Oracle® Communications Design Studio

Modeling Service Request Translations

Release 7.3

E57033-01

July 2015

Oracle Communications Design Studio Modeling Service Request Translations, Release 7.3

E57033-01

Contents

About Design Studio Help for Modeling Service Request Translations

This guide provides information about modeling service request translations for Oracle Communications ASAP.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at
`http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc`.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit
`http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit
`http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

1 Getting Started with Design Studio for ASAP SRT

As part of a customer solution, solution designers and systems integrators can create Activation Service Request Translation (SRT) cartridges to translate service request data sent to Oracle Communications ASAP from customer relationship management (CRM), provisioning control, or other upstream systems. The Activation SRT cartridge contains translation information that enables the SRT to map the contents of orders sent from upstream systems to a format that is recognizable and usable by ASAP.

Activation SRT cartridges (which you can use only with the SRT component of ASAP) are most commonly used when operation support systems (OSS) require a much higher level of data abstraction at the interface point into ASAP, or have significant data customization requirements at the interface point, such as data messaging or lookup requirements.

Related Topics

Understanding ASAP SRT Users and Tasks

Modeling Activation SRT Cartridges

Understanding ASAP SRT Users and Tasks

The following table lists the roles and the tasks each role typically performs in Oracle Communications Design Studio for ASAP SRT.

Role	Task	Reference
Solution Designer	Load (import) cartridges	See "Importing Activation Cartridge Projects" See "Importing Projects"
Solution Designer	Setting up Activation SRT cartridges	"Creating New Activation SRT Cartridges"
Solution Designer	Modeling service bundles	"Creating Service Bundles" "Associating Service Actions with the Service Bundle" "Understanding Service Action Spawning Conditions" "Understanding Upstream Interface Parameters" "Mapping Upstream Interface Parameters to Service Action Parameters" "Understanding Lookups"

Role	Task	Reference
Solution Designer	Build, deploy and undeploy to/from development environments	"Packaging and Deploying ASAP SRT Cartridges"
Developer	Integration with the CRM (upstream) system	"Modeling Translations"
Developer	Configuring Lookups	"Configuring Lookups"

Related Topics

Understanding ASAP SRT Cartridges

Activation SRT Project Editor

Getting Started with Design Studio for ASAP SRT

Modeling Activation SRT Cartridges

Following this procedure and refer to the corresponding topics to model Activation SRT cartridges:

To model Activation SRT cartridges:

1. Import Activation Service cartridges from a cartridge project to obtain service actions that you can use for the SRT cartridge.

   See "Importing Projects" for more information.

2. Create a customer specific service model.

   In most cases you create a common service model based on the imported cartridges. See "About Common Service Models" for more information.

3. Create an Activation SRT cartridge project and set up the Activation SRT cartridge.

   a. Use the Activation Cartridge Project wizard to create an Activation SRT cartridge project and display it in the Studio Projects view of the Design perspective.

      See "Creating New Activation SRT Cartridges" for more information.

   b. Configure the cartridge details in the Activation SRT Project editor to define the content.

      See "Activation SRT Project Editor" for more information.

4. Create one or more translations.

   You configure translations to enable the SRT to accept and recognize messages from upstream systems at runtime. See "Modeling Translations" for more information.

5. Model service bundles to enable the SRT to execute actions (service actions and atomic actions) based on the content of translated messages from upstream systems.

   a. Create service bundles to map information from the incoming (translated) messages from upstream to the appropriate service actions that need to be executed downstream (against the network elements).

      See "Creating Service Bundles" for more information.

b. Associate service actions with the service bundle.

   Select the service actions from the customer specific service model that need to execute for the service bundle. See "Associating Service Actions with the Service Bundle" for more information.

c. Define service action spawning conditions (if required) that allow service actions to be conditionally spawned.

   See "Understanding Service Action Spawning Conditions" for more information.

d. Add upstream interface parameters that describe the parameters that are expected from the upstream system for this service bundle.

   See "Understanding Upstream Interface Parameters" for more information.

e. Map upstream interface parameters to service action parameters that allow the SRT to map the incoming data from the translated order to the service actions selected to run for the service bundle.

