactly as shown in the example.
Example: mergeWalletsTriggers = ["MGW "]

oracleLogin
Syntax:  oracleLogin = "usr/pwd"
Description: The login details the BeClient should use to log in to the SMF database, when performing merge wallet functions.
Type: String
Optionality: Optional
Allowed:
Default: /
Notes:
Example: oracleLogin = "smf/smf"
Chapter 2

Broadcast plug-in

The Broadcast PlbeClient plug-in overrides the beLocationPlugin that would normally load connection details from the database.

The plugins section must include the following configuration to load this plug-in.

```json
{
    config="",
    library="libccsClientPlugins.so",
    function="makeBroadcastPlugin"
}
```

Notes:
- This plug-in must be the last in the plugins subsection.
- This plug-in has no configuration.
- The broadcast plug-in is required by the VRW and VTR plug-ins.

notEndActions

The state conversions subsection supports the following parameter.

```json
notEndActions = [
    {type="str", action="str"},
    ...
]
```

The parameter is described in detail below.

Parameter: notEndActions

Syntax: 
```json
notEndActions = [
    {type="str", action="[ACK|NACK]"},
    ...
]
```

Description: The notEndActions parameter array is used to define the messages associated with dialogs that should not have their dialog closes, because the dialog is closed by default. This facilitates failover.

Type: Parameter array.

Optionality: Required

Allowed:

Default:

Notes:
If the incoming dialog for a call closes and the last response received was of the notEndActions type, the client process sends an ABRT message. The ABRT message allows the VWS to remove the reservation. An example of this situation would be where slee_acs has stopped working.

This parameter is used by libBeClientIF.

For more information about slee_acs, see ACS Technical Guide.

Example:
```json
notEndActions = [
    {type="IR  ", action="ACK  "},
    {type="SR  ", action="ACK  "},
    {type="SR  ", action="NACK"},
    {type="INER", action="ACK  "},
    {type="SNER", action="ACK  "},
    {type="SNER", action="NACK"}
]
```
**voucherStateConversions**

The state conversions subsection supports the following parameter.

```plaintext
voucherStateConversions = {
    str = "ESCHER",
    ...
}
```

The parameter is described in detail below.

**voucherStateConversions**

**Syntax:**

```plaintext
voucherStateConversions = {
    str = "ESCHER",
    ...
}
```

**Description:** Converts from ESCHER encoding to a single character and back.

**Type:** Array

**Optionality:**

**Allowed:**

**Default:**

**Notes:**

**Example:**

```plaintext
voucherStateConversions = {
    A = "ACTV",
    F = "FRZN",
    R = "RDMD"
}
```

**stateConversions**

The state conversions subsection supports the following parameter.

```plaintext
stateConversions = {
    str = "ESCHER",
    ...
}
```

The parameter is described in detail below.

**stateConversions**

**Syntax:**

```plaintext
stateConversions = {
    str = "ESCHER",
    ...
}
```

**Description:** Converts from ESCHER encoding to a single character and back.

**Type:** Array

**Optionality:**

**Allowed:**

**Default:**

**Notes:**

**Example:**

```plaintext
stateConversions = {
    A = "ACTV",
    P = "PREU",
    D = "DORM",
    F = "FROZ"
}
```
\[ S = \text{"SUSP"}, \]
\[ T = \text{"TERM"} \]

**billingEngines**

The `billingEngines` subsection supports the following configuration.

```json
billingEngines = [
    {
        id = int,
        primary = { ip=\text{"ip"}, port=port },
        secondary = [{ ip=\text{"ip"}, port=port }]
    }
]
```

This section overrides connection details that `beLocationPlugin` obtains from the database. It identifies the Voucher and Wallet Servers and assigns their Internet connection details.

**Note:** This section is optional, and is often commented out.

The parameters are described in detail below.

**id**

**Syntax:**

\[ id = \text{int} \]

**Description:**
This unique identifier for this Voucher and Wallet Server configuration.

**Type:**
Integer

**Optionality:**
Required, if this section is used

**Allowed:**

**Default:**

**Notes:**
This parameter is part of the `billingEngines` parameter array.

**Example:**

\[ id = 1 \]

**primary**

**Syntax:**

\[ primary = \{ ip=\text{"ip"}, port=port \} \]

**Description:**
The `primary` parameter group defines the Internet protocol address and associated port number of the primary Voucher and Wallet Server.

**Type:**
Parameter array

**Optionality:**
Mandatory if this section is used

**Allowed:**

**Default:**

**Notes:**
This parameter is part of the `billingEngines` parameter array.

**Example:**

\[ primary = \{ ip=\text{"192.0.2.0"}, port=1500 \} \]

**secondary**

**Syntax:**

\[ secondary = \{ ip=\text{"ip"}, port=port \} \]

**Description:**
The `secondary` parameter group defines the Internet protocol address and associated port number of the secondary Voucher and Wallet Server.

**Type:**
Array

**Optionality:**
Required, if this section is used

**Allowed:**

**Default:**

**Notes:**
This parameter is part of the `billingEngines` parameter array.
Example:  
```
secondary = { ip="192.0.2.1", port=1500 }
```

**ip**

**Syntax:**  
```
ip = "ip"
```

**Description:** The internet address of the Voucher and Wallet Server.

**Type:** String

**Optionality:** Required

**Allowed:**

**Default:** None

**Notes:** This parameter is part of either the primary or secondary parameter group of the `billingEngines` parameter array.

**Example:**  
```
ip = "192.0.2.0"
```

**port**

**Syntax:**  
```
port = port
```

**Description:** The port number associated with the address of the Voucher and Wallet Server.

**Type:** Integer

**Optionality:** Required

**Allowed:**

**Default:** None

**Notes:** This parameter is part of either the primary or secondary parameter group of the `billingEngines` parameter array.

**Example:**  
```
port = 1500
```

**ssl**

Here is an example of the `ssl` subsection of the `peserv.config` configuration.

```
ssl = {
    certificateFile = "/IN/service_packages/PI/my_sslCertificate.pem"
    keyFile = "/IN/service_packages/PI/my_sslKey.pem"
}
```

The parameters in this subsection are described in detail below.

**certificateFile**

**Syntax:**  
```
certificateFile = "filename"
```

**Description:** The file name of the PEM Base64 encoded DER certificate to be used when accepting HTTPS connections.

**Type:** String

**Optionality:** Optional - only required if the interface used is SOAP/HTPPS.

**Allowed:**

**Default:** None

**Notes:**

**Example:**  
```
certificateFile = "/IN/service_packages/PI/sslCertificate.pem"
```
keyFile
Syntax: keyFile = "filename"
Description: The file name of the private key used to create the certificate.
Type: String
Optionality: Optional - only required if the interface used is SOAP/HTTP/SS.
Allowed: Default: None
Notes: Example: keyFile = "/IN/service_packages/PI/privKey.pem"

soap
Here is an example structure of the soap subsection of the PI eserv.config configuration.

    soap = {
        implicitLoginsSupported = false
        validateAuthStrings = true
        expansionRules = [
            {
                command = "CCSCD1"
                action = "QRY"
                parameter = "BALANCES"
                itemName = "BALANCE_ITEM"
                itemSeparator = "|"
                elementSeparator = ":"
                elementNames = [ "BALANCE_TYPE_NAME", "*BUCKETS" ]
            }
        ]
    }

These parameters are described in detail below.

validateAuthStrings
Syntax: validateAuthStrings = true|false
Description: Turning this variable false will bypass checking of AUTH strings.
Type: Boolean
Optionality: Optional (default used if not set).
Allowed: Default: true
Notes: That is useful in situations where a pool of connections is used for access to PI web services and the clients wish to be able to use any open connection for whichever command they wish execute next, regardless of which one they used for the preceding request.
Example: validateAuthStrings = true

implicitLoginsSupported

Syntax: implicitLoginsSupported = true|false

Description: Turning the following variable true will allow any incoming SOAP request to contain username and password. If such a request is received on an unauthenticated connection, the fields will be used to do an implicit Login (just like a real Login only no LoginResponse is sent).

If the login fails a fault is returned; if it succeeds the command is executed.

Type: Boolean
Optionality: Optional (default used if not set).
Allowed: false
Default: false
Notes: That is useful in situations where a pool of connections is used for access to PI web services and the clients wish to be free from knowing whether or not a particular connection requires authentication before use.

Example: implicitLoginsSupported = false

expansionRules

By default when returning PI responses in SOAP format, PI assembles them simplistically by using the parameter name as the tag name, and the value as the tag value. For example the name/value pair MSISDN="1234" is rendered

<pi:MSISDN>1234</pi:MSISDN>

However, some commands return complex, repeating, nested elements in a single response parameter, and the customer may wish to configure so-called expansion rules that cause these to be exploded out into a more XML-like and accessible style.

For any parameter of any command, an expansion rule can be configured, to explain how PI should unpack its value (normally by tokenizing on some separator such as ":"). The expansion rule concept allows for the situation where a list of struct-like items are assembled, with list items being separated by one sort of delimiter (e.g. ":") and the fields within each list item being separated by another (e.g. ":").

A rule is also allowed to treat an element as something which -itself- requires expansion, as shown in the CCSCD1=QRY BALANCES rule below.

Note: All these examples are real world in the sense that they can help provide a more usable rendering of the parameter values returned by PI in each case, but it is a matter of customer preference as to whether or not they are activated at a site.

PI command parameters, present or future, may require similar rules (or different ones).

Note: NCC truncates output for the CCSCD7 command at twenty kilobytes.

Here is an example of the expansion rules sub-section.

```plaintext
expansionRules = [ 

  { 
    command = "CCSCD1"
    action = "QRY"
    parameter = "BALANCES"
    itemName = "BALANCE_ITEM"
    itemSeparator = ":" 
    elementSeparator = ";;"
    elementNames = [ "BALANCE_TYPE_NAME", "*BUCKETS" ]
  }
]
```
eserv.config file example eserv.config.pi_example

PI comes with a file named `eserv.config.pi_example`. It is located in the root of the application directory. This file contains a commented example of the pi section of an `eserv.config` configuration file. As a starting point, when configuring features of the provisioning interface, copy `eserv.config.pi_example` into the main `eserv.config` file.

The content of the `eserv.config.pi_example` file is copied below. Most of the comments have been removed.

```plaintext
pi = {
    localTZ = "TimeZone"
    general = {
```

# debug = 'N'
# oraUser = "/
# synstamp = 'Y'
# timeout = 30
# logLevel = 0
# securityPlugin = ""
# correlationRequestTagName = "CORRELATE"
# correlationResponseTagName = "CORRELATE"

throttling = {
    # sendRate = 0
}

PIbeClient = {

    clientName = "PIbeClient"
    oracleLogin = "/"

    heartbeatPeriod = 10000000
    maxOutstandingMessages = 100
    connectionRetryTime = 2

    plugins = [ # pluggable functionality for the billing engine interface.
        # Voucher recharge (VRW) plugin (need the broadcast plugin)
        { # Voucher recharge (VRW) plugin (need the broadcast plugin)
            config="voucherRechargeOptions",
            library="libccsClientPlugins.sl",
            function="makeVoucherRechargePlugin"
        }
        # Broadcast plugin needed by VRW
        { # Broadcast plugin needed by VRW
            config="", # no config
            library="libclientBcast.sl",
            function="makeBroadcastPlugin"
        } # Broadcast one message to one BE of each pair
        # Activated by sending a message to BE ID 0.
    ]

    # Config for voucher recharge plugin
    voucherRechargeOptions = {
        # Should Scratch Card Alternate Subscriber activate a preuse a/c?
        # Not used by the PIbeClient
        srasActivatesPreuseAccount=false
        voucherRechargeTriggers = [
            "VRW " # this type of message triggers this plugin
        ]
    }

    notEndActions = [
        {type="IR ", action="ACK "}
        {type="SR ", action="ACK "}
        {type="SR ", action="NACK"}
        {type="INER", action="ACK "}
        {type="SNER", action="ACK "}
        {type="SNER", action="NACK"}
    ]

    stateConversions = {
        A = "ACTV",
        P = "PREU",
        D = "DORM",
}
F = "FROZ",
S = "SUSP",
T = "TERM"

voucherStateConversions = {
    A = "ACTV",
    F = "FRZN",
    R = "RDMD"
}

# billingEngines = [
#    {        id = 1, # pair ID
#        primary = { ip="PRIMARY_BE_IP", port=1500 },
#        secondary = { ip="SECONDARY_BE_IP", port=1500 }
#    }
#]

ssl = {
    certificateFile = "/IN/service_packages/PI/my_sslCertificate.pem"
    keyFile = "/IN/service_packages/PI/my_sslKey.pem"
}

soap = {
    implicitLoginsSupported = false
    validateAuthStrings = true
    expansionRules = [
    {
        command = "CCSCD1"
        action = "QRY"
        parameter = "BALANCES"
        itemName = "BALANCE_ITEM"
        itemSeparator = "|"
        elementSeparator = ":"
        elementNames = [ "BALANCE_TYPE_NAME", ";BUCKETS" ]
    }
    {
        command = "CCSCD1"
        action = "QRY"
        parameter = "*BUCKETS"
        itemName = "BUCKET_ITEM"
        itemSeparator = "|"
        elementSeparator = ":"
        elementNames = [ 
            "BUCKET_VALUE",
            "BUCKET_EXPIRY"
        ]
    }
    {
        command = "CCSCD7"
        action = "QRY"
        parameter = "EDRS"
        itemName = "EDR_ITEM"
        itemSeparator = "|"
        elementSeparator = ":"
        elementNames = [ 
            "RECORD_DATE",
            "WALLET_TYPE",
            "EDR_TYPE",
            "EDR_ID",
            "ACCOUNT",
            "ACCOUNT_NAME",
            "ADDRESS",
            "ADDRESS_LINE",
            "CITY",
            "STATE",
            "ZIP_CODE",
            "PHONE_NUMBER",
            "EMAIL_ADDRESS",
            "GENDER",
            "BIRTH_DATE",
            "SOCIAL_SECURITY_NUMBER",
            "NUMBER_OF_DEPENDENTS",
            "NUMBER_OF_RESIDENTS",
            "NUMBER_OF_UNITS"
        ]
    }
]
"CHARGING_DOMAIN_ID",
"CALL_ID",
"SCP_ID",
"SEQUENCE_NUMBER",
"EXTRA_INFORMATION"
]
}
{}
{
command = "CCSVR1"
action = "QRY"
parameter = "BALANCES"
itemName = "BALANCE_ITEM"
itemSeparator = "|"
elementSeparator = ":"
elementNames = [
"BALANCE_TYPE",
"AMOUNT",
"POST_USE_EXPIRY",
"START_DATE",
"END_DATE",
"NEW_BUCKET",
"POLICY",
"MISSING_BALANCE_POLICY",
"REPLACE_BALANCE"
]
}
|

Oracle- CCS commands

Introduction

Some of the application commands support configuration in the eserv.config file.

Note: If the commands package which includes the related command have not be installed, these parameters are not supported.

Local time zone

The Local time zone subsection supports the following parameter.

localTZ = "TimeZone"

The parameter is described in detail below.

localTZ

Syntax: localTZ = "TimeZone"

Description: Sets the time zone the PI uses for sending and receiving dates.

Type: String

Optionality: Optional (default used if not set).

Allowed: The time zone name must be a valid UNIX time zone name such as CET or GMT.

Default: The time zone of the SMS machine (typically GMT).

Notes: PI automatically accounts for summer time alterations within this time zone.
The CCSCD1 subsection of the PI eserv.config configuration supports these parameters:

\[
\text{CCSCD1} = \{
\text{ADD} = \{
\text{initialState} = \text{"state"},
\text{noWalletCreateBeIds} = [ 1, 2 ]
\}
\text{QRY} = \{
\text{currencyType} = \text{"Type"}
\}
\}
\]

The parameters in the ADD and QRY subsections are described in detail below.

**initialState**

Syntax: \text{initialState} = \"<state>\"

Description: The initial wallet state for wallets created using the \text{CCSCD1=}ADD\_INITIAL\_STATE command.

Type: \text{String}

Optionality: Optional.

Allowed:

- \text{P} Pre-use
- \text{A} Active
- \text{D} Dormant
- \text{S} Suspended
- \text{F} Frozen
- \text{T} Terminated

Default: \text{P}

Notes: For more information about the CCSCD1 command, see CCS PI Commands Operations Guide.

Example: \text{initialState} = \"P\"

**noWalletCreateBeIds**

Syntax: \text{noWalletCreateBeIds} = [ be\_ids ]

Description: Comma separated list of billing engine IDs that may not be used for wallet creation.

Type: \text{Array}

Optionality: Optional (default used if missing).

Allowed: List of valid billing engine IDs.

Default: Not set

Notes: Example: \text{noWalletCreateBeIds} = [ 1, 2 ]

**currencyType**

Syntax: \text{currencyType} = \"str\"

Description: Sets the type of currency.

Type: \text{String}

Optionality: Optional.

Allowed:

- \text{user} Use the user’s wallet currency type.
- \text{system} Use the system currency type.
Default: user
Notes: For more information about the CCSCD1 command, see CCS PI Commands Operations Guide.
Example: currencyType = "user"

CCSCD3
The CCSCD3 subsection of the PI eserv.config configuration supports these parameters.

```
CCSCD3 = {
    CTR = {
        creditTransferCP = "creditTransferControlPlan"
    }
    RCH = {
        fixedVoucherNumberLength = 10
        defaultScenarioName = "str"
        activatePreuseAccount = "true"
    }
}
```

The parameters in this subsection are described in detail below.

**creditTransferCP**
The name of the credit transfer control plan.
Default: "CREDIT_TRANSFER"
Type: String

**fixedVoucherNumberLength**
Syntax: fixedVoucherNumberLength = num
Description: The voucher number length. Must be set correctly when performing scenario recharges and the scenario name is specified.
Type: Integer
Optionality: Optional (default used if not set).
Allowed: Valid voucher number length
Default: 10
Notes: Example: fixedVoucherNumberLength = 10

**defaultScenarioName**
Syntax: defaultScenarioName = "scenario"
Description: Specifies the default scenario to use.
Type: String
Optionality: Optional (default used if not set).
Allowed: A valid scenario name.
Default: Default
Notes: Example: defaultScenarioName = "Default"
activatePreuseAccount
Syntax: activatePreuseAccount = "true|false"
Description: Sets whether or not to activate pre-use wallets for recharge attempts.
Type: Boolean
Optionality: Optional (default used if not set).
Allowed: true Activate pre-use wallets for recharge attempts.
false Do not activate pre-use wallets.
Default: true
Notes: Quoted value is required.
Example: activatePreuseAccount = "false"

CCSBPL
The CCSBPL subsection of the PI eserv.config configuration supports these parameters.

CCSBPL = {
  notifyEagain = true
  maxFifoReadRetry = 10
}

The parameters in this subsection are described in detail below.

notifyEagain
Syntax: notifyEagain = true|false
Description: Whether or not to display notice alarms for missed reads from the Fifo queue.
Type: Boolean
Optionality: Optional (default used if not set).
Allowed: true Display notice alarms for missed reads
false Do not display notice alarms for missed reads
Default: false
Notes: Example: notifyEagain = true

maxFifoReadRetry
Syntax: maxFifoReadRetry = maximum
Description: Sets the maximum number of times to retry reading from the Fifo queue.
Type: Integer
Optionality: Optional (default used if not set).
Allowed:
Default: 10
Notes: Example: maxFifoReadRetry = 20

CCSVR1
The CCSVR1 subsection of the PI eserv.config configuration supports these parameters.

CCSVR1 = {
  QRY = {
    suppressScenario = 'N'
  }
}
The parameters in this subsection are described in detail below.

**suppressScenario**

**Syntax:**
\[
\text{suppressScenario} = 'Y'|'N'
\]

**Description:**
Determines whether or not to suppress the SCENARIO return parameter if the voucher has an associated scenario.

**Type:**
Boolean

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

**Allowed:**
- Y: Do not return the SCENARIO return parameter even if the voucher has an associated scenario.
- N: Return the SCENARIO return parameter if the voucher has an associated scenario.

**Default:**
N

**Notes:**
If an invalid parameter value is specified, then 'N' is assumed and an error alarm is output to the PImanager log file.

**Example:**
\[
\text{suppressScenario} = 'N'
\]

---

**CCSSC1**

defaultBEDomainID

**Syntax:**
\[
\text{defaultBEDomainID} = id
\]

**Description:**
BE pair ID to query when no SUBSCRIBER supplied to query for their BE pair supplied for CCSSC1=QRY command.

**Type:**
Integer

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

**Allowed:**

**Default:**
-1

**Notes:**
Must match a value in the CCS.DOMAIN.DOMAIN_ID database table.
-1 = find the first domain that supports charging.

**Example:**
\[
\text{defaultBEDomainID} = 2
\]

---

**Defining the Screen Language**

**Introduction**

The default language file sets the language which the Java administration screens will start in. The user can change to another language once they have logged in.

The default language can be changed by the system administrator.

By default, the language is set to English. If English is your preferred language, you can skip this step and proceed to the next configuration task: Defining the Help Screen Language.
**Default.lang**

When PI is installed, a file called *Default.lang* is created in the application’s language directory in the screens module. This contains a soft-link to the language file which defines the language which will be used by the screens.

If a *Default.lang* file is not present, the *English.lang* file will be used.

The PI *Default.lang* file is `/IN/html/PI/language/Default.lang`.

**Example screen language**

If Dutch is the language you want to set as the default, create a soft-link from the *Default.lang* file to the *Dutch.lang* file.

**Procedure**

Follow these steps to set the default language for your PI Java administration screens.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 1    | Change to the following directory:  
      /IN/html/PI/language  
      **Example command:** `cd /IN/html/PI/language` |
| 2    | Check that the *Default.lang* file exists in this directory. |
| 3    | If the required file does not exist, create an empty file called *Default.lang*. |
| 4    | Ensure that the language file for your language exists in this directory. The file should be in the format:  
      *language.lang*  
      **Where:**  
      *language* = your language.  
      **Example:**  
      Spanish.lang |
| 5    | If the required language file does not exist, perform one of the following actions:  
      - Create a new one with your language preferences  
      - Contact Oracle support  
      To create a language file, you will need a list of the phrases and words used in the screens. These should appear in a list with the translated phrase in the following format:  
      *original phrase=translated phrase*  
      Any existing language file should have the full set of phrases. If you do not have an existing file to work from, contact Oracle support with details. |
| 6    | Create a soft link between the *Default.lang* file, and the language file you want to use as the default language for the PI Java administration screens.  
      **Example command:** `ln -s Dutch.lang Default.lang` |

**Defining the Help Screen Language**

**Introduction**

The default Helpset file sets the language which the help system for the Java Administration screens will start in. The user can change to another language once they have logged in.
The default language can be changed by the system administrator. By default, the language is set to English.

**Default_PI.hs**

When PI is installed, a file called `Default_PI.hs` is created in the application's language directory in the screens module. This contains a soft-link to the language file which defines the language which will be used by the screens.

If a `Default_PI.hs` file is not present, the `English_PI.hs` file will be used.

If a `Default_PI.hs` file is present, the default language will be used.

The default file is `/IN/html/PI/helptext/Default_PI.hs`.

**Example helpset language**

If Dutch is the language you want to set as the default, create a soft-link from the `Default_PI.hs` file to the `Dutch_PI.hs` file.

**Procedure**

Follow these steps to set the default language for your PI Java Administration screens.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 1    | Change to the following directory:  
/IN/html/PI/helptext  
**Example command:** `cd /IN/html/PI/helptext` |
| 2    | Check that the `Default_PI.hs` file exists in this directory. |
| 3    | If the required file does not exist, create an empty file called `Default_PI.hs`. |
| 4    | Ensure that the language file for your language exists in this directory. The file should be in the format: `language_PI.hs`  
**Where:**  
`language` = your language.  
**Example:**  
Dutch_PI.hs |
| 5    | If the required language file does not exist, perform one of the following:  
- Create a new one with your language preferences, or  
- Contact Oracle support.  
To create a language file, you will need a list of the phrases and words used in the screens. These should appear in a list with the translated phrase in the following format: `original phrase=translated phrase`  
Any existing language file should have the full set of phrases. If you do not have an existing file to work from, contact Oracle support with details. |
| 6    | Create a soft link between the `Default_PI.hs` file, and the language file you want to use as the default language for the SMS Java administration screens.  
**Example command:** `ln -s Dutch_PI.hs Default_PI.hs` |
Overview

Introduction

This chapter explains how to use the PI Administration screen.

In this chapter

This chapter contains the following topics.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>PI Administration Screen</td>
<td>37</td>
</tr>
<tr>
<td>PI Commands</td>
<td>38</td>
</tr>
<tr>
<td>PI Hosts</td>
<td>40</td>
</tr>
<tr>
<td>PI MAC Pairs</td>
<td>43</td>
</tr>
<tr>
<td>PI Users</td>
<td>45</td>
</tr>
<tr>
<td>PI Ports</td>
<td>48</td>
</tr>
</tbody>
</table>

PI Administration Screen

Introduction

The Administration screen manages users and templates in the Provisioning Interface. It contains these tabs:

- Users
- Hosts
- Ports
- Commands
- MAC Pairs

Accessing the PI Administration screen

Follow the steps below to access the Administration screen.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Select the Services menu from the Service Management System main menu.</td>
</tr>
<tr>
<td>Step</td>
<td>Action</td>
</tr>
<tr>
<td>------</td>
<td>--------</td>
</tr>
<tr>
<td>2</td>
<td>Select Provisioning.</td>
</tr>
<tr>
<td>3</td>
<td>Select Administration.</td>
</tr>
</tbody>
</table>

Result: You see the PI Administration screen, showing the Commands tab.

## PI Commands

### Introduction

The **Commands** tab of the Administration screen enables you to configure the commands which are available in Provisioning Interface.

**Note:** Commands cannot be added to or removed from the list.
Commands tab

Here is an example Commands tab.

![Commands tab screenshot](image)

Commands fields

This table describes the function of each field.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The PI command name.</td>
</tr>
<tr>
<td>Security Level</td>
<td>The security level required to execute the command.</td>
</tr>
<tr>
<td>Subscriber Domain</td>
<td>Indicates the command applies to an account that belongs to the subscriber domain, that is, the account exists on the SMS, the account wallet may be on a VWS or on a third party billing engine.</td>
</tr>
<tr>
<td>Wallet Domain</td>
<td>Indicates the command applies to an account that belongs to the wallet domain, that is, both the account and wallet exists on the SMS and VWS.</td>
</tr>
<tr>
<td>Voucher Domain</td>
<td>Indicates the command applies to a voucher that belongs to the voucher domain, that is, a voucher that exists on the VWS.</td>
</tr>
</tbody>
</table>
Example screen

Here is an example Edit screen for a PI command.

![Example Edit screen for a PI command]

Editing commands

Follow these steps to edit command information in the PI.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>On the <strong>Commands</strong> tab, select from the list the command to edit.</td>
</tr>
</tbody>
</table>
| 2    | Click **Edit**.  
**Result:** The *edit screen* (on page 40) displays the data for the chosen command record.  
See *Commands fields* (on page 39) for a description of each field. |
| 3    | Change the **Security Level** as required.  
**Note:** Range is 1 to 99 (highest) inclusive. |
| 4    | Click **Save**.  
**Result:** The details are saved to the database. |
| 5    | Soft restart the PI.  For details, see *Soft PI Restart* (on page 60).  
**Result:** The updated configuration details will be loaded by the PImanager. |

PI Hosts

Introduction

The **Hosts** tab of the Administration screen enables you to configure the hosts from which commands can be run in PI. Before a new client can connect, it must be added to the database.
Hosts tab

Here is an example Hosts tab.

<table>
<thead>
<tr>
<th>IP Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.187.246.41</td>
<td>eng-host35-29 SL C - for DAP</td>
</tr>
<tr>
<td>127.0.0.1</td>
<td>default</td>
</tr>
<tr>
<td>192.168.7.11</td>
<td>Lohil’s PC</td>
</tr>
<tr>
<td>192.168.7.93</td>
<td>Sam’s PC</td>
</tr>
</tbody>
</table>

Hosts fields

This table describes the function of each field.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP Address</td>
<td>The unique IP address of the host which will be allowed to run commands in the PI.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> This field cannot be changed after it is first saved.</td>
</tr>
<tr>
<td>Description</td>
<td>A description of the host defined in Internet Address. This can be set to the hostname of the host identified in Internet Address, though this data will not be used to connect to the host.</td>
</tr>
</tbody>
</table>
PI Hosts screen

Here is an example PI Hosts screen.

Adding hosts

Follow these steps to add new hosts to the PI.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 1    | On the Hosts tab, click New.  
Result: The PI Hosts screen (on page 42) displays. See Hosts fields (on page 41) for a description of each field. |
| 2    | Enter the IP Address field the IP address of the host. |
| 3    | Enter the Description field a description of the host. |
| 4    | Click Save.  
Result: The new host details are saved in the database. |
| 5    | Soft restart the PI.  For details, see Soft PI Restart (on page 60).  
Result: The updated configuration details will be loaded by the PImanager. |

Editing hosts

Follow these steps to edit host information in the PI.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>On the Hosts tab, select from the list the host to edit.</td>
</tr>
</tbody>
</table>
| 2    | Click Edit.  
Result: The PI Hosts screen (on page 42) fields is populated with the data for the selected host record. See Hosts fields (on page 41) for a description of each field. |
| 3    | Change the host Description as required. |
| 4    | Click Save.  
Result: The details are saved to the database. |
| 5    | Soft restart the PI.  For details, see Soft PI Restart (on page 60).  
Result: The updated configuration details will be loaded by the PImanager. |

Deleting hosts

Follow these steps to delete a host from the PI.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>In the Hosts tab, select from the list the host to delete.</td>
</tr>
<tr>
<td>Step</td>
<td>Action</td>
</tr>
<tr>
<td>------</td>
<td>--------</td>
</tr>
</tbody>
</table>
| 2    | Click **Delete**.  
  **Result:** The Delete Confirmation screen displays. |
| 3    | Click **OK**.  
  **Result:** The host is removed from the database. |
| 4    | Soft restart the PI. For details, see *Soft PI Restart* (on page 60).  
  **Result:** The updated configuration details will be loaded by the PImanager. |

### PI MAC Pairs

#### Introduction

The **MAC Pairs** tab of the Administration screen enables you to configure the MAC pairs from which commands can be run in PI. MAC pairs are the security keys to encode and decode encrypted data.

#### MAC Pairs tab

Here is an example **MAC Pairs** tab.
MAC Pairs fields

This table describes the function of each field.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAC Pair</td>
<td>The unique MAC pair number for this MAC pair.</td>
</tr>
<tr>
<td>Note:</td>
<td>This field cannot be changed after it is first saved.</td>
</tr>
<tr>
<td>MAC #1</td>
<td>The MAC address of the first MAC address in this MAC pair. This must be an 8 digit number.</td>
</tr>
<tr>
<td>MAC #2</td>
<td>The MAC address of the second MAC address in this MAC pair. This must be an 8 digit number.</td>
</tr>
</tbody>
</table>

PI MACS screen

Here is an example PI MACS edit screen.

Adding MAC Pairs

Follow these steps to add new MAC pairs to the PI.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>On the <strong>MAC Pairs</strong> tab, click <strong>New</strong>.</td>
</tr>
<tr>
<td></td>
<td><strong>Result</strong>: The <strong>PI MACS screen</strong> (on page 44) displays. <strong>See MAC Pairs fields</strong> (on page 44) for a description of each field.</td>
</tr>
<tr>
<td>2</td>
<td>Enter in the <strong>MAC Pair</strong> field the unique MAC pair number.</td>
</tr>
<tr>
<td>3</td>
<td>Enter in the <strong>MAC #1</strong> field the MAC address of the first entry for the MAC pair.</td>
</tr>
<tr>
<td>4</td>
<td>Enter in the <strong>MAC #2</strong> field the MAC address of the second entry for the MAC pair.</td>
</tr>
<tr>
<td>5</td>
<td>Click <strong>Save</strong>.</td>
</tr>
<tr>
<td></td>
<td><strong>Result</strong>: The new MAC pair details are saved in the database.</td>
</tr>
<tr>
<td>6</td>
<td>Soft restart the PI. For details, see <strong>Soft PI Restart</strong> (on page 60).</td>
</tr>
<tr>
<td></td>
<td><strong>Result</strong>: The updated configuration details will be loaded by the Plmanager.</td>
</tr>
</tbody>
</table>
Editing MAC Pairs

Follow these steps to edit MAC pair information in the PI.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>On the <strong>MAC Pairs</strong> tab, select from the list the MAC pair to edit.</td>
</tr>
</tbody>
</table>
| 2    | Click **Edit**.  
   **Result:** The **PI MACS screen** (on page 44) fields will be populated with the data for the selected MAC pair record. See **MAC Pairs fields** (on page 44) for a description of each field. |
| 3    | Change the MAC pair details as required. |
| 4    | Click **Save**.  
   **Result:** The details are saved to the database. |
| 5    | Soft restart the PI. For details, see **Soft PI Restart** (on page 60).  
   **Result:** The updated configuration details will be loaded by the PI manager. |

Deleting MAC Pairs

Follow these steps to delete a MAC pair from the PI.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>On the <strong>MAC Pairs</strong> tab, select from the list the MAC pair to delete.</td>
</tr>
</tbody>
</table>
| 2    | Click **Delete**.  
   **Result:** The Delete Confirmation screen displays. |
| 3    | Click **OK**.  
   **Result:** The MAC pairs are removed from the database. |
| 4    | Soft restart the PI. For details, see **Soft PI Restart** (on page 60).  
   **Result:** The updated configuration details will be loaded by the PI manager. |

PI Users

Introduction

The **Users** tab of the Administration screen enables you to configure the users who can run commands in PI. This tab defines users, passwords and security levels for the PI system.

The first command sent to the PI process should be a connect command, specifying the username and password of the user. Users can only access commands which have a security level less than or equal to their security level. Users can only use the MAC pair specified in their profile, and are restricted to using the port specified on the screen.
Users tab

Here is an example Users tab.

Users fields

This table describes the function of each field.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td>The unique username for this user.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> This field cannot be changed after it is first saved.</td>
</tr>
<tr>
<td>Security Level</td>
<td>The security level, 1 to 99 (inclusive) for this user. The user will be able to execute PI commands with security levels equal or lower than this number.</td>
</tr>
<tr>
<td>Port Number</td>
<td>The port number this user can connect from.</td>
</tr>
<tr>
<td>MAC Pair</td>
<td>The MAC pair this user can connect from.</td>
</tr>
<tr>
<td>Currency</td>
<td>The reporting currency for the user.</td>
</tr>
</tbody>
</table>
Chapter 3

**PI Users screen**

Here is an example PI Users screen.

![PI Users Screen](image)

**Adding users**

Follow these steps to add new users to the PI.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 1    | On the **Users** tab, click **New**.  
      | **Result:** The **PI Users screen** (on page 47) displays. See **Users fields** (on page 46) for a description of each field. |
| 2    | Enter the user name of the userid in the **User** field. |
| 3    | Enter (or get the user to enter) the user's password in the **Enter Password** field. |
| 4    | Enter (or get the user to re-enter) the user's password in the **Confirm Password** field. |
| 5    | Enter in the **Severity Level** field the command security level which this user can execute. |
| 6    | Select the **Port Number** from the drop down list the port the user can connect from. |
| 7    | Select the **MAC Pairs** from the drop down list the MAC pair the user should use. |
| 8    | Select the **Currency** from the drop down list the currency the user should use. |
| 9    | Click **Save**.  
      | **Result:** The new user details are saved in the database. |
| 10   | Soft restart the PI. For details, see **Soft PI Restart** (on page 60).  
      | **Result:** The updated configuration details will be loaded by the PI manager. |

**Editing users**

Follow these steps to edit user information in the PI.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>On the <strong>Users</strong> tab, select from the list the user to edit.</td>
</tr>
</tbody>
</table>
Step | Action
--- | ---
2 | Click **Edit**.  
**Result:** The *PI Users screen* (on page 47) is populated with the data from the selected user record. *See Users fields* (on page 46) for a description of each field.
3 | Change the user details as required.
4 | Click **Save**.  
**Result:** The details are saved to the database.
5 | Soft restart the PI. *For details, see Soft PI Restart* (on page 60).  
**Result:** The updated configuration details will be loaded by the PI manager.

**Deleting users**

Follow these steps to delete a user from the PI.

Step | Action
--- | ---
1 | On the **Users** tab, select from the list the user to delete.
2 | Click **Delete**.  
**Result:** The Delete Confirmation screen displays.
3 | Click **OK**.  
**Result:** The user is removed from the database.
4 | Soft restart the PI. *For details, see Soft PI Restart* (on page 60).  
**Result:** The updated configuration details will be loaded by the PI manager.

**PI Ports**

**Introduction**

The **Ports** tab of the PI Administration screen enables the configuration of the ports the PI processes listens on.
Ports tab

Here is an example Ports tab.

<table>
<thead>
<tr>
<th>Port</th>
<th>Secure</th>
<th>Max Connections</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>2999</td>
<td>N</td>
<td>10</td>
<td>Standard</td>
</tr>
<tr>
<td>3999</td>
<td>N</td>
<td>10</td>
<td>SOAP</td>
</tr>
</tbody>
</table>

Ports fields

This table describes the function of each field.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>The unique port number which will have a PI process listening on it.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> This field cannot be changed after it is first saved.</td>
</tr>
<tr>
<td>Secure</td>
<td>If Y, the port will be secure.</td>
</tr>
<tr>
<td></td>
<td>If N, the port will be insecure.</td>
</tr>
<tr>
<td>Max. Connections</td>
<td>The maximum number of concurrent connections to the port.</td>
</tr>
<tr>
<td>Type</td>
<td>The type of PI commands which can be run on this port.</td>
</tr>
</tbody>
</table>
Chapter 3

PI Ports screen

Here is an example PI Ports screen.

![PI Ports](image)

Adding ports

Follow these steps to add new ports to the PI.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 1    | Click **New** on the **Ports** tab.  
**Result:** The **PI Ports screen** (on page 50) displays. See **Ports fields** (on page 49) for a description of each field. |
| 2    | Enter in the **Port** field the port number. |
| 3    | Select the **Secure** check box if this port should be secure.  
Deselect the **Secure** check box if this port is not required to be secure. |
| 4    | Enter in the **Max. Connections** field the maximum number of concurrent connections this port will support. |
| 5    | Select the **Type** from drop down list the type of commands which can be run on this port. |
| 6    | Click **Save**.  
**Result:** The new port details are saved in the database. |
| 7    | Hard restart the PI. See **Hard PI Restart** (on page 60).  
**Result:** The new configuration details are loaded by the PI manager. |

Editing ports

Follow these steps to edit port information in the PI.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Select the port to edit from the <strong>Ports</strong> tab list.</td>
</tr>
</tbody>
</table>
| 2    | Click **Edit**.  
**Result:** The **PI Ports screen** (on page 50) is populated with the data from the selected port record. See **Ports fields** (on page 49) for a description of each field. |
| 3    | Change the port details as required. |
| 4    | Click **Save**.  
**Result:** The details are saved to the database. |
<p>| 5    | Hard restart the PI. See <strong>Hard PI Restart</strong> (on page 60). |</p>
<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Select the port to delete from the Ports tab list.</td>
<td>The new configuration details are loaded by the PI manager.</td>
</tr>
<tr>
<td>2</td>
<td>Click <strong>Delete</strong>.</td>
<td>The Delete Confirmation screen displays.</td>
</tr>
<tr>
<td>3</td>
<td>Click <strong>OK</strong>.</td>
<td>The port is removed from the database.</td>
</tr>
<tr>
<td>4</td>
<td>Hard restart the PI. See <em>Hard PI Restart</em> (on page 60).</td>
<td>The new configuration details are loaded by the PI manager.</td>
</tr>
</tbody>
</table>

**Deleting ports**

Follow these steps to delete a port from the PI.

- Select the port to delete from the Ports tab list.
- Click **Delete**.
- Click **OK**.
- Hard restart the PI. See *Hard PI Restart* (on page 60).
Overview

Introduction

This chapter explains how to use the PI Tester for standard ports screen.

In this chapter

This chapter contains the following topics.

PI Tester Screen 53
General 54
Management Tests 56
Connection tests 57

PI Tester Screen

Introduction

Use the PI Tester for standard ports screen to check that the PI commands are returning the correct results. It contains the following tabs:

- General
- Management
- Connection

Accessing the PI Tester screen

Follow these steps to access the PI Tester for standard ports screen.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Select the Services menu from the Service Management System main menu.</td>
</tr>
</tbody>
</table>
### General

**Introduction**

Use the **General** tab of the PI Tester for standard ports screen to modify general test attributes such as the test user and MAC address, and to view the results of management commands.

**Note:** The fields on the **General** tab are populated automatically. You only need to change them if you do not want to use the default value for a field.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Select <strong>Provisioning</strong>.</td>
</tr>
<tr>
<td>3</td>
<td>Select <strong>Tester</strong>.</td>
</tr>
</tbody>
</table>

**Result:** You see the PI Tester for standard ports screen.
General tab

Here is an example General tab.

![PI Tester for standard ports](image)

**General fields**

This table describes the function of each field.

**Note:** These fields are automatically populated with the default values.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Username</td>
<td>The user carrying out the tests. The user, defined in the Users tab of the Administration screen, must have the appropriate permissions. Typically, the admin user is used.</td>
</tr>
<tr>
<td>Password</td>
<td>The password for the user.</td>
</tr>
<tr>
<td>MAC key</td>
<td>The MAC key to use. This ensures the connection is secure.</td>
</tr>
<tr>
<td>PI Server</td>
<td>The PI server to connect to.</td>
</tr>
<tr>
<td>Port</td>
<td>The port to connect to. The list of available ports only includes port numbers defined for the PI server the screens came from.</td>
</tr>
<tr>
<td>Management responses</td>
<td>Displays the results of management commands entered on the Management tab of the PI Tester screen.</td>
</tr>
</tbody>
</table>

**Editing the General tab**

Follow these steps to change the general test attributes.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Select the <strong>General</strong> tab on the PI Tester for standard ports screen.</td>
</tr>
<tr>
<td>2</td>
<td>Change the values for the general attributes as required.</td>
</tr>
</tbody>
</table>
Management Tests

Introduction

Use the Management tab in the PI Tester for standard ports screen to send management commands to the PI server, using the values from the General tab. The following commands are available:

- **Kill** - to kill the PI connection for a selected user
- **State** - to see the current state of the PI commands, hosts and users
- **Trace** - to set up a trace on the port specified in the General tab

**Note:** The test responses are reported on the General tab.

Management tab

Here is an example Management tab.

Management fields

This table describes the function of each field.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User selection box</td>
<td>Lets you select a user from the drop down list. You can then kill the user's connection by clicking <strong>Kill</strong>.</td>
</tr>
<tr>
<td>Trace</td>
<td>Lets you switch tracing on, for the port specified on the General tab. The trace log, PI&lt;port&gt;.log is saved to the following directory:</td>
</tr>
<tr>
<td></td>
<td>- if the Plmanager was started with the init tab or the startup script, it is saved in /IN/service_packages/SMS.</td>
</tr>
<tr>
<td></td>
<td>- if the PImanager was started manually, it is saved in /IN/service_packages/MOB_PP/bin.</td>
</tr>
</tbody>
</table>
Using management tests

The following steps explain how to use the management tests.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>To kill a user’s connection, select the user from the list, and click <strong>Kill</strong>.</td>
</tr>
<tr>
<td>2</td>
<td>To find out the state of commands, hosts and users, click <strong>State</strong>.</td>
</tr>
<tr>
<td>3</td>
<td>To put a trace on the port currently selected in the <strong>General</strong> tab, select the <strong>Trace</strong> box.</td>
</tr>
</tbody>
</table>

Connection tests

Introduction

Use the **Connection** tab on the PI Tester for standard ports screen to enter commands directly, and check the results. The results appear in the results window on the **Connection** tab.

For PL/SQL commands, the parameters must be entered in the correct order. For C commands, the required parameters must be entered before the optional parameters. See your specific commands specification for details on the syntax to use for commands and the expected results.

**Note:** The commands you specify may alter the database, so you should use this facility with caution, especially when testing on a live database.

Connection tab

Here is an example **Connection** tab.
Using connection tests

Follow these steps to run tests from the Connection tab on the PI Tester for standard ports screen.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 1    | Click **Connect** to set up a connection to the PI server specified on the **General** tab in the PI Tester for standard ports screen.  
**Result:** The results of the connection attempt, including the systamp that will be used later, appear in the lower window. |
| 2    | In the upper text box, type in the commands you want to test, and click **Send**.  
You must use the following format:  
**COMMAND=ACTION:**REQUIRED_PARAMETER=VALUE,REQUIRED_PARAMETER=VALUE,OPTIONAL_PARAMETER=VALUE,SYNSTAMP=NUMBER  
**Result:** The commands are sent to the PI process and the results appear in the lower window. |
| 3    | Click **Disconnect** to disconnect from the PI server. |
Overview

Introduction

This chapter explains the PI processes which are used. The PImanager is started using the inittab, and the PIbatch process is started manually.

In this chapter

This chapter contains the following topics.

PImanager 59
PIprocess 61
PIbeClient 61
PIbatch 62
PIbatch XML 63

PImanager

Purpose

PImanager starts and stops PIprocesses as required.

Startup

PImanager can be started by either of two methods:

inittab script
The PImanager should normally be started from the inittab script.

/IN/service_packages/PI/bin/PImanagerStartup.sh

If PImanager is started this way, the output will be shown on the screen, rather than put in the log file.

Started directly
The process can be started directly, using the following code:

$ su - smf_oper
$ cd /IN/service_packages/PI/bin
$ ./PImanager [-u <user/password>] [-S Y|N] [-t <n>] [-M <m>] [-h] &

If PImanager is started this way, the output will be shown on the screen, rather than put in the log file.

Optional parameters
This table describes the optional parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-u username/password</td>
<td>/</td>
<td>The Oracle username and password.</td>
</tr>
</tbody>
</table>
### Chapter 5

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-S Y/N</td>
<td>Y</td>
<td>Turn on synstamp processing.</td>
</tr>
<tr>
<td>-t n</td>
<td>no timeout</td>
<td>Set the timeout to $n$ seconds.</td>
</tr>
</tbody>
</table>
| -M m     | use database setting | Override database settings for the PIprocess (on page 61) mode of operation:  
  0=Standard  
  1=XML with command mode  
  2=XML session mode  
  3=SOAP |
| -h       | Display usage and exit. |

**Shutdown**

To stop the Plmanager when it is running from the inittab file, use the utility script (as root user):  
`/IN/service_packages/PI/bin/Plstop.sh`.  
This will also stop all PIprocesses.  
To restart after stopping with Plstop.sh, use the utility script (as root user):  
`/IN/service_packages/PI/bin/Plstart.sh`.  
To stop the Plmanager when not running from the inittab file, send the Plmanager a kill-TERM signal.  
This will stop the Plmanager and any associated PIprocesses.

**Reinitializing the Plmanager**

The PI can be re-initialized using either a “hard” or “soft” reset. The preferred way should be to use the soft reset.

**Soft PI Restart**  
A soft reset is performed by using the utility script:  
`/IN/service_packages/PI/bin/Plreread.sh`.  
This causes the Plmanager to instruct the PIprocesses to re-read the database. However, the PIprocesses will not re-read the database until all connections have been dropped.  
**Note:** This will not cause the Plmanager to start any new PIprocesses added using the PI Ports (on page 48) screen. A hard reset must be done in this case.

**Hard PI Restart**  
To do a hard reset, for example, if new PIprocesses have been added using the PI Ports screen, use the script:  
`/IN/service_packages/PI/bin/Plrestart.sh`.  
This terminates the Plmanager and all PIprocesses, and the inittab will then restart them.  
**Note:** All connections to the PIprocesses will be lost.

**Failure**

If the Plmanager fails, no commands will be processed. All PIprocesses will also fail.

**Output**

The Plmanager writes error messages to the system messages file, and also writes additional output to `/IN/service_packages/PI/tmp/Plmanager.log`.  

---

60  NCC Provisioning Interface User’s and Technical Guide
**PIprocess**

**Purpose**

The PIprocess waits for TCP/IP connections, and processes commands sent to it. These commands can be management commands, such as "Connect", "Status" and "Disconnect", or PIcommands, such as "Query Subscriber".

The PIprocess checks that the user and remote host are valid, and then processes the command, either loading the command from a shared library, or executing a PL/SQL function in the database.

**Startup**

PIprocesses are started by the PImanager process.

If PIprocesses are added using the PI Port screens, the PImanager must be hard restarted to start the new PIprocess processes.

**Shutdown**

PIprocesses are shut down or restarted using the command scripts for the PImanager which started them, or by sending it a `kill -TERM` signal.

**Reinitialising a PIprocess**

To force a PIprocess to re-read the database, send it a `kill -HUP` signal. The PIprocess will re-read the database when all connections to it have closed. It is preferable to re-initialize the PImanager (see above) rather than individual PIprocesses.

**Configuration**

PIprocess is configured using PImanager’s configuration.

**Failure**

If PIprocess fails, PIcommands sent to the port that PIprocess is running on will fail.

**Output**

The PIprocess writes error messages to the system messages file, and also writes additional output to `/IN/service_packages/PI/tmp/PImanager.log`.

---

**PIbeClient**

**Purpose**

The PIbeClient interacts with billing engines. It is only available for installations which include CCS. If VWS is installed, PIbeClient will connect to the beServer on the Voucher and Wallet Server.

**CCS process**

PIbeClient is installed by the piCcsSms package. It will only be available on your system if you have installed CCS.
Chapter 5

**Startup**

PlbeClients are started by the PImanager process as needed.

**Shutdown**

PlbeClient are shutdown or restarted using the command scripts for the PImanager which started them, or by sending it a `kill -TERM` signal.

**Configuration**

The PlbeClient is configured in the `PI` section of `eserv.config`. For more information see `PlbeClient` (on page 10).

**Output**

The PlbeClient writes error messages to the system messages file, and also writes additional output to `/IN/service_packages/PI/tmp/PImanager.log`.

**Plbatch**

**Purpose**

The PI batch program is a utility to allow multiple PI commands to be sent to PIprocesses, with the commands being specified in a file.

**Startup**

Plbatch is started with the command line:

```
PIbatch -D script server
```

**Configuration**

The Plbatch accepts the following command line arguments.

**Usage:**

```
PIbatch script server
```

The available parameters are:

**Plbatch input files**

The Plbatch input scripts contain one command per line. You must order the commands as follows:

1. `!c port user password [synstamp] [mac]` - to instruct the Plbatch to connect to the PI on the given port number, using the specified username and password, with the optional synstamp, and with the MAC provided.
2. List the PI commands and Plbatch commands. The MAC and SYNSTAMP for commands are supplied automatically. **Note:** If you place a `";"` on the command line, you will have to include the MAC and SYNSTAMP in the command.
3. `!d` - to instruct the Plbatch to disconnect from the PI.

**Note:** When you place the `$` character at the beginning of a line, it is executed as a shell command.

**Example input file**

This is an example Plbatch input file.
Chapter 5

Chapter 5, Background Processes

!c 2999 admin admin 15111111
debug on
state
CCSCD4=CHG:MSISDN=1473111222,ADD=1234
!
d

Note: The MAC address in the "!c" connection string is constructed from the data in the MAC Pairs tab on the PI Administration screens. It is the MAC Pair number prefixed to either the first or second MAC address, as required. In this example, the MAC Pair number is 1 and the MAC Address is 51111111.

Failure

If PIbatch fails, the commands in the batch file will not be executed. Individual commands in the batch file can also fail.

Output

The results of the PI batch program are placed in a file. The file has the same name as PIbatch input file and a .result file extension.

The following text appears in the output file for each command sent to the PI:
"Running command command_name the_command_result End of output from command_name"

The output file can also contain some of the following information:

- the -> symbol followed by text sent to the PI,
- the -< symbol followed by text received from the PI,
- comments from the input script,
- and when the PIbatch disconnects from the PI, the word "Disconnected" is written to the output file.

Results file example

This is an example PIbatch results file.

->admin,admin;
<-ACK,SYNSTAMP=2005021010342483;
->CCSCD1=DEL:MSISDN=1107, SYNSTAMP=2005021010342484,MAC=135424;
<-CCSCD1=DEL:DELETEUser:NACK:1 MSISDN 1107 is not valid,SYNSTAMP=2005021010342484,MAC=114357;
Disconnected

PIbatch XML

Purpose

For PIbatch, the XML formatted PI commands are read from an input file, sent to the PIprocess, and results are optionally returned to a results file.

Startup

PIbatch_xml is started with the command line:

PIbatch.XML [-u username] [-p password] [-h hostname] [-n port_number] [-l loop] [-t throttle] [-f trace_file] [file...]

Chapter 5, Background Processes  63
Chapter 5

Configuration

The PIbatch_xml accepts the following command line arguments.

Usage:


The available parameters are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-u usr</td>
<td>admin</td>
<td>The username that should be used for logging into the PIprocess.</td>
</tr>
<tr>
<td>-p pwd</td>
<td>admin</td>
<td>The password that should be used when logging into the PIprocess.</td>
</tr>
<tr>
<td>-h host</td>
<td>localhost</td>
<td>The host name of the PI server.</td>
</tr>
<tr>
<td>-n port</td>
<td>2999</td>
<td>The port number of the PIprocess.</td>
</tr>
<tr>
<td>-l loop</td>
<td>1</td>
<td>How many times to loop through the commands.</td>
</tr>
<tr>
<td>-t throttle</td>
<td>no limit</td>
<td>Maximum number of requests per second.</td>
</tr>
<tr>
<td>-f trace_file</td>
<td>no file</td>
<td>File to append the PI responses to.</td>
</tr>
<tr>
<td>file</td>
<td>standard input</td>
<td>The input file. More than one may be specified.</td>
</tr>
<tr>
<td>-M mode</td>
<td>1</td>
<td>Mode of operation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1=XML</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 2=SOAP</td>
</tr>
<tr>
<td>-c chunk</td>
<td>Y</td>
<td>Stipulate chunking (Y</td>
</tr>
<tr>
<td>-i implicit</td>
<td>Y</td>
<td>Request implicit login (if Y, do not send initial Login request since the document is assumed to contain user credentials).</td>
</tr>
<tr>
<td>-S ssl</td>
<td>Y</td>
<td>Request SSL (secure) operation (value Y creates secure connection).</td>
</tr>
</tbody>
</table>

PIbatch_xml input file

The input file format is a list of XML formatted PI commands. Each input file can have more than one command. There may be more than one input file.

```xml
<?xml version="1.0"?><methodCall>
  <methodName>PI.OP</methodName>
  <params>
    <param><value><string>@TOKEN@</string></value></param>
    <param><value><string>command_name</string></value></param>
    <param><value><string>action_name</string></value></param>
    <param><value><struct>
      <member>
        <name>param_name</name>
        <value><string>param_value</string></value>
      </member>
    </struct></value></param>
  </params>
</methodCall>

@TOKEN@: Will be replaced with the authentication token by the PIbatch_xml program.
PIbatch_XML supports sending/receiving SOAP requests/responses. Here is a SOAP request.

PIbatch_XML SOAP input file.

```xml
<env:Envelope
    xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:enc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:ns1="http://uk.oracle.com/pi">
    <env:Body>
        <ns1:command_name>
            <ns1:AUTH>@TOKEN@</ns1:AUTH>
            <ns1:param_name1>param_value</ns1:param_name1>
            <ns1:param_name2>param_value</ns1:param_name2>
            <ns1:param_name3>param_value</ns1:param_name3>
            etc...
        </ns1:command_name>
    </env:Body>
</env:Envelop>
```

where `command_name` and `param_name` would be substituted with the actual PI command and parameters to be executed.

**SOAP Example:**

```xml
<env:Envelope
    xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:enc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:ns1="http://uk.oracle.com/pi">
    <ns1:CCSCD1_ADD>
        <ns1:AUTH>@TOKEN@</ns1:AUTH>
        <ns1:MSISDN>6122000193</ns1:MSISDN>
        <ns1:PROVIDER>Boss</ns1:PROVIDER>
        <ns1:PRODUCT>nzcl</ns1:PRODUCT>
        <ns1:CHARGING_DOMAIN>5</ns1:CHARGING_DOMAIN>
    </ns1:CCSCD1_ADD>
</env:Body>
</env:Envelope>
```

**Output**

The output file is simply written with the responses from the PI. See the relevant PI command definition for details.
Overview

Introduction

This chapter explains the PI management commands for users listed under the `numbes_users` section of `eserv.config`.

Note: While the commands are identical for numbes and AM users, the parameters and results are different. Numbes users can change GEN database tables, and AM users can change IPB database tables.

In this chapter

This chapter contains the following topics.

- Debug Command
  - Purpose
    - The PI can run in debug mode. You should use this mode only to trace faults.
    - Debug mode can be turned on or off for each component or command while the PI is running. To do this, in the PItester screen, send a debug command to the PIprocess.
    - The general list of components that can be specified is:
      - PIbatch (turn debug on in PIbatch)
      - PImanager (turn debug on in PImanager)
      - PIprocess (turn debug on in non-command parts of PIprocess)
      - PIcCommands (turn debug on in common parts of PI commands)
    - If piCcsSms is installed:
      - PIbeClient (turn debug on in non-beClientIF parts of PIbeClient)
    - Note: Refer to your specific Command Specification for details of all the available commands.
  - Format
    - The format of the debug command is:
debug {on/off} component1 [component2][component3] ... [component];
Component is the command, or command with the equals sign converted to an underscore.

Example:
- CCSCD1
- CCSCD1_ADD
- CCSCD1_DEL

**Example**
The following command examples turns debug on for CCSCD1:

To turn on debug for all CCSCD1 ADD commands, use:

```plaintext
debug on CCSCD1_ADD
```

To turn on debug for all CCSCD1 commands, use

```plaintext
debug on CCSCD1
```

**Note:** This also turns on PlcCommands but only for the command used, that is, CCSCD1_ADD.

**Output**
Debug prints output to the `/IN/service_packages/PI/tmp/PImanager.log` file.

**Traceon Command**

**Purpose**
The traceon command enables tracing of all PI commands. The tracing results are output to a file.

**Tip:** This command is now deprecated. We recommend you use the debug command instead.

**Format**
The format of the traceon command is:

```plaintext
traceon;
```

**Example**
The following command enables tracing of PI commands:

```plaintext
traceon;
```

**Output**
PI command tracing is output to the following file:

```plaintext
PIport_number.trace
```

**Traceoff Command**

**Purpose**
This command disables tracing.

**Tip:** This command is deprecated. We recommend you use the debug command instead.
Format
The format of the traceoff command is:
traceoff;

Example
The following command disables tracing of PI commands:
traceoff;

State Command

Purpose
Use the state command to print the current state of the PI process.

Format
The format of the state command is:
state;

Example
The following command prints the current state of the PI process:
state;

Output
The current state of the PI process is output to the following file:
PIport_number.state

Kill Command

Purpose
Use the kill command to kill a connection from a given username.

Format
The format of the kill command is:
kil username;

Example
The following command kills the connection from user bob:
kil bob;
Sendrate Command

Purpose

Use the sendrate command to specify the maximum number of PI commands that an individual PI process will send per second for processing by the billing engine. This allows each PI process to place only the desired load on the billing engine.

Example: Setting the sendrate for the port being used by PIbatch to the minimum rate of one, keeps its load to a minimum. This helps preserve the billing engine capacity for "live" usage.

Tip: The default sendrate can be set for all PI processes in eserv.config. For details, see eserv.config Configuration (on page 6).

Format

The format of the sendrate command is:

```
sendrate n;
```

The available parameter is:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td></td>
<td>The number of PI commands to send to the billing engine per second. This must be a whole number.</td>
</tr>
</tbody>
</table>

Tip: To turn throttling off, set the sendrate to 0 (zero).

Checking the sendrate

You can use the state command to check the sendrate for the PI processes. This reports the values for all connected PI processes. For details see State Command (on page 69).

P1batch sendrate

You can use the sendrate command to control the sendrate when using P1batch. You:

1. Add the sendrate command to the batch input file after the connect line
2. Add a second sendrate command before the disconnect line to reset the sendrate to its original value

Warning: This will affect all connections to this PI process.

Batch optimization

To optimize the batch, the sendrate can be calculated as follows. The resulting number must be rounded up to the nearest whole number:

```
sendrate = number of commands / maximum time for batch (seconds)
```

Example: For 40000 commands in 4 hours (14400 seconds) the sendrate = 40000 / 14400 = 2.8. The rounded up value for the sendrate is 3.

Using sendrate for performance

You can use the sendrate command to control the load put on the billing engine by the different PI processes.

This can be achieved by setting the default sendrate in eserv.config, and then overriding the default for each individual PI process requiring a different value.
To override the default values, connect to each PIprocess port in turn and set the sendrate to the desired value.

**Example**
In this example there are three PIprocesses running on ports 2999, 3000, 3001. PIbatch is running on port 3001 and it needs to run at a lower rate than the other processes so that it does not overload the billing engines. A higher rate is required for the process running on port 2999. This can be achieved in the following way:

Set the throttling parameter in `eserv.config` as:

```plaintext
pi = {
    throttling = {
        sendRate = 2
    }
}
```

Then in a PIbatch script define the sendrate for ports 3001 and 2999:

```plaintext
# start of PIbatch script
# set PIbatch port to lower rate
!c 3001 PIuser PIpassword mac_number
sendrate 1
state
!d

# set shop port 2999 to higher rate
!c 2999 PIuser PIpassword mac_number
sendrate 3
state
!d
# end of PIbatch script
```

### Logstats on/off Command

**Purpose**

Use the logstats on or off command to switch the output of the timing statistics on, or off, every 30 seconds.

The following five statistics are collected for each command, over a 30 second time period:

- Number of successful uses of the command
- Number of unsuccessful uses of the command
- Minimum response time (in milliseconds)
- Average response time (in milliseconds)
- Maximum response time (in milliseconds)

**Note:** These statistics are for a single period; they are not cumulative.

**Output**

The output from the statistics has the following format:

command name followed by the five statistics separated by a slash (/) character.

**Example Output:**

```
Jul 31 15:30:59 PIprocess:2999(1001) Statistics for last 30 seconds
(command:<command> <successful>/<failed>/<min>/<max>/<avg>):
```
Jul 31 15:30:59 PIprocess:2999(1001)  ACSCLI-ADD 0/0/0/0/0 ACSCLI-DEL 0/0/0/0/0 ACSCLI-QRY 0/0/0/0/0
Chapter 7

About Installation and Removal

Overview

Introduction

This chapter provides details of the installation and removal process for the application.

In this chapter

This chapter contains the following topics.

Installation and Removal Overview 73
Checking the Installation 74

Installation and Removal Overview

Introduction

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

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

PI packages

An installation of Provisioning Interface includes the following packages, on the SMS:

- piSms
- piCluster (if installing on a clustered SMS)
- piAcsSms
- piCcSms
- piSubscriberSms
- piVoucherSms
- piSrmSms
- piWalletSms
- piVpnSms
- piXmsSms
- npPISms

Packages and dependencies

The Prepaid Charging v3.0 on-line provisioning interface engine can be delivered in up to six packages:

- piSms - A framework to execute a PI command. This package is always required.
- piCcSms - Commands and UBE client. This package is always required.
• piSubscriberSms - Command definitions for a subscriber domain. This package is optional.
• piWalletSms - Command definitions for a wallet domain. This package is optional.
• piVoucherSms - Command definitions for a voucher domain. This package is optional.
• piSrmSms - Command definitions for the subscriber relationship manager. This package is optional.

Packages must be installed as shown in the hierarchy:

1. piSms
2. piCcsSms
3. The rest in any order

Packages have the hierarchy shown in the following diagram:

```
  piSms
   |
   v
 piCcsSms
   |
 piSubscriberSms  piWalletSms  piVoucherSms
```

### Updating the eserv.config file manually

Note that at the end of the installation script of some PI packages, there is note advising the installer to manually update the `eserv.config` file using the contents of the example config file. After installation, follow these instructions to configure the `eserv.config` file. For example:

Example configuration files have been installed to
```
/IN/service_packages/PI/etc/eserv.config.pi_example.CCS and
/IN/service_packages/PI/etc/eserv.config.pi_example.CCS.full
```
Please use these as a guide to setting up your runtime configuration file at
```
/IN/service_packages/eserv.config
```
for example. If there is no existing pi section in `eserv.config`, copy the entire contents of the `eserv.config.pi_example.CCS` file to the end of the runtime configuration file. If there is a pi section, copy the relevant items into the pi section.

Refer to `eserv.config Configuration` (on page 6) for details.

### Checking the Installation

**Introduction**

This topic provides a list of things which should be checked to ensure the installation was successful.

**Process list**

When the application is running correctly, the following processes will be being run by smf_oper:

• PImanager
• PIprocess (if PI has been set up on more than one port, there should be as many PIprocesses as configured ports)

**Note:** PIbatch may also be running.
Checking the commands

When you install a PI commands package, this inserts the new commands into new rows in the PI_COMMANDS database table. After completing the installation, check this table to ensure the new commands have been added.

Database tables

The following database tables should have been added to the SMF database:

- PI_COMMANDS
- PI_HOSTS
- PI_PORTS
- PI_MACS
- PI_USERS
NCC Glossary of Terms

ACS
Advanced Control Services configuration platform.

API
Application Programming Interface

BE
Billing Engine

CC
Country Code. Prefix identifying the country for a numeric international address.

CCS
1) Charging Control Services (or Prepaid Charging) component.
2) Common Channel Signalling. A signalling system used in telephone networks that separates
signalling information from user data.

Connection
Transport level link between two peers, providing for multiple sessions.

cron
Unix utility for scheduling tasks.

DTMF
Dual Tone Multi-Frequency - system used by touch tone telephones where one high and one low
frequency, or tone, is assigned to each touch tone button on the phone.

E2BE
Code used to designate some components and path locations used by the UBE.

GUI
Graphical User Interface

HTML
HyperText Markup Language, a small application of SGML used on the World Wide Web.
It defines a very simple class of report-style documents, with section headings, paragraphs, lists, tables,
and illustrations, with a few informational and presentational items, and some hypertext and multimedia.

HTTP
Hypertext Transport Protocol is the standard protocol for the carriage of data around the Internet.
IN
Intelligent Network

IP
1) Internet Protocol
2) Intelligent Peripheral - This is a node in an Intelligent Network containing a Specialized Resource Function (SRF).

IP address
Internet Protocol Address - network address of a card on a computer

ISDN
Integrated Services Digital Network - set of protocols for connecting ISDN stations.

Messaging Manager
The Messaging Manager service and the Short Message Service components of Oracle Communications Network Charging and Control product. Component acronym is MM (formerly MMX).

MM
Messaging Manager. Formerly MMX, see also XMS (on page 80) and Messaging Manager (on page 78).

MSISDN
Mobile Station ISDN number. Uniquely defines the mobile station as an ISDN terminal. It consists of three parts; the country code (CC), the national destination code (NDC) and the subscriber number (SN).

NP
Number Portability

Oracle
Oracle Corporation

PI
Provisioning Interface - used for bulk database updates/configuration instead of GUI based configuration.

PIN
Personal Identification Number

PL/SQL
Oracle's Procedural Language for stored procedures and packages.
SGML

SLC
Service Logic Controller (formerly UAS).

SMS
Depending on context, can be:
- Short Message Service
- Service Management System platform
- NCC Service Management System application

SN
Service Number

SOAP

SQL
Structured Query Language - a database query language.

SRF
Specialized Resource Function - This is a node on an IN which can connect to both the SSP and the SLC and delivers additional special resources into the call, mostly related to voice data, for example play voice announcements or collect DTMF tones from the user. Can be present on an SSP or an Intelligent Peripheral (IP).

SSP
Service Switching Point

TCP
Transmission Control Protocol. This is a reliable octet streaming protocol used by the majority of applications on the Internet. It provides a connection-oriented, full-duplex, point to point service between hosts.

VPN
The Virtual Private Network product is an enhanced services capability enabling private network facilities across a public telephony network.

VWS
Oracle Voucher and Wallet Server (formerly UBE).
XML

eXtensible Markup Language. It is designed to improve the functionality of the Web by providing more flexible and adaptable information identification.

It is called extensible because it is not a fixed format like HTML. XML is a ‘metalinguage’ — a language for describing other languages—which lets you design your own customized markup languages for limitless different types of documents. XML can do this because it's written in SGML.

XMS

Three letter code used to designate some components and path locations used by the Oracle Communications Network Charging and Control Messaging Manager (on page 78) service and the Short Message Service. The published code is MM (on page 78) (formerly MMX).
Index

A
- About Installation and Removal • 75
- About This Document • v
- Accessing the PI Administration screen • 39
- Accessing the PI Tester screen • 55
- ACS • 79
- action • 13
- activatePreuseAccount • 33
- Adding hosts • 44
- Adding MAC Pairs • 47
- Adding ports • 53
- Adding users • 50
- allowedSourceWalletStates • 19
- API • 79
- Audience • v
- authentication • 9

B
- Background Processes • 61
- BE • 79
- billingEngines • 22
- Broadcast plug-in • 15, 20

C
- CC • 79
- CCS • 79
- CCS process • 63
- CCSBPL • 33
- CCSCD1 • 31
- CCSCD3 • 32
- CCSSC1 • 34
- CCSVR1 • 33
- certificateFile • 24
- Checking the commands • 77
- Checking the Installation • 76
- Checking the sendrate • 72
- clientName • 10
- Command package details • 4
- Commands fields • 41, 42
- Commands tab • 41
- Component descriptions • 2
- Component diagram • 2
- config • 14
- Configuration • 5, 63, 64, 66
- Configuration components • 5
- Configuration file format • 6
- Configuration Overview • 5
- Connection • 79
- Connection tab • 60
- Connection tests • 59
- connectionRetryTime • 11
- Copyright • ii
- correlationRequestTagName • 8
- correlationResponseTagName • 8

D
- creditTransferCP • 32
- cron • 79
- currencyType • 31

E
- Database tables • 77
- debug • 8
- Debug Command • 69
- Default.lang • 35
- Default_PI.hs • 36
- defaultBEDomainID • 34
- defaultScenarioName • 32
- Defining the Help Screen Language • 36
- Defining the Screen Language • 34
- Deleting hosts • 45
- Deleting MAC Pairs • 48
- Deleting ports • 54
- Deleting users • 51
- Document Conventions • vi
- DTMF • 79

F
- E2BE • 79
- Editing commands • 42
- Editing hosts • 44
- Editing MAC Pairs • 47
- Editing ports • 53
- Editing the file • 7
- Editing the General tab • 57
- Editing users • 51
- eserv.config Configuration • 3, 5, 6, 72, 76
- eserv.config file example
  - eserv.config.pi_example • 27
- eserv.config files delivered • 6
- eserv.config subsections • 7
- eserv.config.pi_example • 7
- Example • 70, 71
- Example helpset language • 36
- Example input file • 64
- Example screen • 42
- Example screen language • 35
- expansionRules • 26

G
- Failure • 62, 63, 65
- fixedVoucherNumberLength • 32
- Format • 69, 70, 71, 72
- function • 14

H
- General • 7, 56
- General fields • 57
- General tab • 57
- GUI • 79
<table>
<thead>
<tr>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>reportPeriodSeconds • 15</td>
</tr>
<tr>
<td>Results file example • 65</td>
</tr>
</tbody>
</table>

**S**
- Scope • v
- secondary • 23
- securityPlugin • 9
- sendBadPin • 16
- sendRate • 10
- Sendrate Command • 72
- SGML • 81
- Shutdown • 62, 63, 64
- SLC • 81
- SMS • 81
- SN • 81
- soap • 25
- SOAP • 81
- Soft PI Restart • 42, 44, 45, 47, 48, 50, 51, 62
- SQL • 81
- srasActivatesPreuseAccount • 16, 18
- SRF • 81
- ssl • 24
- SSP • 81
- Started directly • 61
- Startup • 61, 63, 64, 65
- State Command • 71, 72
- stateConversions • 22
- suppressScenario • 34
- synstamp • 9
- System Overview • 1

**T**
- TCP • 81
- Throttling • 9
- timeout • 9
- Traceoff Command • 70
- Traceon Command • 70
- Triggering BPL tasks • 4
- type • 13
- Typographical Conventions • vi

**U**
- Updating the eserv.config file manually • 76
- Users fields • 49, 50, 51
- Users tab • 49
- Using connection tests • 60
- Using management tests • 59
- Using sendrate for performance • 72

**V**
- validateAuthStrings • 25
- Voucher and wallet plugins • 14
- Voucher Recharge plug-in • 15
- Voucher Type Recharge plug-in • 17
- voucherRechargeTriggers • 16
- voucherServerCacheCleanupInterval • 17
- voucherServerCacheLifetime • 17
- voucherStateConversions • 21
- voucherTypeRechargeTriggers • 18
- VPN • 81
- VWS • 81

**X**
- XML • 82
- XMS • 80, 82