Oracle Communications® ASAP™ Cartridge 1.0
GA Release for Insignia OTAC 3.0

Insignia OTAC 3.0 Cartridge Guide

Fifth Edition
July 2008

ORACLE®

Copyright and Trademark Information

Contents

1

Cartridge Overview

ASAP cartridges are discrete software components that are developed for the ASAP product. An ASAP cartridge offers specific domain behavior on top of the core ASAP software, and provides the configuration that supports a set of services on a network element (NE).

An ASAP cartridge is not a stand-alone component, but operates in conjunction with the ASAP core product. ASAP cartridges offer the following benefits:

- **Reduced Time to Market** - time to market of new services is reduced through simplified development, implementation, and extension of cartridges on customer sites.
- **Extendable** - cartridges can be extended to include additional services and components that deliver business value, without requiring changes to the original cartridge.
- **Simplified Effort** - the effort and technical knowledge that is required to perform customizations is reduced.
- **Ease of Installation** - cartridges can be installed into an ASAP environment without interfering with the existing install base.

An ASAP cartridge can be used to configure ASAP to provision the following:

- NEs from a specific vendor, such as Nortel or Lucent.
- Technologies, such as Asynchronous Transfer Mode (ATM) and Frame Relay switches, or Internet Protocol (IP) routers.
- Services that are supported on the NE, such as ATM, IP Virtual Private Networks (VPN), Wireless, or Optical.

Cartridges are designed for a specific technology, software load, and service.

An ASAP cartridge supports a particular set of services on an NE. These services are independent of customer-specific service definitions. Professional Services or systems integrators can perform extensions to the cartridge to support customer-specific requirements.

Cartridge content

An ASAP cartridge contains the following:

- Sample NE configuration
- A set of scripts, such as State Tables or Java methods

- A set of atomic actions in the form of Atomic Service Description Layer (ASDL) commands
- A set of Common Service Description Layer (CSDL) commands that form meaningful services
- Sample work orders
- Installation scripts

Prerequisites

System integrators such as managers, designers, programmers, and testers who are responsible for the adaptation and integration of ASAP-based solutions should use this manual as a reference. It assumes that readers possess the following skills:

- A knowledge of ASAP programming concepts
- A good working knowledge of the UNIX operating system
- A thorough understanding of service and network provisioning
- Familiarity with telecommunications

About this guide

This guide provides a detailed description of the Insignia OTAC cartridge. It contains overview and technical information to assist with extending and integrating the cartridge into a customer environment.

The scope of this guide includes ASAP as it pertains to the use of this cartridge. It is not intended to be a complete ASAP reference guide.

For additional information when using this cartridge, refer to the following supporting documentation:

- **Activation documentation set**—for detailed information on the ASAP component.

The Insignia OTAC cartridge provides the ASAP service configuration and network element (NE) interface to send a request to the customer to enable Wireless Application Protocol/ Multimedia Messaging Service feature on NE_INSIGNIA-OTAC_3-0_HOST NEs.

Services, features, and options

This cartridge supports the following services:

Table 1: Supported services

Service	Description
Submit SLR.	This service submits a service level request.
Submit HTR.	This service submits a handset type request.

Hardware and software requirements

The following sections contain the high-level software and hardware environment requirements for provisioning Wireless Application Protocol/Multimedia Messaging Service services on authentication center including:

- Network element (NE) interface
- ASAP version

Network element (NE) interface

The following database tables in SARM are configured to support the NE configuration:

- tbl_host_clli
- tbl_clli_route
- tbl_comm_param
- tbl_resource_pool
- tbl_ne_config

ASAP version

This cartridge was developed and tested using ASAP version 4.7.2.

For more information on the operating environment of this ASAP version, refer to the ASAP version 4.7.2 Release Record.

Connecting to the NE

The cartridge uses the Socket over HTTP protocol.

Related documentations

This cartridge was developed according to the following Network Element Provisioning Specifications:

- PAC489.doc

2

Installing and Testing the Cartridge

This chapter describes the following procedures related to installing and testing the cartridge:

- Downloading the cartridge
- Installing the cartridge using scripts
- Uninstalling the cartridge using scripts
- Testing the cartridge installation
- Deployment of the cartridge using Studio
- Uninstallation and Undeployment of the cartridge using Studio

Downloading the cartridge

Before you can install the cartridge, you must use the internet to download the cartridge's TAR file from Oracle's Customer Portal.

Use the following instructions to download, then unTAR the TAR file.

To download the TAR file

1. Login to Oracle MetaLink internet home page (http://www.metalink.oracle.com).
2. Download the cartridge patch to your workstation.

To unTAR the TAR file

1. On your workstation, create a repository directory—the naming of which is your choice.

   ```
   mkdir <repository dir>
   ```

2. Copy the TAR file into the repository directory.
3. Untar InsigniaOTAC_3_0_X_R1_0_0.<buildId>.tar.

   ```
   tar xvf InsigniaOTAC_3_0_X_R1_0_0.<buildId>.tar
   ```

The directory structure in the repository directory should look like the following illustration. (this illustration describes the minimum required structure; you can enhance this directory structure with additional directories based on your requirements and deliverables).

```
<repository_directory>
       /README
       /installCartridge
       /uninstallCartridge
```

```
/INSIGNIA_OTAC_3_0_R1_0_0.sar
```

Starting ASAP

Before installing the cartridge, ensure that ASAP is running.

To start ASAP

1. To start ASAP, execute the following script:

   ```
   start_asap_sys -d
   ```

2. Ensure the ASAP Daemon (DAEM$ENV_ID) is running by checking the ASAP status using the ASAP script “status”.
3. Check whether the WebLogic instance for this ASAP environment is running. If not, start the WebLogic instance.

The *ASAP Administration Guide* contains more information on starting ASAP, the ASAP Daemon, and WebLogic.

Installing the cartridge using scripts

Run the installation script *installCartridge* to install the cartridge. The script executes the following tasks:

- Configures the Insignia OTAC-specific NE using the SACT.
- Deploys the Insignia OTAC cartridge service model (only if the Insignia OTAC service model is not yet deployed) using the Service Activation Deployment Tool (SADT).
- Copies the Insignia OTAC-specific jar files to the ASAP environment.
- Loads the sample work orders to the SRP database.

For information on the SACT and the SADT, refer to the *ASAP Administration Guide*.

To install the cartridge

1. Run the installCartridge script. At the prompt, type:

   ```
   installCartridge INSIGNIA_OTAC_3_0_R1_0_0.sar
   ```

2. The script prompts you for the values of the following WebLogic login parameters:

   - WebLogic Hostname
   - WebLogic HTTP Port
   - WebLogic Login User ID
   - WebLogic Login Password

   The script loads the NEP-NE configuration and the CSDL-ASDL configuration to the SARM database, and loads sample work orders to the SRP database. The script also copies the cartridge-specific jar files and cpp library file to the ASAP environment.

3. Copy studio_2_6_0.jar file to the $ASAP_BASE/lib directory.
4. Add ${ASAP_BASE}/lib/studio_2_6_0.jar to the CLASSPATH in the JInterpreter file under $ASAP_BASE/programs directory.
5. Restart ASAP to upload the cartridge configuration into ASAP.

Uninstalling the cartridge using scripts

Run the uninstallation script *uninstallCartridge* to uninstall the Insignia OTAC cartridge. The script executes the following tasks:

- Unconfigures Insignia OTAC-specific NEs using the SACT.
- Undeploys the Insignia OTAC cartridge service model (only if the Insignia OTAC service model is already deployed) using the Service Activation Deployment Tool (SADT).
- Removes the Insignia OTAC-specific jar files and cpp library file from the ASAP environment.

For more information on the SACT and the SADT, refer to the *ASAP Administration Guide*.

To uninstall the cartridge

1. Run the *uninstallCartridge* script. At the prompt, type:

```
uninstallCartridge INSIGNIA_OTAC_3_0_R1_0_0.<timestamp>.sar
```

2. The script prompts you for the values of the following parameters:
   - WebLogic Hostname
   - WebLogic HTTP Port
   - WebLogic Login User ID
   - WebLogic Login Password

The script unloads the NEP-NE configuration and CSDL-ASDL configuration from SARM database. It also removes the cartridge specific jar files and cpp library file from the ASAP environment.

Testing the cartridge installation

To test this cartridge installation, you need to know about the network element (NE), services, and basic Activation configuration. You may need to perform adjustments to provision a service for a specific NE, network, or connectivity configuration.

You can test the cartridge installation using one of the following methods:

- **Loopback mode**—does not actually connect to or send commands to the NE.
- **Live mode**—connects to and sends commands to a live NE.

Configuring loopback and live mode parameters

Set the following variables to test the cartridge in loopback or live testing modes.

Loopback mode

Set the following parameter to test the cartridge in loopback mode.

Table 2: Loopback Mode Parameter Settings

Configuration Variable	Parameter Settings	Location
LOOPBACK_ON	1 (default setting)	ASAP.cfg

Live mode

Set the following parameter to test the cartridge in live mode.

Table 3: Live Mode Parameter Settings

Configuration Variable	Parameter Settings	Location
LOOPBACK_ON	0	ASAP.cfg

Communication parameters

The following are the list of parameters for the sample NE configuration XML used by SACT.

Table 4: Communication parameters

param_label	param_value	param_desc
OTAC_URL	http://10.6.1.90/	URL to the OTAC Mediator
CONNECTION_TIMEOUT	20	Connection timeout
READ_TIMEOUT	30	Time to wait for a response from the server
RESPONSELOG	TRUE	Enable / Disable response log

Modifying NE_INSIGNIA-OTAC_3-0_HOST.xml

Use the following procedure to modify NE_INSIGNIA-OTAC_3-0_HOST.xml.

To modify NE_INSIGNIA-OTAC_3-0_HOST.xml

1. Create a new source directory. You can give this directory any appropriate, meaningful name you want to.

   ```
   mkdir <new_source_directory>
   ```

2. Copy INSIGNIA_OTAC_3_0_R1_0_0.sar to this new source directory.

   ```
   cp INSIGNIA_OTAC_3_0_R1_0_0.sar ./<new_source_directory>
   ```

3. Change directory to <new_source_directory>.

   ```
   cd <new_source_directory>
   ```

4. Un-jar INSIGNIA_OTAC_3_0_R1_0_0.sar. This extracts the contents of the sar file.

   ```
   jar xvf INSIGNIA_OTAC_3_0_R1_0_0.sar
   ```

5. Edit <new_source_directory> NE_INSIGNIA-OTAC_3-0_HOST.xml in with the appropriate changes.
6. Create a new sar file at the <new_source_directory> level.

   ```
   CreateSar $PWD
   ```

7. Uninstall the cartridge using INSIGNIA_OTAC_3_0_R1_0_0.sar. (That is, use the original sar file that you copied in Step 2 above—see “Uninstalling the cartridge using scripts” on page 7 for uninstallation instructions).
8. After you uninstall the cartridge, rename the sar file, so you have a backup copy of it.
9. Copy the new sar file from <new_source_directory>.
10. Reinstall the cartridge (see “Installing the cartridge using scripts” on page 7 for installation instructions).

Testing the installation

The following procedure describes the steps required to test the cartridge installation in loopback mode. We recommend that you perform the initial cartridge installation test in loopback mode.

To test in loopback mode

1. Stop ASAP by typing the following command at the UNIX prompt:

   ```
   stop_asap_sys -d
   ```

2. Ensure loop back mode is on. See “Loopback mode” on page 8 for a description of how to set the loop back parameter to “On”.
3. Start ASAP by typing:

   ```
   start_asap_sys -d
   ```

4. Send the sample work orders through the SRP Emulator by typing:

   ```
   run_suite $SRP <ctrl_password> <suite name>
   ```

   You can locate the suite names by typing:

   ```
   grep SUITE * | grep -v END
   ```

   A list of all available suites appears.

   For more information on the SRP Emulator, refer to the *ASAP Administration Guide*.
5. Verify the status of the sample work orders by typing:

   ```
   asap_utils -d l
   ```

   All successful work orders returns to the 104 state.

   To view the sample work orders provided with this cartridge, refer to the Insignia OTAC cartridge source.

Viewing the sample work orders

You find the sample work orders under the **SampleWorkOrders** directory in the sar file. The following procedure describes how to view the sample work orders.

To view the sample work orders

1. Create a repository directory, copy the sar file to the new directory and un-jar the sar file, as described by Step 1 through Step 4 in "Modifying NE_INSIGNIA-OTAC_3-0_HOST.xml" on page 9.
2. Locate and view the sample work order files.

Deployment of the cartridge using Studio

Before installing the cartridge, ensure weblogic and ASAP are started and running.

The following are the steps involved:

1. Open Studio in design perspective. Choose **Import** from the **File** menu and select **Activation Archive (SAR)** under **Studio Wizards** to import the sar file. Browse for the path to the sar file and click **Finish**.
2. Create a new **Service Activation Project**.
3. Define a new **NE Entity**, based on the **NE Template** contained in the cartridge provided by Oracle.
4. Ensure that the primary pool of the newly created NE is different from the NE template primary pool. You can modify it, if necessary.
5. Ensure that the test work order provided with the cartridge targets the newly defined NE. If not, then modify the test work orders file(s).
6. Create a new **Activation Environment Project** from the **Studio** menu. (Use Studio help for more information).
7. Create **Activation Environment** inside the **Activation Environment Project** and configure the **Connection Details** tab with your Environment ID, Activation version and weblogic data.
8. Connect to your environment using the **Connect** button.
9. Select the **Cartridge** tab of the **Activation Environment** and click **Add** to add your projects to the environment. The cartridge and the newly created **Service Activation** should appear in the **Cartridges** list.
10. Deploy the **NetworkActivation** (NA) cartridge provided by Oracle. (No NE information is to be deployed with this cartridge, therefore it isn't necessary to deploy the **NEP map** info).
11. Deploy the **Service Activation** (SA) project as follows:

- On the **Cartridge** tab, select the necessary SA cartridge and press the **Deploy** button.
- Select the **NEP Map** tab of the **Activation Environment**. Choose the necessary **NEP** server from the drop-down box of the **Network Element Processors**.(Use Studio help for more information).
- Select the SA cartridge from the **Network Element Processor Map** and click the **Deploy** button.

12. Verify the **SADT** console to confirm the installation.
13. Go to ASAP environment.
14. Copy studio_2_6_0.jar file to the $ASAP_BASE/lib directory.
15. Add ${ASAP_BASE}/lib/studio_2_6_0.jar to the CLASSPATH in the JInterpreter file under $ASAP_BASE/programs directory.
16. Restart **ASAP** in order to start working with the cartridge.

Uninstallation and Undeployment of the cartridge using Studio

The following are the steps involved:

1. Connect to your environment using the **Connect** button.
2. Select the necessary cartridge from the **Environment Cartridge** list in Studio and click the **Undeploy** button.
3. Verify the **Environment Cartridge** list. The Check Box with the name of the cartridge that is disabled should be unchecked.

3

Atomic Service Description Layer (ASDL) Commands

ASDL commands represent a set of atomic actions that ASAP can perform on a network element (NE). ASAP can combine ASDLs to create meaningful services (CSDLs) within a cartridge.

This chapter presents detailed information on the ASDL parameters that we provide with this cartridge. The following table lists and describes the type of parameter information that is included.

Table 5: ASDL parameter information

Item	Description
Parameter Name	Identifies the parameter that is configured for the stated service.
Description	Describes the parameter.
Range	Describes or lists the range of values that can be used to satisfy this parameter.
Default Value	Configures a default value for the parameter so that it is not mandatory for the upstream system to provide a value.

Table 5: ASDL parameter information

Item	Description
Type	Indicates one of the following parameter types: ◆ S—Scalar, specifies the parameter label transmitted on the ASDL command. Scalar parameters are conventional name-value pair parameters. ◆ C—Compound, specifies the base name of the compound parameter transmitted on the ASDL command. A compound parameter contains structures or arrays of information that are represented by a particular structure name or compound parameter name. Each compound parameter can contain a large number of elements. If you use compound parameters, you only require a single entry in the ASAP translation tables to call the compound parameter and all its associated parameter elements. ◆ I—Indexed, identifies a parameter that contains a sequential numerical index value to tell the SARM that it should execute the same operation (for example, an ASDL command) for all occurrences of that index. Consequently, if there are several options on a particular CSDL command (OPT1, OPT2, OPT3, etc.), you can specify the OPT parameter as an indexed parameter. When you specify the OPT parameter as an indexed parameter, the SARM generates several occurrences of that same ASDL command and each command has a different value for the option being transmitted to the NEP. For more information on parameter types, refer to the *ASAP Developer Reference*.
Class	Indicates one of the following parameter classifications: ◆ R—Required scalar parameter ◆ O—Optional scalar parameter ◆ C—Required compound parameter ◆ N—Optional compound parameter ◆ M—Mandatory indexed parameter ◆ I—Optional indexed parameter ◆ S—Parameter count

For a detailed description of the Required and Optional parameter classifications, refer to the *ASAP Administration Guide.*

ASDL commands

This cartridge provides the following ASDL commands:

- A_INSIGNIA-OTAC_3-0_SUBMIT_HTR
- A_INSIGNIA-OTAC_3-0_SUBMIT_SLR

A_INSIGNIA-OTAC_3-0_SUBMIT_HTR

Provisioning based on the handset type.

It is implemented by the Java method:
com.mslv.activation.cartridge.insignia.otac.x3_0.htr.submit.generated.SubmitHtrProxy.execute

Table 6: A_INSIGNIA-OTAC_3-0_SUBMIT_HTR

Parameter Name	Description	Range	Default Value	Type	Class
MCLI	The Insignia NE identifier			S	R
DESTINATION	Destination MSISDN number			S	R
GROUP_ID	Group ID			S	R
PHONE_ID	Phone ID			S	R
NETWORKPIN	IMSI of the handset			S	O
HOMEPAGE	Home page sent in the OTAC message			S	O
USERNAME	ISP user name			S	O
PASSWORD	ISP password			S	O

MML Commands

```
Requesting OTAC from the WOS based on the handset type. The following string
will be generated with GET Method.

<OTAC_URL + telstra/otacSendInterface.do?>destination=<DESTINATION>

&group_id=<GROUP_ID>

&phone_id=<PHONE_ID>&networkpin=<NETWORKPIN>&Homepage=<HOMEPAGE>&username=<U
SERNAME>&password=<PASSWORD>
```

```
Note:
OTAC_URL Value should be defined similar to http://127.0.0.1/
```

Output Parameters

Return as CSDL Parameter:

A_INSIGNIA-OTAC_3-0_SUBMIT_HTR_UDET=<user defined exit type>

Return as Info Parameter:

A_INSIGNIA-OTAC_3-0_SUBMIT_HTR_RETURN_INFO=<Response Code>:<Response Text>

A_INSIGNIA-OTAC_3-0_SUBMIT_SLR

Provisioning based on the service level.

It is implemented by the Java method: **com.mslv.activation.cartridge.insignia.otac.x3_0.slr.submit.generated.SubmitSlrProxy.execute**

Table 7: A_INSIGNIA-OTAC_3-0_SUBMIT_SLR

Parameter Name	Description	Range	Default Value	Type	Class
MCLI	The Insignia NE identifier			S	R
DESTINATION	Destination MSISDN number			S	R
SERVICELEVEL	Service level of subscriber			S	R
USERNAME	ISP user name			S	O
PASSWORD	ISP password			S	O
HOMEPAGE	Home page sent in the OTAC message			S	O

MML Commands

```
Requesting OTAC from the WOS based on the service level. The following string
will be generated with GET Method.
```

```
<OTAC_URL + telstra/serviceCodeSendInterface.do?>Destination=<DESTINATION>&
ServiceLevel=<SERVICELEVEL>&Homepage=<HOMEPAGE>&username=<USERNAME>&password
=<PASSWORD>
```

```
Note:
```

```
OTAC_URL Value should be defined similar to http://127.0.0.1/
```

Output Parameters

Return as CSDL Parameter:

A_INSIGNIA-OTAC_3-0_SUBMIT_SLR_UDET=<user defined exit type>

Return as Info Parameter:

A_INSIGNIA-OTAC_3-0_SUBMIT_SLR_RETURN_INFO=<Response Code>:<Response Text>

User exit types

User exit types allow cartridge developers and systems administrators to map ASDL exit codes to one of the predefined base exit types. Base exit types determine the product behavior. Cartridges map return codes and status values from a network element to a user defined exit type.

Regular expressions (regex) are used to perform pattern searches on responses from network elements. The pattern is stored in "tbl_user_err" in the SARM database. The user exit type contains a regex pattern that is applied at runtime.

Regular expressions enable users to associate a series of responses to a specific base type. For example, a regular expression "6" can identify a pattern where any response with the character "6" followed by any number of characters will translate to base type of FAIL.

Regular expressions can also allow very specific searches within a response from a network element. Regular expressions are typically compiled before being executed. Compilation produces a binary version of the expression and ensures that the syntax of the regular expression is correct. This compilation occurs using SACT\SADT when user exit types are deployed into ASAP. If the syntax is deemed to be incorrect during compilation, SADT displays an error message and the deployment of the user exit type will fail.

For more information on pattern matching, refer to the *ASAP Developer Reference* and the *ASAP Administration Guide*.

Understanding user exit type XML files

```
…
<userDefinedExitType>
   <neDescriptor>
      <softwareLoad>DYNAMIC_SL</softwareLoad>
      <technology>DYNAMIC_VENDOR-DYNAMIC_TECH</technology>
   </neDescriptor>
   <searchPattern>SUCCESS.</searchPattern>[1]
   <userType>U_SUCCEED</userType>[2]
   <baseType>SUCCEED</baseType>[3]
   <description>The ASDL provisioning was successful</description>
</userDefinedExitType>
<userDefinedExitType>
   <searchPattern>90.</searchPattern>
   <userType>U_FAIL</userType>
   <baseType>FAIL</baseType>
   <description>The ASDL failed - fail the current order
      and stop processing.</description>
</userDefinedExitType>
<userDefinedExitType>
   <searchPattern>101-110[201-215]</searchPattern>[4]
   <userType>U_SOFT_FAIL</userType>
   <baseType>SOFT_FAIL</baseType>
   <description>The ASDL has encountered a soft failure. Processing will
      continue.</description>
</userDefinedExitType>
<userDefinedExitType>
   <searchPattern>801-850</searchPattern>[5]
   <userType>U_MINOR_ERROR</userType>
   <baseType>SOFT_FAIL</baseType>
   <description>The ASDL has encountered a soft failure. Processing will
      continue.</description>
</userDefinedExitType>
<userDefinedExitType>
```

1. Pattern searches accommodate situations in which responses from the device contain small variants that represent the same meaning. The user type contains an associated search pattern that is applied at runtime. Using regular expressions, you can default a series of responses. For example a regular expression "90." can specify a pattern where any response with the character "90" followed by any character will translate to base type of FAIL. If the regular expression is defined as "90*", then any response with the character "90" followed by any number of characters will translate to base type of FAIL
2. The user type that the search pattern maps to.
3. The base type that maps to the user type.
4. 101 to 110 and 201 to 215 will translate to a base type of SOFT_FAIL
5. 801-850 will translate to a base type of SOFT_FAIL. Note that the user type differs from the previous range.

```
      <searchPattern>251-275&&[^261-265]</searchPattern>[1]
      <userType>U_DELAYED_FAIL</userType>
      <baseType>DELAYED_FAIL</baseType>
      <description>The ASDL has failed during provisioning.</description>
   </userDefinedExitType>
   <userDefinedExitType>
      <neDescriptor>
         <softwareLoad>BCS36</softwareLoad>
         <technology>NORTEL_DMS</technology>
         <neVendor>Nortel</neVendor>
      </neDescriptor>
      <searchPattern>*.</searchPattern>
      <userType>U_MAINTAIN</userType>
      <baseType>MAINTENANCE</baseType>
      <description>The ASDL will Wait until the NE comes out of
         Maintenance Mode</description>
   </userDefinedExitType>
```

The previous code sample shows some typical search pattern examples. Some additional examples follow:

- ^.*\b(one|two|three)\b.*$ = matches a complete line of text that contains any of the words "one", "two" or "three"
- ^(?=.*?\bone\b)(?=.*?\btwo\b)(?=.*?\bthree\b).*$ matches a complete line of text that contains all of the words "one", "two" and "three"
- "[^"\r\n]*" matches a single-line string that does not allow the quote character to appear inside the string.
- \b\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}\b matches any IP address.

For more information on search patterns, refer to http://java.sun.com/j2se/1.4.2/docs/api/java/util/regex/Pattern.html.

For more information on user exit types, refer to chapter 3 of the *ASAP Developer Reference*.

User defined ASDL exit types

The following table lists the user defined ASDL exit types.

Table 8: User defined ASDL exit types

Search pattern	User_type	Base_type	Description
200	REQUEST_OK	SUCCEED	Request has been received
400	BAD_REQUEST	FAIL	Required parameters were not supplied

1. 251 to 275 but not 261 to 265 will translate to a base type of DELAYED_FAILURE.

Table 8: User defined ASDL exit types

Search pattern	User_type	Base_type	Description
500	SERVER_ERROR	FAIL	Server failure
IO_EXCEPTION	JAVA_IO_EX	FAIL	Java IO exception
PROV_CARTRIDGE_EXCEPTION	INTERNAL_CART_EX	FAIL	Internal Cartridge Exception
GEN_JAVA_EXCEPTION	GEN_JAVA_EX	FAIL	Generic Java exception
NO_UDET_MATCH_FOUND	NO_MATCH_FOUND	FAIL	No match on UDET
301	PAGE_MOVED	FAIL	Page has moved permanently
302	PAGE_MOVED	FAIL	Page has moved temporarily
304	REQUEST_OK	SUCCEED	Page not modified
401	USER_UNAUTH	FAIL	Authentication is required
403	FORBIDDEN_REQUEST	FAIL	The request is forbidden
404	PAGE_NOT_FOUND	FAIL	Could not find the request URL
405	METHOD_NOT_ALLOWED	FAIL	The request method is not allowed
406	NOT_ACCEPTABLE	FAIL	Unacceptable response
407	PROXY_AUTH_REQD	FAIL	Client must be authenticated by proxy
408	REQUEST_TIMEOUT	FAIL	Server stopped waiting for client request
409	CONFLICT	FAIL	Request could not be completed due to conflict
410	PAGE_NOT_FOUND	FAIL	Could not find the request URL
411	LENGTH_REQD	FAIL	Content-length required in the request
412	PRECONDITION_FAIL	FAIL	Precondition failed in requested header fields

Table 8: User defined ASDL exit types

Search pattern	User_type	Base_type	Description
413	REQUEST_TOO_LARGE	FAIL	Request entity is too large to process
414	REQUEST_URL_TOO_LONG	FAIL	URL length exceeded maximum size
415	UNSUPP_MEDIA_TYPE	FAIL	Request format not supported
501	REQUEST_UNKNOWN	FAIL	Function not implemented in Web server software
502	BAD_GATEWAY	FAIL	Invalid response from server
503	SERVICE_UNAVAILABLE	FAIL	Service under maintenance
504	GATEWAY_TIMEOUT	FAIL	Server has not responded
505	UNSUPP_HTTP_VER	FAIL	HTTP version not supported
NUMBER_FORMAT_JAVA_EXCEPTION	NUMBER_FORMAT_EX	FAIL	Number format exception
HTTP_CARTRIDGE_EXCEPTION	HTTP_CART_EX	FAIL	HTTP cartridge exception

UserExitType.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<serviceModel xmlns:ude="http://www.mslv.com/studio/activation/model/
userDefinedExitType" xmlns:sm="http://www.metasolv.com/ServiceActivation/
2003/ServiceModel" xmlns="http://www.metasolv.com/ServiceActivation/2003/
ServiceModel" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <userDefinedExitType>
      <neDescriptor>
         <softwareLoad>3-0</softwareLoad>
         <technology>OTAC</technology>
         <neVendor>INSIGNIA</neVendor>
      </neDescriptor>
      <searchPattern>200</searchPattern>
      <userType>REQUEST_OK</userType>
      <baseType>SUCCEED</baseType>
      <description>Request has been received</description>
   </userDefinedExitType>
   <userDefinedExitType>
      <neDescriptor>
         <softwareLoad>3-0</softwareLoad>
```

```
            <technology>OTAC</technology>
            <neVendor>INSIGNIA</neVendor>
        </neDescriptor>
        <searchPattern>400</searchPattern>
        <userType>BAD_REQUEST</userType>
        <baseType>FAIL</baseType>
        <description>Required parameters were not supplied</description>
    </userDefinedExitType>
    <userDefinedExitType>
        <neDescriptor>
            <softwareLoad>3-0</softwareLoad>
            <technology>OTAC</technology>
            <neVendor>INSIGNIA</neVendor>
        </neDescriptor>
        <searchPattern>500</searchPattern>
        <userType>SERVER_ERROR</userType>
        <baseType>FAIL</baseType>
        <description>Server failure</description>
    </userDefinedExitType>
    <userDefinedExitType>
        <neDescriptor>
            <softwareLoad>3-0</softwareLoad>
            <technology>OTAC</technology>
            <neVendor>INSIGNIA</neVendor>
        </neDescriptor>
        <searchPattern>IO_EXCEPTION</searchPattern>
        <userType>JAVA_IO_EX</userType>
        <baseType>FAIL</baseType>
        <description>Java IO exception</description>
    </userDefinedExitType>
    <userDefinedExitType>
        <neDescriptor>
            <softwareLoad>3-0</softwareLoad>
            <technology>OTAC</technology>
            <neVendor>INSIGNIA</neVendor>
        </neDescriptor>
        <searchPattern>PROV_CARTRIDGE_EXCEPTION</searchPattern>
        <userType>INTERNAL_CART_EX</userType>
        <baseType>FAIL</baseType>
        <description>Internal Cartridge Exception</description>
    </userDefinedExitType>
    <userDefinedExitType>
        <neDescriptor>
            <softwareLoad>3-0</softwareLoad>
            <technology>OTAC</technology>
            <neVendor>INSIGNIA</neVendor>
        </neDescriptor>
        <searchPattern>GEN_JAVA_EXCEPTION</searchPattern>
        <userType>GEN_JAVA_EX</userType>
        <baseType>FAIL</baseType>
        <description>Generic Java exception</description>
    </userDefinedExitType>
```

```
<userDefinedExitType>
   <neDescriptor>
      <softwareLoad>3-0</softwareLoad>
      <technology>OTAC</technology>
      <neVendor>INSIGNIA</neVendor>
   </neDescriptor>
   <searchPattern>NO_UDET_MATCH_FOUND</searchPattern>
   <userType>NO_MATCH_FOUND</userType>
   <baseType>FAIL</baseType>
   <description>No match on UDET</description>
</userDefinedExitType>
<userDefinedExitType>
   <neDescriptor>
      <softwareLoad>3-0</softwareLoad>
      <technology>OTAC</technology>
      <neVendor>INSIGNIA</neVendor>
   </neDescriptor>
   <searchPattern>301</searchPattern>
   <userType>PAGE_MOVED</userType>
   <baseType>FAIL</baseType>
   <description>Page has moved permanently</description>
</userDefinedExitType>
<userDefinedExitType>
   <neDescriptor>
      <softwareLoad>3-0</softwareLoad>
      <technology>OTAC</technology>
      <neVendor>INSIGNIA</neVendor>
   </neDescriptor>
   <searchPattern>302</searchPattern>
   <userType>PAGE_MOVED</userType>
   <baseType>FAIL</baseType>
   <description>Page has moved temporarily</description>
</userDefinedExitType>
<userDefinedExitType>
   <neDescriptor>
      <softwareLoad>3-0</softwareLoad>
      <technology>OTAC</technology>
      <neVendor>INSIGNIA</neVendor>
   </neDescriptor>
   <searchPattern>304</searchPattern>
   <userType>REQUEST_OK</userType>
   <baseType>SUCCEED</baseType>
   <description>Page not modified</description>
</userDefinedExitType>
<userDefinedExitType>
   <neDescriptor>
      <softwareLoad>3-0</softwareLoad>
      <technology>OTAC</technology>
      <neVendor>INSIGNIA</neVendor>
   </neDescriptor>
   <searchPattern>401</searchPattern>
   <userType>USER_UNAUTH</userType>
```

```
      <baseType>FAIL</baseType>
      <description>Authentication is required</description>
   </userDefinedExitType>
   <userDefinedExitType>
      <neDescriptor>
         <softwareLoad>3-0</softwareLoad>
         <technology>OTAC</technology>
         <neVendor>INSIGNIA</neVendor>
      </neDescriptor>
      <searchPattern>403</searchPattern>
      <userType>FORBIDDEN_REQUEST</userType>
      <baseType>FAIL</baseType>
      <description>The request is forbidden</description>
   </userDefinedExitType>
   <userDefinedExitType>
      <neDescriptor>
         <softwareLoad>3-0</softwareLoad>
         <technology>OTAC</technology>
         <neVendor>INSIGNIA</neVendor>
      </neDescriptor>
      <searchPattern>404</searchPattern>
      <userType>PAGE_NOT_FOUND</userType>
      <baseType>FAIL</baseType>
      <description>Could not find the request URL</description>
   </userDefinedExitType>
   <userDefinedExitType>
      <neDescriptor>
         <softwareLoad>3-0</softwareLoad>
         <technology>OTAC</technology>
         <neVendor>INSIGNIA</neVendor>
      </neDescriptor>
      <searchPattern>405</searchPattern>
      <userType>METHOD_NOT_ALLOWED</userType>
      <baseType>FAIL</baseType>
      <description>The request method is not allowed</description>
   </userDefinedExitType>
   <userDefinedExitType>
      <neDescriptor>
         <softwareLoad>3-0</softwareLoad>
         <technology>OTAC</technology>
         <neVendor>INSIGNIA</neVendor>
      </neDescriptor>
      <searchPattern>406</searchPattern>
      <userType>NOT_ACCEPTABLE</userType>
      <baseType>FAIL</baseType>
      <description>Unacceptable response</description>
   </userDefinedExitType>
   <userDefinedExitType>
      <neDescriptor>
         <softwareLoad>3-0</softwareLoad>
         <technology>OTAC</technology>
         <neVendor>INSIGNIA</neVendor>
```

```
        </neDescriptor>
        <searchPattern>407</searchPattern>
        <userType>PROXY_AUTH_REQD</userType>
        <baseType>FAIL</baseType>
        <description>Client must be authenticated by proxy</description>
    </userDefinedExitType>
    <userDefinedExitType>
        <neDescriptor>
            <softwareLoad>3-0</softwareLoad>
            <technology>OTAC</technology>
            <neVendor>INSIGNIA</neVendor>
        </neDescriptor>
        <searchPattern>408</searchPattern>
        <userType>REQUEST_TIMEOUT</userType>
        <baseType>FAIL</baseType>
        <description>Server stopped waiting for client request</description>
    </userDefinedExitType>
    <userDefinedExitType>
        <neDescriptor>
            <softwareLoad>3-0</softwareLoad>
            <technology>OTAC</technology>
            <neVendor>INSIGNIA</neVendor>
        </neDescriptor>
        <searchPattern>409</searchPattern>
        <userType>CONFLICT</userType>
        <baseType>FAIL</baseType>
        <description>Request could not be completed due to conflict

        </description>

    </userDefinedExitType>
    <userDefinedExitType>
        <neDescriptor>
            <softwareLoad>3-0</softwareLoad>
            <technology>OTAC</technology>
            <neVendor>INSIGNIA</neVendor>
        </neDescriptor>
        <searchPattern>410</searchPattern>
        <userType>PAGE_NOT_FOUND</userType>
        <baseType>FAIL</baseType>
        <description>Could not find the request URL</description>
    </userDefinedExitType>
    <userDefinedExitType>
        <neDescriptor>
            <softwareLoad>3-0</softwareLoad>
            <technology>OTAC</technology>
            <neVendor>INSIGNIA</neVendor>
        </neDescriptor>
        <searchPattern>411</searchPattern>
        <userType>LENGTH_REQD</userType>
        <baseType>FAIL</baseType>
        <description>Content-length required in the request</description>
    </userDefinedExitType>
```

```
<userDefinedExitType>
   <neDescriptor>
      <softwareLoad>3-0</softwareLoad>
      <technology>OTAC</technology>
      <neVendor>INSIGNIA</neVendor>
   </neDescriptor>
   <searchPattern>412</searchPattern>
   <userType>PRECONDITION_FAIL</userType>
   <baseType>FAIL</baseType>
   <description>Precondition failed in requested header fields

   </description>

</userDefinedExitType>
<userDefinedExitType>
   <neDescriptor>
      <softwareLoad>3-0</softwareLoad>
      <technology>OTAC</technology>
      <neVendor>INSIGNIA</neVendor>
   </neDescriptor>
   <searchPattern>413</searchPattern>
   <userType>REQUEST_TOO_LARGE</userType>
   <baseType>FAIL</baseType>
   <description>Request entity is too large to process</description>
</userDefinedExitType>
<userDefinedExitType>
   <neDescriptor>
      <softwareLoad>3-0</softwareLoad>
      <technology>OTAC</technology>
      <neVendor>INSIGNIA</neVendor>
   </neDescriptor>
   <searchPattern>414</searchPattern>
   <userType>REQUEST_URL_TOO_LONG</userType>
   <baseType>FAIL</baseType>
   <description>URL length exceeded maximum size</description>
</userDefinedExitType>
<userDefinedExitType>
   <neDescriptor>
      <softwareLoad>3-0</softwareLoad>
      <technology>OTAC</technology>
      <neVendor>INSIGNIA</neVendor>
   </neDescriptor>
   <searchPattern>415</searchPattern>
   <userType>UNSUPP_MEDIA_TYPE</userType>
   <baseType>FAIL</baseType>
   <description>Request format not supported</description>
</userDefinedExitType>
<userDefinedExitType>
   <neDescriptor>
      <softwareLoad>3-0</softwareLoad>
      <technology>OTAC</technology>
      <neVendor>INSIGNIA</neVendor>
   </neDescriptor>
```

```
    <searchPattern>501</searchPattern>
    <userType>REQUEST_UNKNOWN</userType>
    <baseType>FAIL</baseType>
    <description>Function not implemented in Web server software

    </description>
</userDefinedExitType>
<userDefinedExitType>
    <neDescriptor>
        <softwareLoad>3-0</softwareLoad>
        <technology>OTAC</technology>
        <neVendor>INSIGNIA</neVendor>
    </neDescriptor>
    <searchPattern>502</searchPattern>
    <userType>BAD_GATEWAY</userType>
    <baseType>FAIL</baseType>
    <description>Invalid response from server</description>
</userDefinedExitType>
<userDefinedExitType>
    <neDescriptor>
        <softwareLoad>3-0</softwareLoad>
        <technology>OTAC</technology>
        <neVendor>INSIGNIA</neVendor>
    </neDescriptor>
    <searchPattern>503</searchPattern>
    <userType>SERVICE_UNAVAILABLE</userType>
    <baseType>FAIL</baseType>
    <description>Service under maintenance</description>
</userDefinedExitType>
<userDefinedExitType>
    <neDescriptor>
        <softwareLoad>3-0</softwareLoad>
        <technology>OTAC</technology>
        <neVendor>INSIGNIA</neVendor>
    </neDescriptor>
    <searchPattern>504</searchPattern>
    <userType>GATEWAY_TIMEOUT</userType>
    <baseType>FAIL</baseType>
    <description>Server has not responded</description>
</userDefinedExitType>
<userDefinedExitType>
    <neDescriptor>
        <softwareLoad>3-0</softwareLoad>
        <technology>OTAC</technology>
        <neVendor>INSIGNIA</neVendor>
    </neDescriptor>
    <searchPattern>505</searchPattern>
    <userType>UNSUPP_HTTP_VER</userType>
    <baseType>FAIL</baseType>
    <description>HTTP version not supported</description>
</userDefinedExitType>
<userDefinedExitType>
```

```
        <neDescriptor>
            <softwareLoad>3-0</softwareLoad>
            <technology>OTAC</technology>
            <neVendor>INSIGNIA</neVendor>
        </neDescriptor>
        <searchPattern>NUMBER_FORMAT_JAVA_EXCEPTION</searchPattern>
        <userType>NUMBER_FORMAT_EX</userType>
        <baseType>FAIL</baseType>
        <description>Number format exception</description>
    </userDefinedExitType>
    <userDefinedExitType>
        <neDescriptor>
            <softwareLoad>3-0</softwareLoad>
            <technology>OTAC</technology>
            <neVendor>INSIGNIA</neVendor>
        </neDescriptor>
        <searchPattern>HTTP_CARTRIDGE_EXCEPTION</searchPattern>
        <userType>HTTP_CART_EX</userType>
        <baseType>FAIL</baseType>
        <description>HTTP cartridge exception</description>
    </userDefinedExitType>
</serviceModel>
```

4

Service Definition

The Insignia OTAC cartridge contains a set of CSDLs that map to one or more ASDL commands. You can also create additional CSDLs that map to existing and newly-created ASDLs. An upstream system can assemble any of these CSDL commands onto a work order for provisioning.

This chapter presents detailed information on the CSDL parameters that we provide in this cartridge. The following table lists and describes the type of parameter information that is included.

Table 9: ASDL parameter information

Item	Description
Parameter Name	Identifies the parameter that is configured for the stated service.
Description	Describes the parameter.
Range	Describes or lists the range of values that can be used to satisfy this parameter.
Default Value	Configures a default value for the parameter so that it is not mandatory for the upstream system to provide a value.

Table 9: ASDL parameter information

Item	Description
Type	Indicates one of the following parameter types: ◆ S—Scalar, specifies the parameter label transmitted on the ASDL command. Scalar parameters are conventional name-value pair parameters. ◆ C—Compound, specifies the base name of the compound parameter transmitted on the ASDL command. A compound parameter contains structures or arrays of information that are represented by a particular structure name or compound parameter name. Each compound parameter can contain a large number of elements. If you use compound parameters, you only require a single entry in the ASAP translation tables to call the compound parameter and all its associated parameter elements. ◆ I—Indexed, identifies a parameter that contains a sequential numerical index value to tell the SARM that it should execute the same operation (for example, an ASDL command) for all occurrences of that index. Consequently, if there are several options on a particular CSDL command (OPT1, OPT2, OPT3, etc.), you can specify the OPT parameter as an indexed parameter. When you specify the OPT parameter as an indexed parameter, the SARM generates several occurrences of that same ASDL command and each command has a different value for the option being transmitted to the NEP. For more information on parameter types, refer to the *ASAP Developer Reference*.
Class	Indicates one of the following parameter classifications: ◆ R—Required scalar parameter ◆ O—Optional scalar parameter ◆ C—Required compound parameter ◆ N—Optional compound parameter ◆ M—Mandatory indexed parameter ◆ I—Optional indexed parameter ◆ S—Parameter count

For a detailed description of the Required and Optional parameter classifications, refer to the *ASAP Administration Guide.*

CSDL commands

This cartridge provides the following CSDL Commands:

- C_INSIGNIA-OTAC_3-0_SUBMIT_HTR
- C_INSIGNIA-OTAC_3-0_SUBMIT_SLR

C_INSIGNIA-OTAC_3-0_SUBMIT_HTR

Provisioning based on Handset Type.

Table 10: C_INSIGNIA-OTAC_3-0_SUBMIT_HTR

Parameter Name	Description	Range	Default Value	Type	Class
NE_ID_INSIGNIA-OTAC	The Insignia NE identifier			S	R
DESTINATION	Destination MSISDN number			S	R
GROUP_ID	Group ID			S	R
PHONE_ID	Phone ID			S	R
NETWORKPIN	IMSI of the handset			S	O
HOMEPAGE	Home page sent in the OTAC message			S	O
USERNAME	ISP user name			S	O
PASSWORD	ISP Password			S	O

Mapping to ASDLs

The following table illustrates the CSDL to ASDL mapping for this service.

Table 11: CSDL to ASDL Mapping

CSDL	ASDL
C_INSIGNIA-OTAC_3-0_SUBMIT_HTR	A_INSIGNIA-OTAC_3-0_SUBMIT_HTR

C_INSIGNIA-OTAC_3-0_SUBMIT_SLR

Provisioning based on Service Level.

Table 12: C_INSIGNIA-OTAC_3-0_SUBMIT_SLR

Parameter Name	Description	Range	Default Value	Type	Class
NE_ID_INSIGNIA-OTAC	The Insignia NE identifier			S	R
DESTINATION	Destination MSISDN number			S	R
SERVICELEVEL	Service level of subscriber			S	R
USERNAME	ISP user name			S	O
PASSWORD	ISP Password			S	O
HOMEPAGE	Home page sent in the OTAC message			S	O

Mapping to ASDLs

The following table illustrates the CSDL to ASDL mapping for this service.

Table 13: CSDL to ASDL Mapping

CSDL	ASDL
C_INSIGNIA-OTAC_3-0_SUBMIT_SLR	A_INSIGNIA-OTAC_3-0_SUBMIT_SLR

5

Configuring ASAP to Support Additional NE Instances

You can configure ASAP to support the NE_INSIGNIA-OTAC_3-0_HOST - NEP configuration using the Service Activation Configuration Tool (SACT). Refer to the *ASAP Administration Guide* for more information.

Below is an example of the Activation.Configuration.XML file for the Insignia OTAC cartridge.

```
<?xml version="1.0" encoding="UTF-8"?>
<activationConfig xmlns="http://www.metasolv.com/ServiceActivation/
  2003/ActivationConfig" xmlns:cfg="http://www.mslv.com/studio/acti-
  vation/model/config" xmlns:route="http://www.mslv.com/studio/acti-
  vation/model/routing" xmlns:sm="http://www.metasolv.com/
  ServiceActivation/2003/ServiceModel" xmlns:xsi="http://www.w3.org/
  2001/XMLSchema-instance">
   <connectionPool name="OTAC10">
      <device name="NE_INSIGNIA-OTAC_3-0_HOST_conn_1">
         <environment/>
         <lineType>GENERIC_MESSAGE_BASED_CONNECTION</lineType>
      </device>
   </connectionPool>
   <element name="NE_INSIGNIA-OTAC_3-0_HOST">
      <vendor>INSIGNIA</vendor>
      <technology>OTAC</technology>
      <softwareLoad>3-0</softwareLoad>
      <nepServerName>$NEP</nepServerName>
      <primaryPool>OTAC10</primaryPool>
      <maximumConnections>5</maximumConnections>
      <dropTimeout>1</dropTimeout>
      <spawnThreshold>6</spawnThreshold>
      <killThreshold>3</killThreshold>
      <routingElement name="NE_INSIGNIA-OTAC_3-0_HOST"/>
      <communicationParameter>
         <label>OTAC_URL</label>
         <value>
            <value>http://10.6.1.90/</value>
         </value>
         <description>URL to the OTAC Mediator</description>
         <lineType>GENERIC_MESSAGE_BASED_CONNECTION</lineType>
      </communicationParameter>
      <communicationParameter>
         <label>CONNECTION_TIMEOUT</label>
         <value>
            <value>20</value>
         </value>
         <description>Connection timeout</description>
         <lineType>GENERIC_MESSAGE_BASED_CONNECTION</lineType>
      </communicationParameter>
      <communicationParameter>
         <label>READ_TIMEOUT</label>
         <value>
            <value>30</value>
         </value>
         <description>Time to wait for a response from the server
         </description>
         <lineType>GENERIC_MESSAGE_BASED_CONNECTION</lineType>
      </communicationParameter>
      <communicationParameter>
         <label>RESPONSELOG</label>
         <value>
            <value>TRUE</value>
         </value>
         <description>Enable / Disable response log</description>
         <lineType>GENERIC_MESSAGE_BASED_CONNECTION</lineType>
      </communicationParameter>
   </element>
</activationConfig>
```

Extracting source files

Before you can access an XML file to modify it, you must extract it from the .sar file. Use the following procedure to extract source files from the sar file.

To extract source files

1. Create a repository directory. Copy the .sar file to the new directory and un-jar the sar file.
2. After you un-jar the sar file, you can access the XML files.

Loading a new XML file

When you finish modifying an XML file, you must create a new sar file, then restart the cartridge using the new file.

Follow the instructions in "Testing the installation" on page 10, for directions on how to load a new XML file.