   See "Mapping Upstream Interface Parameters to Service Action Parameters" for more information.

f. Use lookups (if any were configured).

   See "Understanding Lookups" for more information.

Note: Data required by the service bundle that is not provided on the order from the upstream system is obtained by configuring lookups. Lookups can also be used to format parameters (for example, to split parameters apart into constituent parameters, or to join parameters together into a single parameter).

See "Configuring Lookups" for more information.

6. Package the cartridge.

   Use the SRT Project editor to specify which elements will be included in the cartridge SAR file. See "Packaging Activation SRT Cartridges" for more information.

   a. Create the JAR with ANT.
   b. Include JARs in the SAR.
   c. Put external JARs in the NEP classpath on the ASAP server.

7. Deploy the cartridge.

   See "Deploying Cartridge Projects" for more information.

   a. Create a Studio Environment project and a Studio Environment in the Studio Projects view (use corresponding wizards for both the tasks).
   b. On the **Connection Information** tab of the Studio Environment editor, specify how to connect to the activation environment.
   c. In the Cartridge Management view, add cartridges for deployment, deploy them to the run-time environment, undeploy them from the run-time environment, and remove them from the list of cartridges that were added for deployment.
   d. Use the NEP Map editor to deploy and manage network elements.

Related Topics

Getting Started with Design Studio for ASAP SRT

2

Creating ASAP SRT Cartridge Projects

You can create Activation Service Request Translation (SRT) cartridges to enable the SRT to map the contents of orders sent from upstream systems to a format that is recognizable and usable by Oracle Communications ASAP. Refer to the following topics when creating ASAP SRT cartridges:

- Understanding ASAP SRT Cartridges
- Creating New Activation SRT Cartridges
- Importing Activation SRT Cartridges from SAR Files
- Activation SRT Project Editor

Related Topics

Exporting Projects

Importing Projects

About Cartridge Project Upgrades

Packaging and Deploying ASAP SRT Cartridges

Understanding ASAP SRT Cartridges

When creating Activation SRT cartridges, you assemble service actions from an Activation Service cartridge (or, in less common scenarios, service actions from an Activation Network cartridge) into a meaningful group using service bundles. Service bundle are collections of service actions required to implement marketed products.

Related Topics

Configuring Service Bundles

Example: Configuring Service Bundles

Creating ASAP SRT Cartridge Projects

Configuring Service Bundles

There are two steps to configuring service bundles in ASAP, each corresponding to a separate layer of run time translation that is performed by ASAP.

To configure service bundles:

1. Write an XSL translation (XSLT) that accepts an XML document from an upstream system and outputs an XML document, consisting of name value pairs, that is used by the second layer of translation.

When complete, the XSLT is placed in a Oracle Communications Design Studio library in a location that ensures it is deployed by Design Studio to an ASAP environment (and possibly source controlled). The translation enables usage of the service bundles that are configured in step two. The SRT translates order data received from upstream systems into a format recognizable by downstream components of ASAP. The SRT also translates responses from ASAP back into a format recognizable by the upstream systems. Developers may need to create and configure lookups to obtain additional data required to activate the service, but unavailable from upstream systems. Developers may also use lookups to convert specific data elements into a format that is expected by the configuration further downstream.

2. Import one or more Activation Service cartridges into Design Studio, create a customer-specific service model, set up the Activation SRT cartridge, and model the required service bundles by mapping the upstream system order data to the service actions selected for use in the service bundles.

   This step assumes that you have written the XSLT and can import upstream parameters into Design Studio, or that you manually configured upstream parameters and mapped them to service action parameters. See "Understanding Upstream Interface Parameters" and "Modeling Service Bundles" for more information.

Related Topics

Understanding ASAP SRT Cartridges

Creating ASAP SRT Cartridge Projects

Modeling Translations

Example: Configuring Service Bundles

Consider the following run time example:

A telephone company is marketing a new service that includes a land line and a cell phone and sends a request to activate this service to ASAP. An XML order is received by the SRT component of ASAP, where it is first translated into a format that enables the SRT to analyze the order contents. Next, the data within the order is mapped to the service actions and parameters that are required to activate the service bundle. This may involve executing lookups to derive or massage additional data required for activation. The order is then sent to downstream components of ASAP for activation. During the execution of the order, events are generated and passed upstream to the SRT component. The SRT component translates these events to a form recognized by upstream systems and forwards them to the upstream systems.

Related Topics

Understanding ASAP SRT Cartridges

Creating ASAP SRT Cartridge Projects

Modeling Translations

Creating New Activation SRT Cartridges

You use the Activation SRT Cartridge Project wizard to set up an Activation SRT cartridge project. After you create a new project, you configure additional cartridge details in the Activation SRT Project editor.

To create an Activation SRT Cartridge project:

1. Select **Studio**, then select **Show Design Perspective**.
2. In the Studio Design perspective, right-click in the Studio Projects view and select **New**, select **Project**, then select **Activation SRT Project**.

   Alternatively, select **Studio**, select **New**, select **Project,** then select **Activation SRT Project**.The New Studio Activation SRT Cartridge Project wizard appears displaying the Activation SRT Cartridge Info dialog box.
3. Enter a name for the project.
4. Accept the default location or browse for another location.
5. Select the appropriate target version.
6. (Optional) Click **Next** to change the Java configuration in the Java Settings dialog box. For example, you can add libraries in the **Libraries** tab (the configuration can also be changed later).
7. Click **Finish** to complete the cartridge project.

   In the Studio Projects view, a new Activation SRT Cartridge project appears. The project contains one entity, which represents the service cartridge.
8. Configure the SRT Cartridge specifications and parameters in the tabs of the Project editor.

   In the Studio Projects view, double-click an Project entity icon to display the Project editor.

Related Topics

Creating ASAP SRT Cartridge Projects

Modeling Translations

Modeling Service Bundles

Configuring Lookups

Importing Activation SRT Cartridges from SAR Files

Activation SRT Project Editor

Packaging and Deploying ASAP SRT Cartridges

Importing Activation SRT Cartridges from SAR Files

To import Activation SRT cartridges into Design Studio, you need to import the SRT cartridge and the Activation Network or Activation Service cartridge upon which the SRT cartridge depends (for service actions). When you locate the SRT cartridge and appropriate Activation cartridge that you need to import, they may be combined in a single SAR file or packaged individually as separate SAR files. In both cases, importing is a two-stage process:

1. Import the Activation Network or Service cartridge.
2. Import the Activation SRT cartridge (it is recommended that you import in this order, although the reverse order is possible).

Note: When importing an SRT cartridge, error messages may appear indicating that some service bundles are not synchronous with the service actions that they reference. This occurs when some service action parameters have not been mapped in the service bundle. If this occurs, you can synchronize the parameters from the service bundle editor.See "Mapping Upstream Interface Parameters to Service Action Parameters" for more information.

The following procedure describes how to import SRT cartridges from a SAR file, using two example SAR files:

- **both.sar**: In this example, this SAR file contains both an Activation SRT cartridge and an Activation Network or Service cartridge required for the SRT cartridge. The two-stage import sequence is illustrated by importing cartridges from this file.
- **nosrt.sar**: In this example, this SAR file contains only an Activation Network or Service cartridge required for an SRT cartridge. An explanation is provided on how the two-stage import sequence would progress by importing the cartridge from this file.

To import an Activation SRT cartridge from a SAR file:

1. In the Studio Design perspective, right-click in the Studio Projects view and click **Import Activation Archive**.

   Alternatively, select **File**, select I**mport** to display the Import-Select dialog box, then select **Studio Wizard**, select **Activation Archive (SAR)** and click **Next**. The Activation Archive Import Wizard appears.

2. In the Activation Archive Import Wizard, click **Browse**.
3. Search for the SAR file that contains both the desired Activation SRT cartridge and Activation Network or Service cartridge.

   In this example, you would search for **both.sar** file.

   Alternatively, when applicable, search for the SAR file that contains only the Activation Network or Service cartridge that is required for an SRT cartridge in a separate SAR file. In this example, you would search for the **nosrt.sar** file.

4. Click **Open**.

   The wizard now provides a list to select either the Activation Cartridge project (Network or Service cartridge) or the Activation SRT cartridge project contained in the SAR file. Select the Activation Cartridge project for the first import of the two-stage process.

5. Click **Next**.

   On the **Cartridge Details** tab, select or enter the appropriate vendor, technology, and software load.

6. Click **Finish**.

   In the Studio Projects view, the Activation Cartridge project appears with its Project entity.

7. For the second stage of the process, repeat steps 1 through 3.

   This time, when you click **Open**, select the Activation SRT Cartridge project from the list.

Note: Note that *-SRT* will be appended to the project name in the **To project** field.

8. Click **Finish**.

   In the Studio Projects view, the Activation SRT Cartridge project appears with its Project entity. The previously imported Activation Cartridge and the Activation SRT Cartridge, both are now displayed as sealed cartridges.

   Note: Starting at step 2, the same two-stage import procedure could be repeated with cartridges packaged as separate SAR files. First, you would import the Activation cartridge from the **nosrt.sar** file, and then import the Activation SRT cartridge from the **both.sar** file (assuming it were dependent on the Activation cartridge of the **nosrt.sar** file. Otherwise, a dependent Activation SRT cartridge could be imported from an appropriate SAR file.

See "Importing Projects" for more information about sealed and unsealed status, read only status, and previous releases.

Related Topics

Creating ASAP SRT Cartridge Projects

Understanding ASAP SRT Cartridges

Creating New Activation SRT Cartridges

Activation SRT Project Editor

Packaging and Deploying ASAP SRT Cartridges

Importing Projects

Importing Cartridges from Environments

Activation SRT Project Editor

Use the Activation SRT Project editor to configure the Activation SRT cartridge. In the Studio Projects view, double-click the Project entity to open the Activation SRT Project editor.

Notes:

- Network elements and environments for activation are not defined inside an SRT cartridge project. Only items that get bundled for delivery are defined.
- An Activation SRT cartridge project is also a Java project (builds on functionality of Java project). A separate Java project for development is not required.
- Eclipse online documentation for a Java project and its configurations, properties, and settings also applies to the Java configuration of a service cartridge project.

When working with the Activation SRT Project editor, see the following topics:

- Activation SRT Project Editor Properties Tab
- Project Editor Copyright Tab
- Project Editor Dependency Tab
- Activation SRT Project Editor Packaging Tab
- Activation SRT Project Editor Locations
- Activation SRT Project Editor Blueprint Tab

Activation SRT Project Editor Properties Tab

Use the **Properties** tab to configure the cartridge properties.

Field	Use
Description	Specify a name for the project as it should appear in Design Studio and in the run-time environment.
Provider	Specify a name or description for the project to help identify the project within the Design Studio environment. If you have purchased a cartridge project from a third party, this field will contain the name of the third-party provider. For cartridges purchased from Oracle, for example, this field displays Oracle Corporation.
Identifier	Specify a database name for the project.
Major Version Number, Minor Version Number, Maintenance Pack, Generic Patch, and **Customer Patch**	Define values to create a 5-segment release version numbering scheme. You can define release version numbers for any unsealed and deployable Design Studio cartridge project. Cartridge projects must always have a valid version number. When you create a cartridge project for the first time, Design Studio applies the following default values: ■ **Major Version Number:** 1 ■ **Minor Version Number:** 0 ■ **Maintenance Pack:** 0 ■ **Generic Patch/Customer Patch:** 0 You can edit the values in these fields to create new project instances in the run-time environment. When you edit any of these fields, you create a new instance of the corresponding project when you deploy in a run-time environment. You can create new instances of a project when you want to deploy a new version of a project while keeping the older version in the run-time environment. For example, when you make changes to a cartridge project, you can increment the version number and deploy the instance of the project to the run-time environment, while retaining the older version in the environment. **Note:** Modifying these field values does not create a separate instance of the project in Design Studio. When changing version numbers, Oracle recommends that you use a source control system to ensure that you are able to return to the previous version. **Important:** Design Studio cannot support multiple versions of a project in the same workspace. Multiple versions of a project in the same workspace create conflicting model entities.

Field	Use
Build Number	Indicates which version of the metadata is used by the corresponding project. If you have enabled the automatic build feature, Design Studio increases the build number automatically every time you save. To enable the automatic build feature, from the **Project** menu, select **Build Automatically**.
Target Version	Select the version of the run-time application instance to which you want to deploy the cartridge project. Design Studio builds your project to be compatible with the run-time software version you select from this list. Select the highest version number that is equal to or less than the version of the run-time software to which you want to deploy the project. For example, if you are deploying to release 7.2, select the highest version number that is equal to or less than version 7.2. **Note:** The list appears only for cartridge projects that are deployable to run-time environments. When you select a new value from the list, Design Studio automatically initiates a new build. Some entity configurations may no longer be valid for the new application version.
State	Click any one of the following: ▪ **Seal:** Prevents changes to the project. For example, you may seal a project after the cartridge design is complete, debugged, and tested to prevent users who import the project from rebuilding or overwriting the original build artifacts. ▪ **Unsealed:** If you want to make changes to the project, then rebuild it to obtain a new archive file. **Note:** To modify files defined as read only, you must edit the entity read-write properties. See "Defining Entity Read-Only Properties" for more information.

Related Topics

Activation SRT Project Editor

Creating ASAP SRT Cartridge Projects

Activation SRT Project Editor Packaging Tab

Use the **Packaging** tab to specify what entities will be included in the cartridge SAR file.

Field	Use
Include all from Project	Select to include all entities from a specific resource. For example, if you want to include all of your Java libraries, select Java Libraries from the left-side column, then select **Include all from Project** check box. The system will include all libraries in the package file.
Select	Click to add an entity in the package file.
Open	Click to open a specific entity.
Remove	Click to clear a specific entity from the package file.

Related Topics

Activation SRT Project Editor

Packaging Activation SRT Cartridges

Activation SRT Project Editor Locations

Use the **Locations** tab to display where items get stored. This tab is not configurable.

Note: The Default Implementation Package name is used as a prefix for the generated code. You should accept default values and follow recommended naming conventions for anything you create.

Related Topics

Activation SRT Project Editor

Activation SRT Project Editor Blueprint Tab

Use the **Blueprint** tab to view the generated documentation of the project, including cartridge properties and SRT model configuration. This tab is read only.

Related Topics

Activation SRT Project Editor

3

Modeling ASAP SRT Implementations

When modeling an Oracle Communications ASAP Service Request Translation (SRT) implementation, you can include lookups in the service bundles that you use for an Activation SRT cartridge.

Note: It is recommended that a technical resource such as a developer implement lookups.

Related Topics

Configuring Lookups

Lookup Editor

Configuring Lookups

Lookups are executed by the SRT during the processing of an upstream XML service request. Lookups can be used to look up additional parameters that are required for activation of services, to format parameters so that they are compatible with service action parameters, and so forth. The Lookup library contains custom lookups that are used by the SRT only.

To create new lookups:

1. Select **Studio**, then select **Show Design Perspective**.
2. In the Studio Projects view, right-click and select **New**, select **Activation**, then select **Lookup**.

   Alternatively, select **Studio**, select **New**, select **Activation**, then select **Lookup**. The Studio Model Entity wizard appears.
3. Select the applicable project from the list and enter a name for the lookup.
4. Click **Browse** to open a Select Location dialog box and select a location from a list.

   This populates the **Location** field and specifies the folder under the cartridge project where the new lookup will be located. If no folder exists, type in the location name, which creates a new folder with that name.

   Note: Specifying a location is optional, but is recommended to organize your elements and to avoid locating elements directly below the root folder.

   ---
5. Click **Finish**.

The new lookup appears under your project folder in Studio view.

Use the Lookup editor to configure the lookup.

6. Double-click the lookup entity.

   The Lookup editor opens.

7. From the Type list, select a lookup type.
8. If you select type as JAR, see "Selecting Lookup Class for JAR format" for more information.
9. In the Lookup Properties area, define the properties for the lookup.
10. In the **Input Parameters** tab, click **Add**.

    An input parameter is added.

11. Click the specific row and configure in the Input Parameter Details area.
12. In the **Output Parameters** tab, click **Add**.

    An output parameter is added.

13. Click the specific row and configure in the Output Parameter Details area.
14. Select **File**, then select **Save**.

Related Topics

Lookup Editor

Example: Configuring Lookups

Modeling ASAP SRT Implementations

Example: Configuring Lookups

If you have selected ASAP version 5.2 and have defined lookups with input parameters that map to other lookups, an additional source attribute is generated in the lookup model xml.

Precondition: SRT project is created for Activation 5.2.

Precondition: Two lookups are created in the same cartridge (for example LK1, LK2)

1. Create a lookup.

   For example, LK3.

2. Add an input parameter (lookup type).
3. Map to another lookup.

   For example, map to LK1.

4. Save the lookup.
5. From the Package Explorer view, cartridgeBuild/model, examine the lookup.xml.

   For example, LK3.xml: A new attribute has been added called `sourceDocument=`

Consider the following code example, which illustrates the previous stepped procedure:

```
<inputParameter>
<parameterName>inParm4</parameterName>
<lookupParameterName> outParm1</lookupParameterName>
</inputParameter>
```

would become

```
<inputParameter>
<parameterName>inParm1-lk3</parameterName>
<lookupParameterName sourceDocument="LK1">outParm1-lk1</lookupParameterName>
</inputParameter>
```

Related Topics

Lookup Editor

Modeling ASAP SRT Implementations

Selecting Lookup Class for JAR format

To select lookup class in the jar file that performs the lookup:

1. On the Lookup editor, click **Select** to open Select Lookup Class dialog box.
2. Do any one of the following:
   a. In the **Select entries** field, enter the name of a class which exists in a **.jar** file referenced in the classpath.
   b. Enter any character or string of characters contained in the class name. Matching class names appear in the **Matching items** area of the dialog box.
3. Select a class name in the dialog box and click **OK** to populate the class name in the **Class** field.

Related Topics

Lookup Editor

Lookup Editor

Use the Lookup editor to configure lookups. In the Studio Projects view, double-click a lookup entity to open the Lookup editor.

When working with the Lookup editor, see the following topics:

- Lookup Editor Editor Tab
- Lookup Editor Blueprint Tab

Lookup Editor Editor Tab

Use the **Editor** tab for defining the input parameters to the lookup and output parameters produced by the lookup.

Lookup and Lookup Properties

Field	Use
Description	Specify a description for the Lookup editor.

Field	Use
Type	Select one of the following to specify the format of the lookup: ■ **JAR** This option requires no parameters as the lookup logic is in the lookup class selected in the **Class** field. ■ **SQL** This option requires the parameters **jdbc:sqlStatement** and **jdbc:dataSource**. ■ **javascript** This option requires the parameters **bsf:scriptEngine** and **bsf:script**. ■ **webservice** This option requires the parameters **WebServiceEndPoint**, **WebServiceName**, **TargetNamesapce**, **NSPrefix**, **RequestOperationName**, **RequestMessage**, **ResponseOperationName**, and **ResponseMessage**. ■ **xpath** This option requires the parameter **xpath:function**. **Notes:** ■ The specified file type should have the same file name as the lookup and its filename extension must be **.js** or **.sql**, depending on the type of file (javascript or SQL). ■ The specified file must be packaged in the cartridge.
Class	If you selected **JAR** in the **Type** field, select the class in the jar file that performs the lookup. See "Selecting Lookup Class for JAR format" for more information. ASAP will run the `main` method within the specified class when the lookup is called.
No Caching	Select if the results of the lookup are not to be cached in the memory. Use this option if the data is variable, and therefore no benefit is derived from caching in memory and retrieving (but is slower than cached information).
Session Caching	Select if the results of the lookup are to be cached in the memory for the duration of the session (should the results be needed again by another lookup or service bundle). A session is the time it takes to fulfill the upstream request. Use this option if multiple clients with different information want to share a cache.
Global Caching	Select if the results of the lookup will remain available until the cache timeout expires.
Cache Max Size (entry)	Specify the maximum number of actual entries in the cache that will be maintained at any one time. In this field, you can enter a maximum value of eight numeric characters. This field is available for **Session Caching** and **Global Caching**.
Cache Timeout (ms)	Specify the number of milliseconds for which the cache is valid. In this field, you can enter a maximum value of eight numeric characters. This field is available for **Session Caching** and **Global Caching**.

Lookup Details

Field	Use
Add	In the **Input Parameters** tab, click to add input parameters to the lookup. The parameters you specify here could represent parameters being passed in from the upstream on the work order, or parameters that have been generated by another lookup, or both.
Name	In the **Input Parameters** tab, click on a parameter and edit in the Input Parameter Details area, as required, the name of the parameter.
Mapping Type	In the **Input Parameters** tab, click on a parameter and select any one of the following from the list: ■ Select **Environment Setting** to enable you to configure an environment variable as an input to the lookup. You may also specify a default value to assign to the variable if the environment variable is not actually defined at runtime. **Note:** The **Use Environment ID** check box is not applicable to ASAP. ■ Select **Input Parameter** to specify that a parameter will be provided to the lookup, as specified in the **Upstream Interface Parameter** field, from the incoming transformed XML document. ■ Select **Lookup** to specify that a parameter will be provided to the lookup from the output of another lookup that executes prior to this one. You must select the **Lookup** name and **Lookup Output Parameter** as follows: - **Lookup:** Name of another lookup in the same project. - **Lookup Output Parameter:** Another lookup's output parameter which will be an input to this lookup.
Add	In the **Output Parameters** tab, click to specify the new output parameters produced by the lookup.
Name	In the **Output Parameters** tab, click on a parameter and edit in the Output Parameter Details area, as required, the name of the parameter.
XPath	In the **Output Parameters** tab, click on a parameter and specify the xpath, in the Output Parameter Details area, associated with the parameter created by the lookup (all Design Studio lookups return an XML document. The xpath is used to specify the location of the desired parameter within the XML document). An example of a simple XPath would be **//MCLII**.
Description	In the **Output Parameters** tab, click on a parameter and enter a description of the parameter in the Output Parameter Details area.

Related Topics

Lookup Editor

Modeling ASAP SRT Implementations

Lookup Editor Blueprint Tab

Use the **Blueprint** tab to view the generated documentation for the lookup entity. This tab is read only.

Related Topics

Lookup Editor

Modeling ASAP SRT Implementations

4 Modeling Service Bundles

A service bundle is a grouping of one or more service actions, and are required only if you are using the Oracle Communications ASAP SRT component. Refer to the following topics when modeling service bundles:

Understanding Service Bundles

Service bundles provide a level of abstraction over the service action layer that you can use to represent marketing-level configurations. You can use service bundles when upstream systems are not capable of breaking down marketing bundles into their constituent services or providing the exact data that is required for activation.

For example, a wireless service provider may offer a basic marketing package that provides the following services:

- Voice calls
- Free voice mail
- Free call waiting and call forwarding

These services require the following activations in the network:

- Create a new subscriber (Home Location Register - HLR).
- Enable authentication of the subscriber (Authentication Center - AUC).
- Enable number portability (Flexible Number Router - FNR).
- Create and assign a voice mailbox (Voice mail Server - VMS).
- Activate the call waiting and call forwarding features (Home Location Register - HLR).

The service provider may also offer a more expensive marketing package that builds upon the basic marketing package by providing a few more advanced services, which would also require the activation of advanced features (Home Location Register - HLR).

While it is possible to model a single service action to activate each of the network services, introducing an additional level of abstraction at the service bundle level results in a less complicated and less customized service model at the service action level. Additionally, more generic service actions permit reuse of the service actions across service bundles. The following service bundle configuration could be created to implement the above examples:

Note: In the configurations below, the `C_ADD_SUBSCRIBER, C_VMS_ADD_SUB, and C_HLR_ADD_BASIC-FEATURES` service actions can be reused in both the basic and advanced service bundle configurations.

```
SB_WIRELESS_PREPAID_ADD_SUBSCRIBER_BASIC
   C_HLR_ADD_SUB
      A_HLR_ADD_SUB
      A_AUC_ADD_SUB
      A_FNR_ADD_SUB
   C_VMS_ADD_SUB
      A_VMS_ADD_SUB
   C_HLR_ADD_BASIC-FEATURES
      A_HLR_ADD_CW
      A_HLR_ADD_CF

SB_WIRELESS_PREPAID_ADD_SUBSCRIBER_ADVANCED
   C_HLR_ADD_SUB
      A_HLR_ADD_SUB
      A_AUC_ADD_SUB
      A_FNR_ADD_SUB
   C_VMS_ADD_SUB
      A_VMS_ADD_SUB
   C_HLR_ADD_BASIC-FEATURES
      A_HLR_ADD_CW
      A_HLR_ADD_CF
   C_HLR_ADD_ADVANCED-FEATURES
      A_HLR_ADD_CFB
      A_HLR_ADD_CFNRC
      A_HLR_ADD_CFNRY
```

Related Topics

Creating Service Bundles

Associating Service Actions with the Service Bundle

Understanding Service Action Spawning Conditions

Understanding Upstream Interface Parameters

Mapping Upstream Interface Parameters to Service Action Parameters

Understanding Lookups

Creating Service Bundles

After you set up the Activation SRT cartridge, you create service bundles. To enable usage of the service bundles for implementation and deployment of the SRT cartridge, ensure that the upstream system is integrated (the translation is configured by a developer) prior to deploying the cartridge to the environment.

To create a service bundle:

1. Select **Studio**, then select **Show Design Perspective**.
2. In the Studio Projects view, right-click and select **New**, select **Activation**, then select **Service Bundle**.

   Alternatively, select **Studio**, select **New**, select **Activation**, then select **Service Bundle**. The Service Bundle Wizard appears.
3. Select the correct Activation SRT cartridge project for this element and enter a name for the entity.
4. Click **Browse** to open a Select Location dialog box and select a location from a list.

   This populates the **Location** field and specifies the folder under the cartridge project where the new service bundle will be located. If no folder exists, type in the location name, which creates a new folder with that name.

   Note: Specifying a location is optional, but is recommended to organize your elements and to avoid locating elements directly below the root folder.
5. Click **Finish** to create the service bundle.

   The Service Bundle editor appears.
6. In the **Description** field, enter a description for the service bundle.
7. In the Service Bundle Properties area, add a product label and product value.

   These values map the upstream order (which contains the label and value) to this particular service bundle. At run time, this mapping enables the SRT to determine the service bundle to execute based on the incoming XML document.
8. Enable **Include order data in response** to pass a more comprehensive response back upstream (beyond just the simple success/failure response) for this service bundle.

   For example, if this service bundle is selected based on the service bundle properties criteria and the resulting ASAP order succeeds, then any response information associated with the ASAP work order will be retrieved automatically and will be available to be passed upstream.

Related Topics

Understanding Service Bundles

Associating Service Actions with the Service Bundle

Understanding Service Action Spawning Conditions

Understanding Upstream Interface Parameters

Mapping Upstream Interface Parameters to Service Action Parameters

Understanding Lookups

Configuring Translations

Associating Service Actions with the Service Bundle

After you have defined a service bundle, you can associate service actions with the service bundle.

To associate service actions with the service bundle:

1. In the Studio Projects view, double-click a Service Bundle entity.

   The Service Bundle editor opens.

2. On the **Service Actions** tab, click **Add** to open the Service Action Selection dialog box.

   Using the Service Action Selection dialog box, you can either select or create a service action entity.

3. Do the following to select a service action entity:

   a. On the Service Action Selection dialog box, select a service action.

   b. