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.
Contents

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

Chapter 1

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

Overview ........................................................................................................................................ 1
What is Data Access Pack? .............................................................................................................. 1
DAP Template Language .................................................................................................................. 4
Profile Tag Formats ......................................................................................................................... 8
XML and SOAP over HTTP/HTTPS .................................................................................................. 8
HP-SA ............................................................................................................................................. 11
XML Interface ................................................................................................................................ 12
WSDL ............................................................................................................................................ 14
Statistics ........................................................................................................................................ 15
Accessing the DAP application ....................................................................................................... 16

Chapter 2

Resources ........................................................................................................................................ 19

Overview ........................................................................................................................................ 19
Resources Screen ........................................................................................................................... 19
ASP ................................................................................................................................................. 20
Operations ...................................................................................................................................... 23
Operation Sets ............................................................................................................................... 34

Chapter 3

Import WSDL .................................................................................................................................. 37

Overview ........................................................................................................................................ 37
Import WSDL Screen ...................................................................................................................... 37
Import WSDL Files .......................................................................................................................... 38
Operation Request Configuration ..................................................................................................... 40
Operation Response Configuration .................................................................................................. 41

Chapter 4

Configuration ................................................................................................................................. 45

Overview ........................................................................................................................................ 45
Configuration Overview .................................................................................................................. 45
eserv.config Configuration ............................................................................................................. 46
DAP eserv.config configuration ....................................................................................................... 48
SLEE.cfg Configuration .................................................................................................................... 61
Configuration for Optimal Performance .......................................................................................... 61

Chapter 5

Background Processes .................................................................................................................... 63

Overview ........................................................................................................................................ 63
c_rehash ....................................................................................................................................... 63
dapIF ............................................................................................................................................. 64
About This Document

Scope
The scope of this document includes all the information required to install, configure and administer the Oracle Communications Network Charging and Control Data Access Pack application.

Audience
This guide was written primarily for system administrators and persons installing and administering the DAP application. The documentation assumes that the person using this guide has a good technical knowledge of the system.

Prerequisites
Although there are no prerequisites for using this guide, familiarity with the target platform would be an advantage.

A solid understanding of Unix and a familiarity with IN concepts are an essential prerequisite for safely using the information contained in this 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:

- Oracle Communications Network Charging and Control ACS User's Guide
- Oracle Communications Network Charging and Control CPE User's Guide
- Oracle Communications Network Charging and Control SMS Technical Guide
- Oracle Communications Network Charging and Control SMS User's Guide
- Oracle Communications Network Charging and Control SLEE Technical Guide
- Oracle Communications Network Charging and Control DAP PICS
- RFC 2616: Hypertext Transfer Protocol – HTTP/1.1
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>Button</td>
<td>The name of a button to click or a key to press.</td>
</tr>
<tr>
<td></td>
<td>Example: To close the window, either click Close, or press Esc.</td>
</tr>
<tr>
<td>Key+Key</td>
<td>Key combinations for which the user must press and hold down one key and then press another.</td>
</tr>
<tr>
<td></td>
<td>Example: Ctrl+P, or Alt+F4.</td>
</tr>
<tr>
<td>Monospace</td>
<td>Examples of code or standard output.</td>
</tr>
<tr>
<td>Monospace Bold</td>
<td>Text that you must enter.</td>
</tr>
<tr>
<td>variable</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.</td>
</tr>
<tr>
<td></td>
<td>Example: Operator Functions &gt; Report Functions</td>
</tr>
<tr>
<td></td>
<td>Example: IN/html/SMS/HelpText/</td>
</tr>
<tr>
<td>hypertext link</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.

What is Data Access Pack? 1
DAP Template Language 4
Profile Tag Formats 8
XML and SOAP over HTTP/HTTPS 8
HP-SA 11
XML Interface 12
WSDL 14
Statistics 15

What is Data Access Pack?

Introduction

Oracle Communications Network Charging and Control Data Access Pack (DAP) provides the capability to send requests to external Application Service Providers (ASP and optionally receive responses for further processing by the IN platform. The protocol that the system uses is determined by the ACS service library (libacsService).
Diagram

The following diagram shows the architecture of the DAP solution.

Synchronous and asynchronous connections

Responses over a connection are expected to be asynchronous if the <!--[!--CORRELATE--> or <!--[!--CORRELATE-ID--> tag is included in the template which specifies the request. In this case, only the ACK is checked in the (initial) synchronous response, and any later responses (using the same <!--[!--CORRELATE--> tag) are not checked. Asynchronous mode is not supported for HTTPS connections. The listening port only supports HTTP connections.

Responses over a connection are expected to be synchronous if the <!--[!--CORRELATE--> tag is not included in the template which specifies the request. In this case, DAP parses the synchronous response for component fields.

"Synchronous" mode for HTTPS is supported. This is done using openssl (on page 67) SSL sockets to encrypt a request/response pair to a remote HTTPS server.

**Note:** The server does not request any client-side authentication.

For more information about:

- Configuring operations, see Operations (on page 23).
- The <!--[!--CORRELATE--> tag, see Correlate (see "Correlation" on page 24)
Correlation

Correlation is a way to ensure that an asynchronous response is associated with an originating request. Requests that require an asynchronous response are identified by the setting of a correlated flag in the template.

HTTP and HTTPS Connections

DAP supports concurrent connections to multiple ASPs, using either HTTP or HTTPS. More than one of each type of connection can be open at once, including multiple HTTPS connections.

Supported protocols

This table describes the function of each field.

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOAP</td>
<td>Over HTTP or HTTPS. For more information about this protocol, see SOAP (on page 8).</td>
</tr>
<tr>
<td>XML</td>
<td>Over HTTP or HTTPS.</td>
</tr>
<tr>
<td>HPSA</td>
<td>HP-SA formatted XML messages over TCP. For more information about HP-SA handling, see HP-SA (on page 11).</td>
</tr>
<tr>
<td>PIXML</td>
<td>PI commands using XML. For more information about this protocol, see DAP and the PI.</td>
</tr>
</tbody>
</table>

Message flow

This table describes the message flow for a standard DAP message.

<table>
<thead>
<tr>
<th>Stage</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1     | When run in a control plan, the DAP Send Request feature node (dapMacroNodes (on page 65) and libdapChassisActions (on page 66)) sends a request over the SLEE to the interface specified by the InterfaceHandle (on page 59) parameter for the specified protocol (usually dapIF (on page 64)). The message is populated from the macro node configuration with the:  
  * Protocol  
  * Template ID  
  * List and number of request parameters  
  * "Wait flag"

  Note: FAST_KEY parameters are replaced with the actual run-time parameter. For example, the value <aCN> is replaced with the calling number. Other parameters are simply used "as is". |
| 2     | dapIF receives a DAP request event over the SLEE, and extracts the request template ID. dapIF uses the ID to query its cache or the SLC database for the XML request template details.  
  For each tag name/value pair found in the request, parameter substitution is performed. If dapIF determines a response is required, it searches the script for a correlation parameter. dapIF replaces the correlation tag with a unique ID. The correlation tag is set in the XML template using the XML tag with the string defined by correlationTagName (on page 51) (usually <!--CORRELATION-->). This id will be used to assign the incoming response.  
  For more information about: |
<table>
<thead>
<tr>
<th>Stage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>If a primary TCP/IP connection is already established with the ASP but no ASP interaction is associated with that socket (that is, dapIF is idle), then this connection can be re-used immediately and dapIF sends the message. If no connection is available (and the limit has not yet been reached), dapIF opens a TCP/IP connection to the ASP specified by the host and (optional) port number found in the destination URL.</td>
</tr>
<tr>
<td>4</td>
<td>dapIF wraps the XML in an HTTP header and footer. For more information about how the headers and footers are constructed, see Message header construction (on page 9).</td>
</tr>
<tr>
<td>5</td>
<td>If the wait flag of the incoming DAP request event was set to false, dapIF immediately sends a DAP response event to slee_acs. The DAP response has Operation Status set to true to indicate success. For HTTP and HTTPS connections, if no response is expected and the request was sent on a secondary connection, it is closed unless there is no primary connection, in which case it becomes the primary connection. For HP-SA connections, the connection is closed.</td>
</tr>
<tr>
<td>6</td>
<td>dapIF starts a timer with the sooner of the value specified in either the template or the socket timeout (connectionTimeout (on page 51)).</td>
</tr>
<tr>
<td>7</td>
<td>dapIF checks whether there is any incoming traffic on any of the TCP/IP sockets it has previously opened with ASPs. If a TCP/IP connection is already established with the ASP and there is an ASP interaction currently associated with that socket (that is, we are waiting for a response), then the new request from the ASP is queued. If activity comes from an ASP connection and dapIF is waiting for a response from the ASP, then the incoming TCP/IP traffic is accepted and data is read. The tags are parsed and a DAP response request is constructed. For more information about parsing messages from ASPs, see ASP message parsing (on page 10). For more information about parsing HP-SA messages, see HP-SA response messages (on page 12). dapIF sends the message back to libdapChassisActions over the SLEE.</td>
</tr>
<tr>
<td>8</td>
<td>When the DAP response event is received by libdapChassisActions, and the node exits based on whether the response was a Success or an Error.</td>
</tr>
</tbody>
</table>

**DAP and the PI**

The DAP provides the ability to automatically log in to the PI when using the PIXML protocol. This feature enables PI commands to be triggered from a control plan using the DAP Send Request feature node. The DAP interface will process the PI response before returning it to the DAP Send Request feature node.

For more information on DAP feature nodes, see NCC Feature Nodes Reference Guide.

**DAP Template Language**

**Introduction**

DAP uses a template language to describe the format of the messages (requests) that are sent to ASPs. This template language controls variable substitution and repetition of subtrees.
Variables

There are two formats that variables can take in the XML document.

1. An empty XML element:
   `<phone_number></phone_number>
   This is interpreted as a variable called `phone_number`

2. A specific format of text string:
   `<<$phone_number>>
   This is interpreted as a variable called `$phone_number`.

In addition to user defined variables, there are several other variables which are substituted automatically by the DAP interface.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;!--CORRELATE--&gt;</td>
<td>This is substituted with the correlation ID wrapped with an element specified by the <code>correlationTagName</code> (on page 51) configuration value.</td>
</tr>
<tr>
<td>&lt;!--CORRELATION_ID--&gt;</td>
<td>This is substituted with the correlation ID. The value of the correlation ID should be treated as an opaque, variable length string.</td>
</tr>
<tr>
<td>&lt;!--TIMESTAMP--&gt;</td>
<td>This is substituted with the current time, in the format <code>YYYYMMDDHHmmSS</code>, wrapped with an element specified by the <code>timestampTagName</code> (on page 58).</td>
</tr>
</tbody>
</table>

Repetition of subtrees

Using profile fields contained in array profile fields, it is possible to repeat sections of a DAP request template.

This is done through the use of the `dap_main_key` attribute. When a tree has the `dap_main_key` specified in the root, the subtree will be duplicated for each instance of the variable in `dap_main_key`.

For example:

The variable `FF_numbers` is configured to point to an array of three elements (121, 122, 123), the following template stub:

```
<number_list>
  <phone_number dap_main_key="FF_numbers"><<$FF_numbers>></phone_number>
</number_list>
```

This will result in the following template being sent to the ASP:

```
<number_list>
  <phone_number>121</phone_number>
  <phone_number>122</phone_number>
  <phone_number>123</phone_number>
</number_list>
```

Multiple variables

It is possible to have multiple variables in a repeated subtree. If there are not enough elements to provide each subtree with a different value, the first value in the array will be repeated for the remaining values.

Example:

$FF_number = (121,122,123)
$FF_shortCode = (555,666)
$FF_enabled = "Yes"

<number_details dap_main_key="FF_number">
    <phone_number>121</phone_number>
    <short_code>555</short_code>
    <enabled>Yes</enabled>
</number_details>

Will result in the following:

<number_details>
    <phone_number>122</phone_number>
    <short_code>666</short_code>
    <enabled>Yes</enabled>
</number_details>

Detailed example

This detailed example shows what is sent to the ASP given the variables and template used.

Variables:

CustomerName “Bill”
$CallTo “5551212”
$CallFrom “5557399”
$FF_list(5550000,5550001,5550002)

Template:

<ProvideDiscount>
    <RequestType>Regular Call</RequestType>
    <CustomerName>Bill</CustomerName>
    <Destination>5551212</Destination>
    <Source>5557399</Source>
    <FriendsAndFamily>
        <PhoneNumber dap_main_key="$FF_list">5550000</PhoneNumber>
        <PhoneNumber dap_main_key="$FF_list">5550001</PhoneNumber>
        <PhoneNumber dap_main_key="$FF_list">5550002</PhoneNumber>
    </FriendsAndFamily>
</ProvideDiscount>

The following will be sent to the ASP:

<ProvideDiscount>
    <RequestType>Regular Call</RequestType>
    <CustomerName>Bill</CustomerName>
    <Destination>5551212</Destination>
    <Source>5557399</Source>
    <FriendsAndFamily>
        <PhoneNumber>5550000</PhoneNumber>
        <PhoneNumber>5550001</PhoneNumber>
        <PhoneNumber>5550002</PhoneNumber>
    </FriendsAndFamily>
</ProvideDiscount>
Template contents

This is a list of the various DAP Templates used for real time notifications and their data contents:

**Wallet Expiry**
- TIMESTAMP
- NOTIFICATION_NAME
- WALLET_NAME
- CLI
- PRODUCT_TYPE
- OLD_STATE
- NEW_STATE

**Wallet State Change**
- TIMESTAMP
- NOTIFICATION_NAME
- WALLET_NAME
- CLI
- PRODUCT_TYPE
- OLD_STATE
- NEW_STATE

**Charging**
- TIMESTAMP
- NOTIFICATION_NAME
- WALLET_NAME
- CLI
- PRODUCT_TYPE
- BALANCE_TYPE
- BALANCE_UNIT
- COST
- OLD_BALANCE
- NEW_BALANCE

**Recharging**
- TIMESTAMP
- NOTIFICATION_NAME
- WALLET_NAME
- CLI
- PRODUCT_TYPE
- BALANCE_TYPE
- BALANCE_UNIT
- AMOUNT
- OLD_BALANCE
- NEW_BALANCE
Balance Expiry

- TIMESTAMP
- NOTIFICATION_NAME
- WALLET_NAME
- CLI
- PRODUCT_TYPE
- BALANCE_TYPE
- BALANCE_UNIT
- EXPIRED_AMOUNT
- OLD_BALANCE
- NEW_BALANCE

Profile Tag Formats

Introduction

The profile block values need to be converted to the receiving application's expected format, so that requests can be transmitted to other systems, and for them to communicate back.

This is impossible without a set of supported types and detailed information about what format the data is sent and received will be in.

Supported tag types

This table describes the formats and meanings of the supported ACS profile tags.

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STRING</td>
<td>Any character string.</td>
</tr>
<tr>
<td>NSTRING</td>
<td>String containing only digits, the letters A-F, and the characters # and *.</td>
</tr>
<tr>
<td>INTEGER</td>
<td>Signed base 10 integer, range -2147483648 to 2147483647 inclusive.</td>
</tr>
<tr>
<td>UINTeger</td>
<td>Unsigned base 10 integer, 0 to 4294967295 inclusive.</td>
</tr>
<tr>
<td>DATE</td>
<td>ISO 8601 time date-time format (eg: YYYYMMDDTHHHMMSS)</td>
</tr>
<tr>
<td>BOOLEAN</td>
<td>String containing &quot;1&quot; for true or &quot;0&quot; for false.</td>
</tr>
<tr>
<td>DISCOUNT</td>
<td>String of the following format: maxCharge,period1Discount,period2Discount</td>
</tr>
<tr>
<td>VXMLANN</td>
<td>A plain text string, it is opaque to ACS. ACS expects (but does not enforce) that it is a valid URL that can be used to retrieve a VXML document. For example: <a href="http://example.org/ExampleDocument.vxml">http://example.org/ExampleDocument.vxml</a></td>
</tr>
</tbody>
</table>

XML and SOAP over HTTP/HTTPS

SOAP

When creating an XML template through the Resources Screen (see "Resources" on page 19), the XML is parsed at a simple level for syntactical validity against the XML standard. Simple Object Access Protocol (SOAP) messages are formatted XML messages. DAP does not use syntactical parsing to check for properly formatted SOAP messages.
Parameter substitution

When dapIF (on page 64) is requested to send a message to an ASP, it resolves any variables in the template as follows:

- If the tag name starts with a \ and is then followed by a $, dapIF assumes the $ is part of an existing expression that does not require substitution.
- If the tag name starts with a $, every occurrence of the tag name is replaced by the tag value.
- If the tag name does not start with a $, dapIF looks for XML tags with the given name and sets their value accordingly.
- <date> fields are populated with the date in ISO 8601 format:
  \YYYY-MM-DDThh:mm:ss
- If dapIF determines a response is required, it searches the script for a correlation parameter and replaces it with a unique ID. This substitution results in a user-specific version of the template. The correlation tag is set in the XML template using the XML tag defined by correlationTagName (on page 51) (usually \<!--CORRELATION-->). This ID is used to assign the incoming response. The ID the CORRELATION takes place on has the form:
  \<CORRELATION>HHHHHHHHHHHHHHHHSSSSSSSSSSSSSSSS</CORRELATION>
Where:
- H is the hostID, 8 hex characters in length.
- T is the current time with accuracy of seconds, 8 hex characters in length.
- S is a sequence number from 0 to FFFFFFFF, generated by the macro node each time it is invoked which ensures that the resulting string is unique.

Example: For a correlated message with tag/value pairs of: $1 and 999, and MSISDN and 00441473289900:

\<!--CORRELATION-->
\<emergencyNumber>$1</emergencyNumber>
\<msisdn></msisdn>

This results in:

\<CORRELATION>abcdef121234561212345678</CORRELATION>
\<emergencyNumber>999</emergencyNumber>
\<msisdn>00441473289900</msisdn>

For more information about overall message handling, see Message flow (on page 3).

Message header construction

When dapIF is constructing a message to send to an ASP it constructs a message from an initial HTTP request line, followed by HTTP headers:

\POST path HTTP/1.1
\Host: host[:port]
\SOAPAction: url
\User-Agent: Oracle DAP
\Cache-Control: no-cache
\Pragma: no-cache
\Content-Type: text/html; charset=utf-8
\Content-Length: length
\clientUrl: http://<listenHost>:<listenPort>/ACK

Where:
- path has been extracted from the destination URL
- host has been extracted from the destination URL.
- port (if any) has been extracted from the destination URL.
- url is the full destination URL.
- length is the length (in bytes) of the template body

Notes:
The SOAPAction header line is only sent if the protocol associated with the ASP is set to S (meaning SOAP).

The clientUrl header is only sent if the listenHost (on page 52) parameter is configured.

The request line and each of the above HTTP headers is terminated by a CRLF sequence as specified in 5.1 of RFC 2616: Hypertext Transfer Protocol – HTTP/1.1.

The HTTP headers are followed by an empty line, consisting only of the CRLF sequence. The HTTP body (that is, the request template body), is sent to the ASP.

If the protocol associated with the ASP is set to S (meaning SOAP), the body is surrounded by the following SOAP header:

```xml
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?><soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmns:xsd="http://www.w3.org/2001/XMLSchema"
```

and the following SOAP footer:

```xml
</soapenv:Body></soapenv:Envelope>
```

For more information about overall message handling, see Message flow (on page 3).

**ASP message parsing**

The first line of the response buffer contains the HTTP response status line (as defined in 6.1 of RFC 2616: Hypertext Transfer Protocol – HTTP/1.1). The HTTP status code is extracted from the HTTP response status line. If it is not 200 (indicating success), an alarm is raised and a DAP response event is returned to libdapChassisActions (on page 66).

If the original DAP request event contained response tag names, these will have been stored in the ASP queue. dapIF parses the HTTP response body to retrieve the value associated with these tags. The DAP response event will have its Operation Status set to true to indicate success. The DAP response event will be sent on the SLEE dialog which the original DAP request event arrived from.

If there is correlation and the response is the:

- First response, then the socket is still closed but the request is queued using the key of the correlation ID and with a timeout of the timeout value for a response.
- Second response (that is, it contains a correlation ID), then that correlation ID is searched for in the queue. If found, a response is generated using the parameters found in the response XML.

**Note:** Correlation data is received on a ‘listening’ socket (these are defined by listenPort (on page 52)).

For more information about overall message handling, see Message flow (on page 3).

**HTTPS connections**

A new SSL connection is made whenever there is a queued request to be sent to the remote HTTPS server (that is, whenever an ASP connection is made on a secure connection). New connections start as HTTP, and are moved to HTTPS if a secure connection is established.

When the connection is opened:

- openssl (on page 67) (if not already initialized):
  - Loads our keys
  - Seeds the random number generator
  - Verifies the location of the certificates directory
- An SSL socket is created
- An SSL handshake is performed
- Server authentication (on page 11) is optionally performed
Note: An error is reported if a secure connection cannot be made, or server authentication fails. However, dapIF (on page 64) does not abort on these errors and continues to run for other response/request pairs on other ASPs.

On a database cache reread (if the ASP has been deleted or modified) the secure socket is shut down, and if needed, restarted. When dapIF exits normally it sends close_notify messages to the server for each open socket.

Server authentication

Server authentication against a public certificate provided by the remote HTTPS server is available on a per-ASP basis. If not configured, the SSL connection will only have handshaking performed which ensures a minimum of session keys are used for encrypting the traffic to the HTTPS server. This protects against snoop-style attacks.

Certificate checking

Certificate checking, when performed, will be done by checking the hostname from the URL in the ASP against the common name field in the public certificate from the remote server. This check ensures that more than just the names match, by establishing that the server is who it says it is by encrypting something with its private key that matches the locally-held public key in the public certificate. This protects against spoofing-style attacks.

No checking of the certificate chain to the certificate authority will be performed. We will trust any public certificate, and authenticate to it.

Response validation

Checking of the response from the HTTPS server is limited to checking whether the ACK is returned as HTTP 200. Any further lines of the response are read, but are not parsed.

Certificate management

The certificates are stored as *.pem files in the directory specified by certificatePath (on page 50). The dapReadyCertificates.sh (on page 69) tool prepares the certificates into the form required by openssl (on page 67).

Note: There is no need to have any certificates if server authentication is not turned on.

SOAP support over HTTP

DAP supports SOAP by allowing the use of HTTP 1.1 as a container protocol. The basic HTTP implementation only accepts HTTP/200 as a success response, treating other success messages such as “204 – No Content” as error conditions.

HP-SA

Introduction

HP-SA messages are generally handled the same way as XML and SOAP messages, though the contents of the messages are different, and HP-SA is not supported over HTTPS.

For more information about overall message handling, see Message flow (on page 3).
Parameter substitution

When dapIF (on page 64) is requested to send a message to an ASP, it resolves any variables in the template as follows:

- The message_id field (in the header of each request) will be generated as a number in the range 0 to MAXINT. The message_id field will be incremented for each successive request.
- The system time is used at the time of request construction to populate the date_time field.
- All other fields are either hard-coded in the message template, or populated using the profile field values provided as DAP parameters. For more information about how these parameters are populated, see Parameter substitution (on page 9).

HP-SA response messages

Response messages will be received on a new connection to the port defined by listenPort (on page 52). Once received, the connection will be closed. Two responses will be received in the following order for each request:

1. Command Received Acknowledgment
2. Command Processed Acknowledgment

Both have the same form (activation response).

Each response has a response ID that correlates with the request's message ID.

Response status/details command received

The following response status/details are possible for the command received acknowledgment:

- OK/<no details>
- NOK/Invalid XML

Response status/details command processed

The following response status/details are possible for the command processed acknowledgment:

- OK/<no details>
- NOK/Workload Failure
- NOK/<platform name>:Network Problem
- RB_OK/<platform name1>:OK | <error code>; ...;<platform nameN>:OK | <error code>
- RB_NOK/<platform name1>:OK | <error code>; ...;<platform nameN>:OK | <error code>

XML Interface

Description

The XML interface is a dedicated DAP interface that allows an XML script to be sent to an ASP as a request and receive another XML script as a response in order to be parsed.
Synchronous request

This diagram shows a simple example of a synchronous message flow.

Asynchronous request

This diagram shows a simple example of an asynchronous message flow.

Synchronous message flow

The following table describes the message flow between the XML Interface and ASP.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The XML interface sends a request to the ASP.</td>
</tr>
<tr>
<td>2</td>
<td>ASP returns an ACK and response on the same socket.</td>
</tr>
</tbody>
</table>

Note: In this example transaction, a response value is required, however a callback is not needed.

Asynchronous message flow

The following table describes the message flow between the XML Interface and ASP.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The XML Interface sends the request to the ASP. The request contains the clientUrl information.</td>
</tr>
<tr>
<td>2</td>
<td>The ASP returns an ACK on the same socket.</td>
</tr>
</tbody>
</table>
### Step-by-Step Actions

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>The ASP initiates a new request back to the XML Interface using the initial <code>clientUrl</code> information, on a new socket.</td>
</tr>
<tr>
<td>4</td>
<td>The XML Interface returns an ACK on the same socket.</td>
</tr>
</tbody>
</table>

### Message contents

The XML interface is responsible for issuing a TCP/IP based message to an ASP when it receives an XML request message from a client. The message contains the specified XML string and any substituted parameters.

The response from the ASP is parsed to retrieve any requested parameter strings and these are sent back to the client in the form of an XML response.

### WSDL

#### SOAP bindings

DAP supports WSDL 1.1 and Simple Object Access Protocol (SOAP) bindings. Since WSDL is a complex specification, some parts of the specification do not match with the capabilities of the DAP SOAP implementation.

The current SOAP implementation only allows templates to be created that make use of the `soap:body` and the `soap:fault` element. Therefore, the `soap:header` and `soap:headerfault` elements in the WSDL SOAP bindings are not supported. For more information on DAP support for SOAP bindings, see *NCC Data Access Pack PICS Guide*.

**Note:** OSD will report an error when `soap:header` or `soap:headerfault` is encountered in the WSDL file.

#### XSD support

XSD is supported by DAP as a type definition language.

WSDL provides an extension format allowing several different type languages to be used to describe the format of the messages used by the services, however it recommends the use of XSD. Since the XSD standard is 300 pages long, DAP only supports the use of XSD as the type definition language.

**Note:** DAP will report an error if a type definition language other than XSD is encountered in the WSDL file.

#### WSDL styles

Web Services Description Language (WSDL) allows several different encodings to be specified, each resulting in the message being presented with a different style.

DAP attempts to be as permissive as possible with the style and use declarations in accordance with the WSDL document.

DAP supports the following styles:

- `style=rpc`
- `style=document`

DAP supports the following use declarations:

- `use=literal`
- `use=encoded`
Other encodings

Since the SOAP specification allows different encodings other than XML to be used to transfer the information, it is necessary to specify which encodings DAP will support. DAP only supports the soap-encoding.

However, even in soap-encoding, there are issues with the protocol. The portions of the standard that define how low-level types are encoded are supported but encoding data as references is not supported.

Note: DAP will report an error if an encodingStyle other than soap-encoding is encountered in the WSDL file.

Transmission services

DAP supports the One-way transmission primitive as well as the Request-response transmission primitive.

WSDL supports the definition of services where the view of the service is from the client. However, SOAP does not typically support these definitions. To determine which side of the protocol is being defined, DAP only supports WSDL files generated for a server in the following cases where the server receives a request:

- With no response
- And provides a response

Statistics

Introduction

The DAP macro-node (dapMacroNodes (on page 65)) collects statistics using the standard Service Management System statistic mechanism and stores them to the SMF database. Refer to SMS Technical Guide for details on how the statistics are collected.

Statistics collected

This table describes the statistics that are collected.

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OP-SENT</td>
<td>Count of the number of new requests sent to an ASP. This statistic is incremented each time a new request is sent to an ASP. The name of the template used is put in the statistics “details” field.</td>
</tr>
<tr>
<td>OP-SUCCESS</td>
<td>This statistic is incremented each time an ASP returns a successful response. The name of the template used is put in the statistics “details” field.</td>
</tr>
<tr>
<td>OP-FAIL</td>
<td>This statistic is incremented each time an ASP returns a failure response. The name of the template used is put in the statistics “details” field.</td>
</tr>
</tbody>
</table>

Note: These statistics have a period of 300 seconds.

ASP based statistics

ASP is recorded in the detail field for ASP based statistics.
The table describes the ASP statistic generated for each statistic, if the ASP column indicates "Y". The error values shown here returned by the XML interface to the node.

<table>
<thead>
<tr>
<th>ASP</th>
<th>Statistic</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>Success</td>
<td>-</td>
</tr>
<tr>
<td>Y</td>
<td>Unable to connect to ASP</td>
<td>connectconnect</td>
</tr>
<tr>
<td>Y</td>
<td>Unable to send request to ASP</td>
<td>cannotwrite</td>
</tr>
<tr>
<td>Y</td>
<td>ASP Protocol Failure (HTTP Error or ASP closed the connection prior to initial HTTP/ response)</td>
<td>asperror</td>
</tr>
<tr>
<td>Y</td>
<td>ASP closed the connection while the response is being read</td>
<td>noresponse</td>
</tr>
<tr>
<td>Y</td>
<td>An operation has timed out waiting for a response</td>
<td>responsetimeout</td>
</tr>
<tr>
<td>Y</td>
<td>An expected parameter is missing from the response</td>
<td>missingparam</td>
</tr>
<tr>
<td>Y</td>
<td>A valid correlation ID is missing from the response</td>
<td>malformeddata</td>
</tr>
<tr>
<td>Y</td>
<td>A reply to the request has been received after the call back is received</td>
<td>sequenceerror</td>
</tr>
<tr>
<td>Y</td>
<td>A request has been resent</td>
<td>retryattempt</td>
</tr>
<tr>
<td>Y</td>
<td>Too many requests are waiting for callbacks from an ASP</td>
<td>maxqueuesizerreached</td>
</tr>
</tbody>
</table>

Notes:
- Only external errors will be reported as statistics.
- The total number of requests generated by the system are recorded as a statistic.
- The total number of requests for each ASP are recorded as a statistic.

**Accessing the DAP application**

**Introduction**

The Data Access Pack (DAP) application screens are accessed using the Service Management System screens.

To begin configuring the DAP application, the SMS screens must first be configured and running. For more information about how to set up the SMS screens, see *SMS User's Guide*. 

---

16  NCC Data Access Pack User's and Technical Guide
SMS main menu

Here is an example of the Service Management System main menu showing the DAP menu options.

![SMS main menu](image)

DAP screens

There are two DAP screens:

- *Resources* (on page 19)
- *Import WSDL* (on page 37)
Overview

Introduction

This chapter explains how to use the Resources screen to configure Application Service Providers (ASP) and make operations available to DAP.

In this chapter

This chapter contains the following topics.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resources Screen</td>
<td>19</td>
</tr>
<tr>
<td>ASP</td>
<td>20</td>
</tr>
<tr>
<td>Operations</td>
<td>23</td>
</tr>
<tr>
<td>Operation Sets</td>
<td>34</td>
</tr>
</tbody>
</table>

Resources Screen

Introduction

The DAP Resources screen is used to configure the Oracle Communications Network Charging and Control Data Access Pack DAP application. It contains these tabs:

- **ASP** (on page 20)
- **Operations** (on page 23)
- **Operation Sets** (on page 34)

Accessing the Resources screen

Follow these steps to open the DAP Resources screen.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Select the <strong>Services</strong> menu from the SMS main screen.</td>
</tr>
<tr>
<td>2</td>
<td>Select <strong>DAP</strong>.</td>
</tr>
<tr>
<td>3</td>
<td>Select <strong>Resources</strong>.</td>
</tr>
</tbody>
</table>
Step | Action
--- | ---

The **ASP** tab allows you to specify the ASPs that are available to the DAP application.

**ASP tab**

The ASP tab gives a view of all the ASPs that have been created for the service. Here is an example.

**ASP tab fields**

This table describes the function of each field. The table is sorted by the Name field.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>(Required) Unique name for this ASP connection.</td>
</tr>
</tbody>
</table>
### Field Description

**The ASP name may be up to 64 alphanumeric characters in length.**

*Note:* When you save a new ASP, this field becomes read only and may not be edited.

**Description**

*(Required)* A textual description of the ASP. The description may be up to 64 characters in length.

**Destination URL**

*(Required)* Sets the destination URL to which the request should be sent. Specify either the host name of the ASP or the IP address. The value you enter may be up to 124 characters in length.

*Important:* If your destination URL starts with "http:" or "https:“, then you must append a trailing forward-slash, "/", after the host name and port. For example:

- `http://domain_name:port/`
- `http://domain_name:port/mydoc`

Where:

- `domain_name` is the URL domain name or IP address.
- `port` is the port number to use.

**Protocol**

The protocol field contains the protocol that should be used when interacting with the ASP.

*Note:* The available protocols are configured in the `eserv.config` file (see "Mapping parameters" on page 59).

**Connection**

The type of the connection (normal/HTTP or encrypted/HTTPS).

*Note:* This field is only available for the SOAP and XML protocols.

**Authenticate**

Whether to check server authentication.

*Note:* This field is only available for HTTPS connectivity.

For more information about server authentication, see *HTTP and HTTPS Connections* (on page 3).

**PI User**

The PI user for whom PI commands may be triggered by the DAP Send Request feature node. The specified PI user will be automatically logged on when a PI command is first triggered.

*Note:* This field is only available for the PIXML protocol.

This field is populated by the *PI Users* tab in the PI Administration screen. For more information, see *PI User’s Guide*.

**HTTP Auth User**

The user name to authenticate with the ASP server using HTTP basic authentication.

*Note:* This field is only available for the SOAP and XML protocols.

**Max Secondary Connections**

This field contains the maximum number of secondary connections that can be created by the DAP interface.
## ASP configuration

The table in the ASP tab on the Resources screen displays the ASPs that are currently available in the system. Follow these steps to edit or create ASP entries.

For more information about the fields on this screen, see ASP tab fields (on page 20).

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 1    | To create a new ASP:  
     - On the ASP tab, click New.  
To edit an existing ASP:  
     - In the table on the ASP tab, select the ASP to edit and click Edit.  
Result: You see the ASP Configuration screen (See example on page 23). |
| 2    | (Required) In the Name field type the name of the ASP.  
**Note:** When you save a new ASP, this field becomes read only and may not be edited. |
| 3    | (Required) In the Description field, enter a textual description of the ASP. |
| 4    | (Required) In the Destination URL field, enter the destination URL to which the request should be sent.  
**Note:** This may be the host name of the ASP or the IP address. |
| 5    | In the Protocol field select the protocol to use when interacting with the ASP.  
Result: The following fields are enabled, depending on the protocol selected.  
<table>
<thead>
<tr>
<th>Protocols</th>
<th>Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML and SOAP</td>
<td></td>
</tr>
</tbody>
</table>
     - Connection  
     - Do HTTP Authentication?  
| PIXML |  
     - PI User |
| 6    | If the protocol is XML or SOAP, select whether the connection should use HTTP or HTTPS.  
7    | If you selected a HTTPS connection, if you want to make sure it matches the server name in the HTTPS certificate, select the Authenticate Server check box.  
8    | If the protocol is PIXML, select the required PI user from the drop down list in the PI User field.  
9    | Select the maximum number of secondary connections that can be created by the DAP interface in the Max Secondary Connections field.  
10   | If the protocol is XML or SOAP, you can select the Do HTTP Authentication? check box.  
If ticked, enter the:  
     - User name to authenticate with, in the HTTP Auth User field  
     - Password to authenticate with, in the HTTP Auth Password field  
**Note:** The password field shows dots rather than the text of the password. |
| 11   | Click Save.  
Result: The ASP is saved in the database.  
**Note:** When editing an ASP description, the Save button will be disabled if the new description is the same as the original description. |
ASP Configuration screen

Here is an example ASP Configuration screen.

Deleting an ASP

Follow these steps to delete an ASP from the service.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>In the table on the ASP tab, select the ASP to delete.</td>
</tr>
</tbody>
</table>
| 2    | Click Delete.  
**Result:** You see the Delete Confirmation prompt.  
**Note:** An error is raised if the ASP has any operations attached. |
| 3    | Click OK.  
**Result:** The ASP is removed from the database. |

Operations

Introduction

You use the Operations tab on the Resources screen to configure the operations that are available to the DAP application. These operations are used by the Send Request (pre DAP 2.1 versions) or DAP Request (DAP 2.1 onward versions) feature nodes.
Operations tab

The Operations tab contains a view of all the operations created in the service. Here is an example.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>ASP</th>
<th>Timeout</th>
<th>Correlate</th>
<th>Change User</th>
<th>Change Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>GoogleSearchIn</td>
<td>GoogleSearchPort</td>
<td>60000</td>
<td>N</td>
<td>SU</td>
<td>2005-07-24</td>
<td></td>
</tr>
<tr>
<td>WadImportRFOCBl</td>
<td>WadImportTestRFOCBl</td>
<td>60000</td>
<td>N</td>
<td>SU</td>
<td>2005-07-27</td>
<td></td>
</tr>
</tbody>
</table>

Operations tab 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 name of the operation. The operation name may be up to 64 alphanumeric characters in length and is required. Note: Once an operation has been saved this field becomes read only and may not be edited.</td>
</tr>
<tr>
<td>Description</td>
<td>A textual description of the operation. The description may be up to 64 characters in length and is required.</td>
</tr>
<tr>
<td>ASP</td>
<td>The ASP associated with the operation. This field is populated by the records configured on the ASP (on page 20) tab.</td>
</tr>
<tr>
<td>Timeout</td>
<td>The timeout (in milliseconds) that dapIF (on page 64) should wait for a response from the ASP.</td>
</tr>
<tr>
<td>Correlate</td>
<td>Indicates return parameters are expected. See Correlation (on page 24).</td>
</tr>
</tbody>
</table>

Correlation

When the correlation comment <!--CORRELATE--> or <!--CORRELATE-ID--> is specified within the request script, it is implied that return parameters will be sent back by the ASP. The return parameters can be stored within ACS in a user defined profile block and field.

The correlation tag is only supported in asynchronous connections. For more information about asynchronous connections, see Synchronous and asynchronous connections (on page 2).

Adding an operation

The table in the Operations tab displays the operations that are currently available in the system. Follow these steps to create operations.
For more information about the fields on this screen, see *Operations tab fields* (on page 24).

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 1    | To create a new template, on the Operations tab, click New.  
*Result:* You see the New Operation screen (See example on page 26).  
*Note:* The panel at the bottom of the screen lists all the errors that must be fixed before the **Save** button becomes available. |
| 2    | (Required) In the Name field type the name of the template. The template name may be up to 64 alphanumeric characters in length.  
*Note:* Once a template has been saved this field becomes read only and may not be edited. |
| 3    | (Required) In the Description field type a textual description of the template. The description may be up to 64 characters in length. |
| 4    | Select the ASP to associate with the template from the **ASP Name** drop down list. |
| 5    | In the Timeout field optionally change the timeout (in milliseconds) that the DAP interface should wait for a response from the ASP. |
| 6    | If required, select the operation set for this operation from the **Operation Set** drop down list. |
| 7    | If required, type in the SOAP action in the **SOAP Action** field. This permits the SOAP action in the HTTP header field to be overridden.  
*Note:* This field is only available for ASPs using the SOAP protocol. If left empty it will send the destination URL configured for the ASP (by default). |
| 8    | If the ASP is using the SOAP protocol and you want to send the SOAP Header tag, then select the **Send SOAP Header** check box.  
*Note:* The SOAP Header tag sent by DAP is empty, that is, it takes the form "<soapenv:Header/>". Ensure Send SOAP Header is not selected if the ASP is unable to accept an empty SOAP Header tag. |
| 9    | The script text field forms the body of the request template. In the **Request Template section** (See example on page 27), enter a valid script in XML format. For details of script format see **Script Format** (on page 25).  
*Note:* If there are any variables in the script (that is, anything within an element surrounded with "<<" and ">>", and with a "$" prefix, or an empty element), they will be listed in the Request Parameters section with an error status and corresponding message in the Error/Notices panel. |
| 10   | Resolve any request parameter configuration errors. See **Request parameter configuration** (on page 27). |
| 11   | If there are response parameters to add and configure, click **Responses** tab and configure as required. See **Response parameter configuration** (on page 28). |
| 12   | When there are no errors listed in the Error/Notice panel, the **Save** button becomes available.  
Click **Save**. |

**Script format**

The script entered in the script text field of a template forms the body of the request template. The expected format is XML.
A parameter is defined either using the $ prefix or an empty element:

```xml
<voicemail>
  <!--CORRELATE-->
  <msisdn>$1</msisdn>
  <language></Language>
  <date>$time$date</date>
</voicemail>
```

The parameters specified in the example are $1, language, $date and $time.

DAP requests sent through the SLEE will have a limitation of 10 parameters per script. This will be enforced at run time. A parameter can be up to 32 characters in length.

**Note:** The "$" character will not be treated as a parameter within the text of a tag if it is escaped, that is, \\

### New Operation screen - Request

Here is an example New Operation screen, displaying the **Request** tab, before data is entered.
New Operation screen - Request Template

Here is an example New Operation screen, after data is entered in the Request Template.

Request parameter configuration

Follow these steps to configure operation request parameters.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 1    | Click on the parameter to configure.  
      | **Result:** The parameter name appears in the **Name** field to the right; the **Iterator For** panel is populated with all the Request Template elements, and the editable parameter fields are made available. |
| 2    | In the **Description** field type a textual description of the parameter. |
| 3    | Select the parameter feature node behavior from the **Node Disposition** drop down list.  
      | Valid dispositions are:  
      |   - **Hidden** - this parameter cannot be seen in the feature nodes and must be configured in this screen.  
      |   - **Text** - this parameter must be completed in the feature node configuration as |
Step | Action
--- | ---
1 | Select the Response tab.  
**Result:** You see the New Operation screen (See example on page 29), Response tab.
2 | Click Add to to start adding a new parameter.  
**Result:** A new editable field is added to the Response Parameters list.
3 | Click in the empty Name field.
4 | Type the parameter name.
5 | Enter.  
**Result:** The field status is checked and displayed, any errors are listed in the Error/Notice panel, and the parameter configuration fields (See example on page 30) become available.
6 | In the Description field, type what the parameter is to be used for.
7 | In the Search Expression field, type a valid XPath search expression.
8 | If this parameter must be returned, select the Required check box.
9 | If this parameter can be edited in the Feature Nodes, select the Node Editable check box.
10 | If the return of this parameter is an error condition, select the Indicates Error check box.
**Note:** The Required and Indicates Error check boxes are mutually exclusive.
11 | Select the profile type and location from the Profile Field Type, Profile Block and Profile Field drop down lists.  
**Result:** The last error message should disappear and the Save button becomes available.
**Note:** If an expected profile field is missing, it can be added via the ACS Configuration screens. For information about configuring profile fields, see NCC Advanced Control Services User's Guide.
Repeat steps 2 to 11 for all the response parameters that you want to add.

Click **Save**.

**New Operation screen - Response**

Here is an example New Operation screen, displaying the **Response** tab, before data is entered.
New Operation screen - Response Parameters

Here is an example New Operation screen, after data is entered in the Response Parameters.

![New Operation Screen](image)

Editing an operation

The table in the **Operations** tab displays the operations that are currently available in the system. Follow these steps to edit operations.

For more information about the fields on this screen, see *Operations tab fields* (on page 24).

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 1    | To edit an existing template, on the table on the **Operations** tab, select the operation to edit, then click **Edit**.  
**Result:** You see the *Edit Operation screen* (See example on page 32), with the Name grayed out (it is read only) and the **Request** tab displayed. |
<p>| 2    | If required, in the <strong>Description</strong> field, change the description of the operation. The description may be up to 64 characters in length. |
| 3    | If required, change the ASP to associate with the operation from the <strong>ASP</strong> drop down list. |
| 4    | If required, in the <strong>Timeout</strong> field, change the timeout that the <strong>DAP</strong> interface should wait for a response from the ASP. The timeout is measured in milliseconds. |
| 5    | If required, select a different <strong>Operation Set</strong> from the drop down list. |</p>
<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 6    | If required, type/modify the SOAP Action in the **SOAP Action** field.  
**Note:** This field is only available if the ASP is using the SOAP protocol. It permits the SOAP Action in the HTTP Header field to be overridden. If left empty, then the destination URL configured for the ASP will be sent. |
| 7    | If the ASP is using the SOAP protocol then select/deselect the **Send SOAP Header** check box as required.  
**Note:** The SOAP Header tag sent by DAP is empty. Ensure Send SOAP Header is not selected if the ASP is unable to accept an empty SOAP Header tag. |
| 8    | If required, change the **Request Template** script and (re)configure parameter changes as indicated by the **Error/Notices** panel. See *Request parameter configuration* (on page 27). |
| 9    | Select the **Response** tab.  
**Result:** You see the Edit Operation screen, **Response** tab, with the Name grayed out (it is read only). |
| 10   | To add a new response parameter, see *Response parameter configuration* (on page 28).  
To edit a response parameter, see *Response parameter configuration* (on page 28), steps 6 onwards.  
To delete a response parameter:  
Select the parameter  
Click **Delete**.  
**Tip:** If a parameter is deleted in error:  
- add a new parameter  
- type the same name as per deleted parameter  
- click **Enter** |
| 11   | When there are no errors listed in the Error/Notice panel, the **Save** button becomes available.  
Click **Save**. |
Edit Operation screen - Request Template

Here is an example Edit Operation screen, with the Request tab displayed.

Finding a DAP template

Follow these steps to search for a DAP operation.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 1    | On the Operations tab, click Find.  
**Result:** You see the Find Operations screen (See example on page 33). |
| 2    | Enter the search criteria in one of the query fields using the drop down list and click Find.  
**Result:** When you click Find a query is triggered and the first 100 records matching the value in the query field will be returned. The results appear in the table on the Operations tab.  
**Note:** If you select a value in more than one query field then the Find button will be disabled. |
Find Operations screen

Here is an example Find Operations screen.

<table>
<thead>
<tr>
<th>ASP Name</th>
<th>ACS Parameter</th>
<th>Request/Response Parameter</th>
</tr>
</thead>
</table>

Click Find.

Result: You see the Copy DAP Operation screen (See example on page 33).

Copy DAP templates

Follow these steps to copy a defined DAP operation.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>On the Operations tab, select the operation to copy.</td>
</tr>
</tbody>
</table>
| 2    | Click Copy.  
      | Result: You see the Copy DAP Operation screen (See example on page 33). |
| 3    | In the Name field, type a unique name for the new operation. |
| 4    | Click Save.  
      | Result: The operation will be saved to the database under the new name.  
      | Note: This operation will have exactly the same details as the original operation. See Editing an operation (on page 30) to change the operation details. |

Copy DAP Operation screen

Here is an example Copy DAP Operation screen.

Deleting an operation

Follow these steps to delete an Operation from the service.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>In the table on the Operations tab, select the operation to delete.</td>
</tr>
</tbody>
</table>
| 2    | Click Delete.  
      | Result: You see the Delete Confirmation prompt.  
      | Note: An error is raised if the operation exists in an active control plan. |
Step | Action
--- | ---
3 | Click OK.  
Result: The operation is removed from the database.

**Operation Sets**

**Introduction**

Operation sets are used to limit the operations to selected users.

**Operation Sets tab**

Here is an example **Operation Sets** tab.

![Operation Sets Tab Example](image)

**Add or edit an operation set**

The table in the **Resources** screen **Operation Sets** tab displays the operation sets that are currently available in the system.

Follow these steps to add a new operation set or edit an existing set.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 1    | On the **Operation Sets** tab:  
- To create a new set, click **New**  
- To edit an existing set, select the set, then click **Edit**  
Result: You see the **Operation Set Configuration screen** (See example on page 35). |
| 2    | In the **Name** field, enter or edit the name of the operation set. |
| 3    | Select the check box for all the listed users you want in this operation set. |
| 4    | Click **Save**. |
Operation Set Configuration screen

Here is an example Operation Set Configuration screen.

Deleting an operation set

Follow these steps to delete an operation set.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>In the table on the <strong>Operation Sets</strong> tab, select the operation set to delete.</td>
</tr>
</tbody>
</table>
| 2    | Click **Delete**.  
   **Result:** You see the Delete Confirmation prompt. |
| 3    | Click **OK**.  
   **Result:** The operation set is removed from the database. |
Chapter 3

Import WSDL

Overview

Introduction

This chapter explains how to use the DAP Import WSDL screen.

In this chapter

This chapter contains the following topics.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Import WSDL Screen</td>
<td>37</td>
</tr>
<tr>
<td>Import WSDL Files</td>
<td>38</td>
</tr>
<tr>
<td>Operation Request Configuration</td>
<td>40</td>
</tr>
<tr>
<td>Operation Response Configuration</td>
<td>41</td>
</tr>
</tbody>
</table>

Import WSDL Screen

Introduction

The Oracle Communications Network Charging and Control Data Access Pack DAP Import WSDL screen allows you to import and configure predefined web services from ASPs.

It contains these functions:

- Import WSDL files (on page 38)
- ASP Configuration (on page 20)
- Operation Request Configuration (on page 40)
- Operation Response Configuration (on page 41)

What is WSDL?

Web Services Description Language (WSDL) is a XML based language that provides a model for describing web services.

The reason for using WSDL is to import predefined web services from ASPs, thereby speeding up configuration of DAP messages.

For restrictions on what parts of the specification are supported, see NCC Data Access Pack PICS Guide.

Accessing the Import WSDL screen

Follow these steps to open the DAP Import WSDL screen.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Select the Services menu from the SMS main screen.</td>
</tr>
<tr>
<td>2</td>
<td>Select DAP.</td>
</tr>
<tr>
<td>3</td>
<td>Select Import WSDL.</td>
</tr>
</tbody>
</table>
### Import WSDL Files

**Importing a WSDL file**

Follow these steps to import a WSDL file.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 1    | On the *Import WSDL screen* (See example on page 39), perform one of the following actions to select a file:  
  - Enter the file location and name in the *File name field*  
  - Click *Browse* to find the file  
  The file has the suffix of `.wsdl` (for example *DAP.wsdl*)  
  **Result:** The *Import* button becomes available. |
| 2    | Click *Import*.  
  **Result:** The WSDL file is imported, then:  
  - The *Operation/Description* panel is populated from the imported file  
  - The *No operation has been added* message is displayed in the Notice/Error box. |
| 3    | To add:  
  - *All operations*, select the *Add* check box  
  - *Selected operations*, select the check box for each of the required operations  
  **Result:**  
  - Any configuration requirements are displayed in the Notice/Error box.  
  - The configuration fields under the *ASP*, *Request*, and *Response* tabs are populated with data from the WSDL and become available for editing.  
  - The *Operation Set* field becomes available.  
  **Warning:** If you select multiple operations, you must configure all of them before the *Save* button is available. If there are any kind of fatal system problems, you will need to re-do any unsaved configuration. |
<p>| 4    | Select the <em>Operation Set</em> to use from the drop down list. |</p>
<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Click on the operation to configure.</td>
</tr>
<tr>
<td>6</td>
<td>Configure the ASP as required. See ASP (on page 20) for field descriptions.</td>
</tr>
<tr>
<td>7</td>
<td>Configure the request as required. See Operation Request Configuration (on page 40) for details.</td>
</tr>
<tr>
<td>8</td>
<td>Configure the response as required. See Operation Response Configuration (on page 41) for details.</td>
</tr>
<tr>
<td>9</td>
<td>Repeat steps 4 to 7 for each of the operations to configure.</td>
</tr>
<tr>
<td>10</td>
<td>When there are no errors listed in the Error/Notice panel, the Save button becomes available. Click Save.</td>
</tr>
</tbody>
</table>

**Note:** Once saved, the ASP and the operation is created. You can view and edit them on the DAP Resources screen (on page 19).

### Import WSDL screen

Here is an example Import WSDL screen.
Operation Request Configuration

Introduction

You use the Request tab on the Import WSDL screen to configure the request parameters that are within the imported operation script. The imported operations are used by the DAP Request feature node.

Request tab

Here is an example of the Request tab on the Import WSDL screen after a WSDL file has been imported.

Configuring requests

Follow these steps to configure operation request parameters.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>In the Request tab (See example on page 40), expand the parameter tree to see all the elements that can be configured (click + signs).&lt;br&gt;&lt;br&gt;Note: All the elements will have a C, P or O preceding them. The meanings are:&lt;br&gt;&lt;ul&gt;&lt;li&gt;C = a Complex parameter that has more than one sub parameters.&lt;/li&gt;&lt;li&gt;P = the last element down a branch. This is the Parameter that can be configured.&lt;/li&gt;&lt;li&gt;O = an Optional parameter that may or may not be configured.&lt;/li&gt;&lt;/ul&gt;</td>
</tr>
<tr>
<td>2</td>
<td>Select the parameters to configure (select the check box preceding the parameter).&lt;br&gt;&lt;br&gt;Result: All the selected parameter names are shown as red to indicate that the parameter has not yet been configured properly. The Error/Notices section will show</td>
</tr>
</tbody>
</table>
Chapter 3

Step | Action
---|---
the reason.

**Tip:** Hover over a parameter to get a tip if a box cannot be selected or deselected.

**Note:** Sometimes all parameters under a selected complex parameter are automatically selected because they are all required.

3 Click a parameter name to configure.

**Result:** The parameter name appears in the Name field to the right and the **Iterator For** panel is populated with all the Request elements, and the editable parameter fields are made available.

4 If this parameter is to be used as a unique id, select the **Correlation ID** check box and then start configuring a new parameter.

5 In the **Description** field type a textual description of the parameter.

6 Select the parameter feature node behavior from the **Node Disposition** drop down list.

Valid dispositions are:

- **Hidden** - this parameter cannot be seen in the feature nodes and must be configured in this screen.
- **Text** - this parameter must be completed in the feature node configuration as a text field.
- **Profile Block** - this parameter must be completed in the feature node configuration as a profile block location.
- **Either** - this parameter must be completed in the feature node configuration optionally as a text field or profile block location.

7 If **Hidden** is the disposition, complete the configuration as appropriate.

- Type the **Default Value**, and, or
- Select the required **Profile Field Type**, **Profile Block** and **Profile Field** from their drop down lists.

**Tip:** The **Xsd Type** field indicates what type of data is expected in the DAP message for this parameter.

**Note:** If an expected profile field is missing, it can be added via the ACS Configuration screens. For information about configuring profile fields, see *NCC Advanced Control Services User’s Guide*.

8 If this parameter is being used as an iteration value for a Request element, scroll to find the element in the **Iterator For** list and select the **Add** check box.

**Result:** This parameter is added to the Request template element as an iteration.

**Tip:** A parameter can be an iterator for more than one element, however one element registers just one parameter as the iterator.

9 Repeat steps 3 to 8 for all the parameters that require configuration.

**Operation Response Configuration**

**Introduction**

You need to configure any responses required by the imported script. You perform this on the **Response** tab.
Response tab

Here is an example of the Response tab on the Import WSDL screen after a WSDL file has been imported.

Configuring responses

Follow these steps to configure all response parameters.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 1    | In the Response tab (See example on page 42), expand the parameter tree to see all the elements that can be configured (click + signs).  
**Note:** All the elements will have a C, P or O preceding them. The meanings are:  
- C = a Complex parameter that has more than one sub parameters.  
- P = the last element down a branch. This is the Parameter that can be configured.  
- O = an Optional parameter that may or may not be configured. |
| 2    | Select the parameters to configure (select the check box preceding the parameter).  
**Result:** All the selected parameter names are shown as red to indicate that the parameter has not yet been configured. |
| 3    | Click a parameter name to configure.  
**Result:** The parameter name appears in the Name field to the right and the editable parameter fields are made available. |
<p>| 4    | In the Description field, type what the response is expected to be. |
| 5    | If this parameter must be returned, select the Required check box. |</p>
<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>If this parameter can be edited in the Feature Nodes, select the <strong>Node Editable</strong> check box.</td>
</tr>
</tbody>
</table>
| 7    | If the return of this parameter is an error condition, select the **Indicates Error** check box.  
**Note:** The Required and Indicates Error check boxes are mutually exclusive. |
| 8    | Select the profile type and location from the **Profile Field Type**, **Profile Block** and **Profile Field** drop down lists.  
**Note:** If an expected profile field is missing, it can be added via the ACS Configuration screens. For information about configuring profile fields, see *NCC Advanced Control Services User's Guide*.  
**Result:** The last error message for this parameter will disappear from the Error/Notices panel. |
| 9    | Repeat steps 3 to 9 to configure all the response parameters. |
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>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration Overview</td>
<td>45</td>
</tr>
<tr>
<td>eserv.config Configuration</td>
<td>46</td>
</tr>
<tr>
<td>DAP eserv.config configuration</td>
<td>48</td>
</tr>
<tr>
<td>SLEE.cfg Configuration</td>
<td>61</td>
</tr>
<tr>
<td>Configuration for Optimal Performance</td>
<td>61</td>
</tr>
</tbody>
</table>

Configuration Overview

Introduction

This topic provides a high level overview of how the Oracle Communications Network Charging and Control Data Access Pack DAP component is configured.

Configuration components

DAP is configured by the following components:

<table>
<thead>
<tr>
<th>Component</th>
<th>Locations</th>
<th>Description</th>
<th>Further Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>eserv.config</td>
<td>All SLC machines</td>
<td>The DAP is configured in the DAP section of the eserv.config file.</td>
<td>eserv.config Configuration (on page 46)</td>
</tr>
<tr>
<td>SLEE.cfg</td>
<td>All SLC machines</td>
<td>This configures how the SLEE runs and manages dapIF (on page 64). SLEE configuration must include the DAP.</td>
<td>Startup (on page 64) SLEE Technical Guide</td>
</tr>
<tr>
<td>acs.conf</td>
<td>All SLC machines</td>
<td>The ACS framework must be configured to accept DAP calls.</td>
<td>ACS Technical Guide</td>
</tr>
<tr>
<td>SMF database</td>
<td>SMS</td>
<td>Statistics, profile block and EFM alarms configuration for DAP. Configured automatically when lcaSms is installed.</td>
<td>Statistics (on page 15)</td>
</tr>
<tr>
<td>DAP screens</td>
<td>SMS</td>
<td>The Resources screen configures ASP and DAP template records in SMF</td>
<td>Resources Screen (see &quot;Resources&quot; on page 19)</td>
</tr>
<tr>
<td>ACS control plans</td>
<td>SMS</td>
<td>Specific control plans must be configured.</td>
<td>CPE User's Guide</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:

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

or

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

or

```plaintext
{ 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.

Example eserv.config

The following is an example of DAP configuration in `eserv.config` on SLC.

```plaintext
DAP = {
    Mapping = [
        # XML protocol
        {
            Protocol = "H"
            InterfaceHandle = "dapIF"
        },
        # SOAP protocol
        {
            Protocol = "S"
            InterfaceHandle = "dapIF"
        },
        # HPSA protocol
        {
            Protocol = "A"
            InterfaceHandle = "dapIF"
        },
        # PIXML protocol
        {
            Protocol = "P"
            InterfaceHandle = "dapIF"
        },
    ],
    concatenate = false
    #listenHost = ""
    #listenPort = 4099
    connectionTimeout = 0
    correlationTagName = "CORRELATE"
    #responseTagName = "CORRELATE"
    timestampTagName = "TIMESTAMP"
    PollInterval = 500
    cacheAgeSeconds = 60
```
# maxRetries=0
retryTimeout=30

pendingFilename = "/IN/service_packages/DAP/tmp/pendingRequests.txt"
PollCount = 5000
pendingQueueInMemory = true
disableMaxSecCheck = false
maxQueueCheckTimeout = 5
maxQueueLength = 500
nonBlockingConnections = true
persistentConnections = true
enableRetries = true
certificatePath = "/IN/service_packages/DAP/certificates"
certificatesName = "CAfile.pem"
openSSLPath = "/IN/service_packages/DAP/bin"

DAP eserv.config configuration

Introduction

The DAP eserv.config file example is installed by dapSms, dapScp and dapExtras in
/IN/services_packages/DAP/etc.

The eserv.config file and DAP section is required on all nodes running a DAP client capable of sending
DAP requests.

Example eserv.config DAP section

Here is an example of the DAP section of the eserv.config.

DAP = {
    Mapping = [
        {
            Protocol = "H"
            InterfaceHandle = "dapIF"
        },
        {
            Protocol = "S"
            InterfaceHandle = "dapIF"
        },
        {
            Protocol = "A"
            InterfaceHandle = "dapIF"
        },
        {
            Protocol = "P"
            InterfaceHandle = "dapIF"
        }
    ],
    templates = [
        {
            id = 83
            prefix = "xmlData=
        }
    ]
}
soapAction="PaymentService#OTPPayment"
sendHeaderTag=false
}
]
concatenate=false
listenHost=""
listenPort=4099
connectionTimeout=0
correlationTagName="CORRELATE"
uncorrelatedRequestDir="/IN/service_packages/DAP/tmp/"
responseTagName="CORRELATE"
hostnameInPost=true
timestampTagName="TIMESTAMP"
PollInterval=500
PollCount=100

maxRetries=0
retryTimeout=10
sessionTimeout=86400

pendingFilename="/IN/service_packages/DAP/tmp/pendingRequests.db"
pendingQueueInMemory=false

nonBlockingConnections=true
persistentConnections=false
persistentConnectionCheckTimeout=0

maxQueueLength=500
maxQueueCheckTimeout=100
enableRetries=true
certificatePath="/IN/service_packages/DAP/certificates/"
certificatesName="CAfile.pem"
openSSLPath="/IN/service_packages/DAP/bin"

sendRequestDateFormat="%Y-%m-%d"
sendRequestDateTZ="US/Eastern"

prefixTagName="CUSTOM_PREFIX"
suffixTagName="CUSTOM_SUFFIX"

connectionFailureRetryTime=10

soapHeaderOverride="<?xml version="1.0" encoding="ISO-8859-1"
standalone="no">
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
<soapenv:Header/>
useTemplateSOAPTags=false

}
DAP parameters

DAP accepts the following `eserv.config` parameters.

**appendCRLFAfterBody**

*Syntax:* `appendCRLFAfterBody = true|false`

*Description:* Set to true to append "\r\n" (carriage return, line feed) after sending the message body.

*Type:* Boolean

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

*Allowed:* true or false

*Default:* true

*Example:* `appendCRLFAfterBody = true`

**cacheAgeSeconds**

*Syntax:* `cacheAgeSeconds = seconds`

*Description:* The number of seconds before the template cache expires.

*Type:* Integer

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

*Allowed:*

*Default:* 60

*Example:* `cacheAgeSeconds = 60`

**certificatePath**

*Syntax:* `certificatePath = "dir"

*Description:* The location of the:

- Servers’ public certificates
- Concatenated certificates file produced by `dapReadyCertificates.sh` (on page 69)

*Type:* String

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

*Allowed:*

*Default:* "/IN/service_packages/DAP/certificates"

*Notes:* The file of concatenated certificates is defined by `certificatesName` (on page 50). Any file in this directory which does not have a filename matching `certificatesName` will be concatenated into the file specified by `certificatesName`. For an overview of how certificates are handled, see Certificate management (on page 11).

*Example:* `certificatePath = "/IN/service_packages/DAP/certificates"

**certificatesName**

*Syntax:* `certificatesName = "file"

*Description:* The filename of the concatenated servers' public certificates produced by `dapReadyCertificates.sh` (on page 69).

*Type:* String

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

*Allowed:*

*Example:* `certificatesName = "file""
Default: "CAfile.pem"
Notes: The directory this file is in is configured in the certificatePath (on page 50).
For an overview of how certificates are handled, see Certificate management (on page 11).
Example: certificatesName = "CAfile.pem"

concatenate
Syntax: concatenate = true|false
Description: Whether to concatenate the body of the XML requests.
Type: Boolean
Optionality: Optional (default used if not set).
Allowed: true, false
Default: false
Notes: Example: concatenate = false

connectionFailureRetryTime
Syntax: connectionFailureRetryTime = seconds
Description: How long to wait in seconds between connection attempts after a failed connection attempt.
Type: Integer
Optionality: Optional (default used if missing)
Allowed: Default: 10
Notes: Example: connectionFailureRetryTime = 20

connectionTimeout
Syntax: connectionTimeout = mseconds
Description: The number of milliseconds before a connection to an ASP times out.
Type: Integer
Optionality: Optional (default used if not set).
Allowed:
Default: 0 (wait indefinitely)
Notes: Can be overridden by the timeout for correlation objects.
Example: connectionTimeout = 0

correlationTagName
Syntax: correlationTagName = "name"
Description: The correlation tag in the XML messages.
Type: String
Optionality: Optional (default used if not set)
Allowed:
Default: "CORRELATE"
Notes: For more information about correlation, see Correlation (on page 24).
Example: \texttt{correlationTagName = "CORRELATE"}

\textbf{enableRetries}

Syntax: \texttt{enableRetries = true|false}

Description: Sets if DAP will attempt to resend failed requests

Type: Boolean

Optionality: Optional (default used if not set).

Allowed: true, false

Default: true

Notes:

Example: \texttt{enableRetries = true}

\textbf{hostnameInPost}

Syntax: \texttt{hostnameInPost = true|false}

Description: Include the full or shortened version of destination in the POST command in requests.

Type: Boolean

Optionality: Optional (default used if not set).

Allowed: true, false

Default: true

Notes:

Example: \texttt{hostnameInPost = true}

\textbf{listenHost}

Syntax: \texttt{listenHost = "host"}

Description: Specifies the host name to put in the outgoing clientUrl HTTP header.

Type: String

Optionality: \texttt{listenHost} and \texttt{listenPort} are both required if the ASP returns parameters in a separate request.

Allowed:

Default: "" (none)

Notes:

Example: \texttt{listenHost = ""}

\textbf{listenPort}

Syntax: \texttt{listenPort = port}

Description: Specifies the port to put in the outgoing clientUrl HTTP header.

Type: Integer

Optionality: \texttt{listenHost} and \texttt{listenPort} are both required if the ASP returns parameters in a separate request.

Allowed: 4099

Default: 4099

Notes: This allows for the situation when the ASP initiates requests to the XML Interface.

For more information about message flows, see \textit{Message flow} (on page 3).

Example: \texttt{listenPort = 4099}
Mapping

Mapping between protocols and SLEE interface handles. Refer to *Mapping parameters* (on page 59).

**maxQueueCheckTimeout**

**Syntax:**

```
maxQueueCheckTimeout = seconds
```

**Description:**

The number of seconds between each diagnostic check of the pending queue size for each ASP.

**Type:**

Integer

**Optionality:**

Optional (default used if not set).

**Allowed:**

A valid number

**Default:**

100

**Notes:**

A warning will be logged if an ASP queue size is larger than the value specified for `maxQueueLength`.

**Example:**

```
maxQueueCheckTimeout = 100
```

**maxQueueLength**

**Syntax:**

```
maxQueueLength = size
```

**Description:**

The maximum size for the message queue. When set to greater than zero, any new requests to an ASP will be rejected if the request queue size for the ASP exceeds the defined value.

**Type:**

Integer

**Optionality:**

Optional (default used if not set).

**Allowed:**

A valid number

**Default:**

500

**Notes:**

A warning will be logged if an ASP queue size is larger than `maxQueueLength`.

**Example:**

```
maxQueueLength = 500
```

**maxRetries**

**Syntax:**

```
maxRetries = number
```

**Description:**

If a COMMAND_ACK is not received following a request, `maxRetries` defines the number of subsequent requests to attempt before expiring the request.

**Type:**

Integer

**Optionality:**

Optional (default used if not set)

**Allowed:**

0

**Default:**

0

**Notes:**

This parameter applies only to HPSA.

**Example:**

```
maxRetries = 0
```

**nonBlockingConnections**

**Syntax:**

```
onBlockingConnections = true|false
```

**Description:**

Specifies whether or not to use non-blocking sockets for connections to ASPs.

**Type:**

Boolean

**Optionality:**

Optional (default used if not set)

**Allowed:**

true, false

**Default:**

true
Example:  
nonBlockingConnections = false

openSSLPath
Syntax:  
openSSLPath = "path"
Description:  
The location of the binaries to use for encrypting the file specified by 
certificatesName (on page 50).
Type:  
String
Optionality:  
Optional (default used if not set).
Allowed:
Default:  
"/IN/service_packages/DAP/bin"
Notes:  
This should usually be set to the location of openssl (on page 67) and c_rehash 
(on page 63).
For an overview of how certificates are handled, see Certificate management (on 
page 11).
Example:  
openSSLPath = "/IN/service_packages/DAP/bin"

pendingFilename
Syntax:  
pendingFilename = "path/file"
Description:  
Location in which to store pending requests in the event that redelivery is required.
Type:  
String
Optionality:  
Optional (default used if not set).
Allowed:
Default:  
"/IN/service_packages/DAP/tmp/pendingRequests.db"
Notes:  
The pendingRequests file contains one line per message in the format:
<msg_id>::<asp_url>::<port>::<protocol>::<timeout>::<xml>::[return_ 
parameter_name::]+
Example:  
pendingFilename = "/IN/service_packages/DAP/tmp/pendingRequests.db"

pendingQueueInMemory
Syntax:  
pendingQueueInMemory = true|false
Description:  
Sets whether or not to boost performance by holding the queue of pending DAP 
requests in non-persistent memory.
Type:  
Boolean
Optionality:  
Optional (default used if not set).
Allowed:  
true, false
Default:  
false
Notes:  
When set to true, then no persistent records are stored in the location defined in 
the pendingFilename parameter.
Example:  
pendingQueueInMemory = true

persistentConnections
Syntax:  
persistentConnections = true|false
Description:  
Specifies whether the primary connection to an ASP will remain open, or persist, 
between requests.
Type:  
Boolean
Optionality:  
Optional (default used if not set).
Allowed: true, false
Default: false
Notes: When set to true, secondary connections will persist.
Example: persistentConnections = true

persistentConnectionCheckTimeout
Syntax: persistentConnectionCheckTimeout = seconds
Description: Specify whether to check persistent socket connection for closure before sending new request on it.
Value is number of seconds connection has been idle before check is to be performed. Zero value means no checking.
Type: Integer
Optionality: Optional (default used if not set).
Allowed: Default: 0
Notes: Example: persistentConnectionCheckTimeout = 0

PollCount
Syntax: PollCount = number
Description: Defines the number of zero wait polls to perform during idle periods, after which the PollInterval (on page 55) timeout is re-applied.
Type: Integer
Optionality: Optional (default used if not set).
Allowed: A valid number
Default: 5000
Notes: Enables tight polling in periods of high traffic
Example: PollCount = 2000

PollInterval
Syntax: PollInterval = milliseconds
Description: Number of milliseconds that dapIF (on page 64) will sleep before processing SLEE events.
Type: Optional (default used if not set)
Allowed: Default: 500
Notes: Example: PollInterval = 500

prefixTagName
Syntax: prefixTagName = "name"
Description: Identifies the tag encapsulating the custom prefix text.
Type: String
Optionality: Optional
Chapter 4

Allowed:
Default: Not set
Notes: The tag itself is discarded before sending.
Example: prefixTagName = "CUSTOM_PREFIX"

responseTagName
Syntax: responseTagName = "name"
Description: The name of the expected correlation tag in the XML response.
Type: String
Optionality: Optional (default used if not set).
Allowed: A valid correlation tag name.
Default: correlationTagName (on page 51) value
Notes: If not set, then the value specified for the correlationTagName parameter will be used.
Example: responseTagName = "CORRELATE"

retryTimeout
Syntax: retryTimeout = seconds
Description: After a request is issued, this defines the time to wait for a COMMAND_ACK
before it attempting a new request.
Type: Integer
Optionality: Optional (default used if not set)
Allowed: 10
Default: 10
Notes: This parameter applies only to HPSA.
Example: retryTimeout = 10

sendRequestDateFormat
Syntax: sendRequestDateFormat = "dateformat"
Description: Indicates the date format to use for the System date (formatted) option while
setting variables.
Type: Date
Optionality: Optional (default used if not set).
Allowed: Default: "%Y-%m-%d"
Notes:
Example: sendRequestDateFormat = "%Y-%m-%d"

sendRequestDateTZ
Syntax: sendRequestDateTZ = "name"
Description: Alternative timezone abbreviation when using the 'System date (formatted)'
("<fdt>") option. This calculates the <fdt> date from the specified timezone.
Type: String
Optionality: Optional (default used if not set).
Allowed: Default: GMT, NCC default TZ for SLC; otherwise the default system TZ
Notes:
Example:  

sendRequestDateTZ = "US/Eastern"

sessionTimeout

Syntax:  
sessionTimeout = seconds

Description:  
The number of seconds to negotiate an open SSL session with a remote server before it is reset by the server

Type:  
Integer

Optionality:  
Optional (default used if not set).

Allowed:  

Default:  
86400 (24 hours)

Notes:  
Example:  

sessionTimeout = 86400

soapHeaderOverride

Syntax:  

Description:  
Override the standard XML and SOAP header tags.

Type:  
String

Optionality:  
Optional (default used if not set).

Allowed:  

Default:  
Not used

Notes:  
Applies to SOAP only.

Note this does not affect the close tags </soapenv:Body></soapenv:Envelope>, which will still be added by DAP.

Example:  


suffixTagName

Syntax:  
suffixTagName = "name"

Description:  
Identifies the tag encapsulating the custom suffix text.

Type:  
String

Optionality:  
Optional

Allowed:  

Default:  
Not set

Notes:  
The tag itself is discarded before sending.

Example:  
suffixTagName = "CUSTOM_SUFFIX"
templates
Syntax: templates = [parameters]
Description: Allows overriding of certain values on a per template basis. See templates parameters (on page 59).
Type: Array
Optionality: 
Allowed: 
Default: 
Notes: Not used by default.
Example: templates = [
    {
        id = 83
        prefix = "xmlData=
        soapAction="PaymentService#OPPPayment"
        sendHeaderTag=false
    }
]

timestampTagName
Syntax: timestampTagName = "tag"
Description: Expected time stamp tag in the XML message.
Type: String
Optionality: Optional (default used if not set)
Allowed: 
Default: "TIMESTAMP"
Notes: 
Example: timestampTagName = "TIMESTAMP"

uncorrelatedRequestDir
Syntax: uncorrelatedRequestDir = "path"
Description: The ASPManager process writes to an uncorrelated log file at this path if it receives a correlated response for which it cannot find a matching correlated entry.
Type: String
Optionality: Optional (default used if not set).
Allowed: 
Default: 
Notes: 
Example: uncorrelatedRequestDir = "/IN/service_packages/DAP/tmp/

useTemplateSOAPTags
Syntax: useTemplateSOAPTags = true|false
Description: Do not include any SOAP header tags in dapIF. Only use those from the template.
Type: Boolean
Optionality: Optional (default used if not set).
Allowed: true, false
Default: false
Notes: The close tags (</soapenv:Body></soapenv:Envelope>) will be added if not already present at the end of the template.

Example: useTemplateSOAPTags = false

Mapping parameters

The following parameters are valid for the Mapping section of the DAP configuration.

InterfaceHandle

Syntax: InterfaceHandle = "Dap_Interface"

Description: Specifies which SLE Interface to connect to.

Type: String

Optionality: Optional (default used if not set).

Allowed: dapIF single SLE Interface handle for the DAP Interface

[*dapIF_1*, *dapIF_2*,...] list of multiple SLE handles

Default:"

Notes: This must be the SLE Interface for the DAP Interface as defined in the SLE config. Multiple SLE handles can be defined in a list and requests will be load balanced between them.

Example: InterfaceHandle = "dapIF"

Protocol

Syntax: Protocol = "H|S|A|P|L"

Description: The protocol to be used between the SLE Interface and the external ASP.

Type: String

Optionality: Allowed: H XML S SOAP A HPSA P PIXML L LDAP

Default:"

Notes: For XML and SOAP, the ASP configuration defines whether it is over HTTP or HTTPS. For more information about this configuration, see ASP tab fields (on page 20). LDAP is only available if LDAP interface for DAP has been installed and configured. Refer to LDAP Interface for DAP Technical Guide for details.

Example: Protocol = "H"

templates parameters

The following parameters are valid for the templates section of the DAP configuration.
id

Syntax: \( id = id \)
Description: ID of the template ID to act on.
Type: Integer
Optionality: Optional
Allowed: 
Default: 
Notes: 
Example: \( id = 83 \)

prefix

Syntax: \( prefix = "pref" \)
Description: The prefix to add in front of XML or SOAP
Type: String
Optionality: Optional
Allowed: 
Default: not added
Notes: 
Example: \( prefix = "xmlData=\)"

sendHeaderTag

Syntax: \( sendHeaderTag=true|false \)
Description: Send the soapenv:Header tag
Type: Boolean
Optionality: Optional (default used if not set).
Allowed: true, false
Default: true
Notes: Only applicable for SOAP
Example: \( sendHeaderTag=false \)

soapAction

Syntax: \( soapAction="action" \)
Description: SOAP action to specify in the header
Type: String
Optionality: Optional
Allowed: 
Default: If not specified uses destination URL
Notes: Only applicable for SOAP
Example: \( soapAction="PaymentService#OTPPayment" \)
SLEE.cfg Configuration

Introduction

The SLEE.cfg file must be configured to enable the DAP to work. Because all necessary SLEE configuration is done at installation time by the configuration script, this section is for information only.

The SLEE configuration file is located at /IN/service Packs/SLEE/etc/SLEE.cfg.

Refer to SLEE Technical Guide for details on SLEE configuration.

DAP SLEE configuration

During installation, the following line will be added to the SLEE.cfg file:

```
INTERFACE=dapIF dapIF.sh /IN/service Packs/DAP/bin EVENT
```

Usage:

```
INTERFACE=uniqueIdentifier interfaceName interfacePath interfaceType [eventCount dialogCount]
```

Note: The use of the SLEE Interface type UDG has been deprecated.

Larger SLEE events

DAP requires that a pool of SLEE events of at least 3072 bytes be configured, for example:

```
MAXEVENTS=count 3072
```

(where count is the pool size)

For most efficient use of shared memory, a pool of SLEE events of 1024 bytes should also be configured, for example:

```
MAXEVENTS=count 1024
```

Configuration for Optimal Performance

Introduction

You can configure the DAP to optimize its performance. The optimal configuration settings will depend on the capabilities of the ASP.

HTTP version 1.1

If the ASP supports HTTP version 1.1 then the following configuration is required to optimize DAP performance. Set the DAP parameters:

- nonBlockingConnections = false
- persistentConnections = true

HTTP version 1.0

If the ASP supports HTTP version 1.0 then the following configuration is required to optimize DAP performance. Set the DAP parameter:

- persistentConnections = false
Multiple instances

You can increase the speed of traffic through the DAP by using multiple instances of the DAP interface. The speed will increase by the maximum speed of a single DAP interface multiplied by the number of instances.

Multiple instances of the DAP interface can be started by adding more interface definitions to the DAP SLEE configuration. See SLEE.cfg Configuration (on page 61) for details.

Note: If you add more interfaces to SLEE.cfg, then you must configure the list of interface handles in the DAP section of eserv.config. See Mapping parameters (on page 59) for details.

General

You can increase the volume of traffic through the DAP interface by increasing the maximum secondary connections allowed to the ASP. For details, see Configuring an ASP (see "ASP configuration" on page 22).

DAP performance can also be improved by specifying the following configuration. Set the DAP parameters:

- PollInterval = 10
- PollCount = 5000
- pendingQueueInMemory = true

For more information on configuring DAP parameters refer to DAP eserv.config configuration (on page 48).

Warning: If pendingQueueInMemory is set to true then the pending queue will not be stored in persistent storage and therefore cannot be recovered following a failure and restart of the DAP process.
Overview

Introduction

This chapter explains the processes which run automatically as part of the application. These processes are started automatically by one of the following:

- inittab
- crontab
- Service Logic Execution Environment SLEE

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

In this chapter

This chapter contains the following topics.

- c_rehash 63
- dapIF 64
- dapMacroNodes 65
- dapTypeConversion 65
- libdapChassisActions 66
- libDAPManager.so 66
- openssl 67
- sqlite3 67

**c_rehash**

Purpose

c_rehash is used by openssl (on page 67) to encrypt the concatenated server certificates into the file specified by certificatesName (on page 50).

Location

This binary is located on both SLCs and VWSs.

Startup

c_rehash is used by openssl if it is in the directory specified by the openSSLPath (on page 54) parameter.

Configuration

c_rehash has some configuration which is set when used by openssl (on page 67). To configured this process, use the configuration available to openssl.
# dapIF

## Purpose

dapIF is a SLEE interface. It is the main Oracle Communications Network Charging and Control Data Access Pack DAP client that sends and receives XML requests to external ASPs. It listens for SLEE requests and messages from ASPs.

It can trigger a PI command from a control plan using the DAP Send Request feature node when communicating with an ASP using the PIXML protocol.

## Location

This binary is located on both SLCs and VWSs.

## Startup

The interface is started by the SLEE, through the `/IN/service_packages/DAP/bin/dapIF.sh` shell script.

## Configuration

dapIF is configured in the DAP section of `eserv.config` and the DAP Resources screen.

For more information about the:

- `eserv.config` parameters, see *DAP eserv.config configuration* (on page 48)
- DAP Resources screen, see *Resources Screen* (see "Resources" on page 19)

## Command line parameters

dapIF accepts the following command-line parameters at start up.

```
dapIF [-u usr/pwd|--user usr/pwd]
```

**Note:** Either the `-u` or the `--user` option can be used.

### `-u usr/pwd`

**Syntax:**

```
-u usr/pwd
```

**Description:**
The userid and password combination to use to log into the local Oracle instance.

**Type:**

String

**Optionality:**

Optional (default used if not set).

**Allowed:**

Default: `/`

**Notes:**

- Cannot be used with the `--user /usr/pwd` (on page 64) option.

**Example:**

```
-u smf
```

### `--user /usr/pwd`

**Syntax:**

```
--user /usr/pwd
```

**Description:**
The userid and password combination to use to log into the local Oracle instance.

**Type:**

String

**Optionality:**

Optional (default used if not set).

**Allowed:**

Default: `/`

**Notes:**

- Cannot be used with the `-u usr/pwd` (on page 64) option.
Example: --user smf/smf

Failure
In case of failure alarms will be raised to the syslog.

Output
There is no output from this process.

**dapMacroNodes**

**Purpose**
This slee_acs plug-in provides the DAP macro nodes.
The nodes provided are:
- Send Request
- DAP Request
For more information about:
- Macro node libraries, see *ACS Technical Guide*
- The CPE, see *CPE User's Guide*

**Location**
This library is located on SLCs.

**Startup**
If dapMacroNodes is configured in *acs.conf*, it is made available to slee_acs when slee_acs is initialized. It is included in the *acsChassis* section of *acs.conf* in a *MacroNodePluginFile* entry as follows:

acsChassis
MacroNodePluginFile dapMacroNodes.so

**Configuration**
dapMacroNodes is configured in the *DAP* section of *eserv.config*. For more information, see *DAP eserv.config configuration* (on page 48).

**dapTypeConversion**

**Purpose**
This SLEE-ACS plug-in provides conversions from ACS profile fields to types usable by DAP.

**Location**
Located on SLCs.
Chapter 5

Startup
This library will be loaded based on configuration made by the packages on install.

Configuration
This library has no specific configuration.

libdapChassisActions

Purpose
This slee_acs plug-in implements the chassis actions which are used by the DAP macro nodes when they need to interact with components outside slee_acs.

Location
This library is located on SLCs.

Startup
If libdapChassisActions is configured in acs.conf, it is made available to slee_acs when slee_acs is initialized. It is included in the acsChassis section of acs.conf in a ChassisPlugin entry.

acsChassis
  ChassisPlugin libdapChassisActions.so

Configuration
This binary has no specific configuration.

libDAPManager.so

Purpose
The libDAPManager.so is a combined connection manager and XML interface. A DAP client will use libDAPManager.so to communicate with an ASP.

Start-up
The library is linked at run time by a DAP client.

Location
libDAPManager.so is located wherever a DAP client is installed (for example, on SMS, SLC or VWS.

eserv.config configuration
libDAPManager has configuration available in the DAP section of eserv.config. For more information, see DAP eserv.config configuration (on page 48).

Command line parameters
Command line parameters for libDAPManager.so are the same as those for dapIF. Refer to Parameters.
Failure
In the event of a failure, alarms will be written to the system log.

Output
There is no output from this process.

openssl

Purpose
openssl is used by `dapReadyCertificates.sh` (on page 69) to start `c_rehash` (on page 63) with appropriate configuration.

Location
This binary is located on both SLCs and VWSs.

Startup
openssl is started by `dapReadyCertificates.sh` (on page 69).

Note: `dapReadyCertificates.sh` attempts to use openssl at the location specified by `openSSLPATH` (on page 54).

Configuration
openssl has some configuration which is set when used by `dapReadyCertificates.sh` (on page 69). To configured this process, use the configuration available to `dapReadyCertificates.sh`.

sqlite3

Purpose
sqlite3 is an embedded SQL database engine which reads and writes directly to database file on disk. Programs that link with the sqlite3 library can have SQL database access without running a separate RDBMS process.

It handles the DAP pending queue.

Location
This binary is located on SMSs.

Configuration
sqlite3 accepts the following command line parameters.

```
sqlite3 /tmp/pendingRequests.db '<VACUUM|PRAGMA integrity_check>;'  
```

To resize the database after a lot of data has been removed, run VACUUM as `acs_oper`.

To raise alarms about malformed db files, use PRAGMA integrity_check.
Overview

Introduction

This chapter explains the Oracle Communications Network Charging and Control Data Access Pack DAP tools and utilities that are available.

In this chapter

This chapter contains the following topics.

dapReadyCertificates.sh 69
dapSchemaTool 70

dapReadyCertificates.sh

Purpose

dapReadyCertificates.sh prepares the certificates in certificatePath (on page 50) into the form required by openssl (on page 67) (by concatenating them in human-readable form and running c_rehash (on page 63) on them). dapReadyCertificates.sh concatenates the files into the file specified by certificatesName (on page 50).

For an overview of how certificates are handled, see Certificate management (on page 11).

Warning: The concatenated file is overwritten each time dapReadyCertificates.sh is run. To keep the existing file, move it to a directory other than the one specified by certificatePath, or rename it with a suffix other than .pem (otherwise it will be concatenated into the new file along with the other *.pem files in certificatePath).

Location

This binary is located on both SLCs and VWSs.

Configuration

dapReadyCertificates.sh accepts the following parameters from the DAP section of eserv.config:

- certificatePath (on page 50)
- certificatesName (on page 50)
- openSSLPath (on page 54)
dapSchemaTool

Purpose
This tool is used to export existing ASPs and operations from one server (that is, a testing server) and import them to another server (that is, a production server).

Usage

Format

dapSchemaTool -E data|I data -n name [-s operation_set_name] [-f filename] -u db_user -p db_password [-c TNS_connect_string]

Export example

/dapSchemaTool -E asponly -n abc -u SMF -p SMF -c server_SMF

Import example

/dapSchemaTool -I operation -n xyz -f testOutput.txt -u SMF -p SMF -c server_SMF

Arguments

This table describes the function of each command argument.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-E</td>
<td>Export from database</td>
</tr>
<tr>
<td>-I</td>
<td>Import to database</td>
</tr>
<tr>
<td>-n</td>
<td>ASP, operation, or operation set name to export or import</td>
</tr>
<tr>
<td>-s</td>
<td>Operation set name (optional, only required if the given operation name is not unique)</td>
</tr>
<tr>
<td>-f</td>
<td>Filename for exported data (optional, required for importing, uses the standard output for exporting if not specified)</td>
</tr>
<tr>
<td>-u</td>
<td>Oracle username of database to use</td>
</tr>
<tr>
<td>-p</td>
<td>Oracle password of database to use</td>
</tr>
<tr>
<td>-c</td>
<td>Oracle TNS connect string (optional, uses $ORACLE_SID to connect to the local database if not specified)</td>
</tr>
</tbody>
</table>

Data arguments for -E and -I options - one of these must be used

| asponly  | Only the given ASP |
| asp      | The given ASP and all attached operations, as well as the operation set each operation belongs to, and all parameters each operation contains |
| operation| The given operation, its parameters, and the associated ASP and the operation set (the associated operation set will be exported but will not be imported using this argument) |
| operationset | The given operation set and its all operations, as well as the associated ASP and parameters of each operation |
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 71
Checking the Installation 71
Post Install Requirements 72

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

Data Access Pack packages

An installation of Oracle Communications Network Charging and Control Data Access Pack includes the following packages, on the:

- SMS:
  - dapSms
  - ccsDapSms
- SLC:
  - dapScp
- VWS:
  - dapExtras

Checking the Installation

Checking dapSms installation

On successful installation the dapSms package will have created the following directories:

- `/IN/service_packages/DAP/db`
- `/IN/service_packages/DAP/lib`
Chapter 7

- IN/service_packages/DAP/tmp
The following feature nodes will have been installed and added to the ACS database:
  - Send Request
  - DAP Send Request
  - DAP VXML

**Checking dapScp installation**

On successful installation the dapScp package will have created the following directories:

- IN/service_packages/DAP/bin
- IN/service_packages/DAP/db
- IN/service_packages/DAP/etc
- IN/service_packages/DAP/lib
- IN/service_packages/DAP/tmp

The following binaries and interfaces will have been installed:

- IN/service_packages/DAP/bin/dapIF

The following configuration files will have been installed:

- IN/service_packages/DAP/etc/example.eserv.config

The following shared libraries will have been installed:

- IN/service_packages/DAP/lib/dapMacroNodes.so
- IN/service_packages/DAP/lib/libdapChassisActions.so

**Post Install Requirements**

**Script to run**

The DAP/bin/postPinstall.sh script must be run to apply the triggers to the DAP/PI tables for DAP_USERS.

**Note:** This must be run after both DAP and PI have been installed.
NCC Glossary of Terms

ACS
Advanced Control Services configuration platform.

ANI
Automatic Number Identification - Term used in the USA by long-distance carriers for CLI.

ASP
- Application Service Provider, or

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

CLI
Calling Line Identification - the telephone number of the caller. Also referred to as ANI.

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

CPE
Control Plan Editor (previously Call Plan Editor) - software used to define the logic and data associated with a call - for example, "if the subscriber calls 0800 nnnnnn from a phone at location xxx then put the call through to bbb bbb bbb bbb".

cron
Unix utility for scheduling tasks.

crontab
File used by cron.

DAP
Data Access Pack. An extension module for ACS which allows control plans to make asynchronous requests to external systems over various protocols including XML and LDAP.

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.

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.

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).

Oracle

Oracle Corporation

PI

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

PICS

Protocol Implementation Conformance Statement applicable to the relevant protocol.

Service Provider

See Telco.

SGML

Standard Generalized Markup Language. The international standard for defining descriptions of the
structure of different types of electronic document.
SLC
Service Logic Controller (formerly UAS).

SLEE
Service Logic Execution Environment

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.

Telco
Telecommunications Provider. This is the company that provides the telephone service to customers.

Telecommunications Provider
See Telco.

URL
Uniform Resource Locator. A standard way of specifying the location of an object, typically a web page, on the Internet.
VWS
Oracle Voucher and Wallet Server (formerly UBE).

WSDL
Web Services Description Language.

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 ‘metalanguage’ — 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.
HTTP and HTTPS Connections • 3, 21
HTTP version 1.0 • 62
HTTP version 1.1 • 62
HTTPS connections • 10

id • 60
Import WSDL • 17, 37
Import WSDL Files • 37, 38
Import WSDL screen • 38, 39
Import WSDL Screen • 37
Importing a WSDL file • 38
IN • 74
Installation and Removal Overview • 71
InterfaceHandle • 3, 59
Introduction • 1, 5, 8, 12, 15, 16, 19, 20, 24, 35, 37, 40, 42, 45, 46, 48, 61, 62, 71
IP • 74
IP address • 74
ISDN • 74

Larger SLEE events • 61
libdapChassisActions • 3, 10, 66
libDAPManager.so • 66
listenHost • 10, 52
listenPort • 10, 12, 53
Loading eserv.config changes • 47
Location • 63, 64, 65, 66, 67, 69

Mapping • 53
Mapping parameters • 21, 53, 59, 62
maxQueueCheckTimeout • 53
maxQueueLength • 53
maxRetries • 53
Message contents • 14
Message flow • 3, 9, 10, 12, 53
Message header construction • 4, 9
MSISDN • 74
Multiple instances • 62
Multiple variables • 6

New Operation screen - Request • 25, 27
New Operation screen - Request Template • 25, 28
New Operation screen - Response • 29, 30
New Operation screen - Response Parameters • 29, 31
nonBlockingConnections • 54

Operation Request Configuration • 37, 39, 40
Operation Response Configuration • 37, 39, 42
Operation Set Configuration screen • 35, 36
Operation Sets • 19, 35, 39
Operation Sets tab • 35
Operations • 2, 19, 24
Operations tab • 24
Operations tab fields • 24, 25, 31
Oracle • 74
Other encodings • 15
Output • 65, 67
Overview • 1, 19, 37, 45, 63, 69, 71

Parameter substitution • 3, 9, 12
pendingFilename • 54
pendingQueueInMemory • 55
persistentConnectionCheckTimeout • 55
persistentConnections • 55
PI • 74
PICS • 74
PollCount • 55
PollInterval • 55, 56
Post Install Requirements • 72
prefix • 60
prefixTagName • 56
Prerequisites • v
Profile Tag Formats • 8
Protocol • 60
Purpose • 63, 64, 65, 66, 67, 69, 70

Related documents • v
Repetition of subtrees • 5
Request parameter configuration • 25, 28, 32
Request tab • 40
Resources • 9, 17, 19, 45, 64
Resources Screen • 19, 39
Response parameter configuration • 26, 29, 32
Response status/details command processed • 12
Response status/details command received • 12
Response tab • 42
Response validation • 11
responseTagName • 56
retryTimeout • 56

Scope • v
Script format • 25, 26
Script to run • 72
sendHeaderTag • 61
sendRequestDateFormat • 57
sendRequestDateTZ • 57
Server authentication • 11
Service Provider • 74
sessionTimeout • 57
SGML • 74
SLC • 75
SLEE • 75
SLEE.cfg Configuration • 61, 62
SMS • 75
SMS main menu • 17
SN • 75
SOAP • 3, 9, 75
SOAP bindings • 14
SOAP support over HTTP • 11
soapAction • 61
soapHeaderOverride • 57
SQL • 75
sqlite3 • 67
SRF • 75
SSP • 75
Startup • 45, 63, 64, 65, 66, 67
Start-up • 66
Statistics • 15, 45
Statistics collected • 15
suffixTagName • 58
Supported protocols • 3
Supported tag types • 8
Synchronous and asynchronous connections • 2, 25
Synchronous message flow • 13
Synchronous request • 13
System Overview • 1

T
TCP • 75
Telco • 75
Telecommunications Provider • 75
Template contents • 7
templates • 58
templates parameters • 58, 60
timestampTagName • 5, 58
Tools and Utilities • 69
Transmission services • 15
Typographical Conventions • vi

U
-u usr/pwd • 64, 65
uncorrelatedRequestDir • 59
URL • 75
Usage • 70
--user /usr/pwd • 64
useTemplateSOAPTags • 59

V
Variables • 5
VWS • 76

W
What is Data Access Pack? • 1
What is WSDL? • 37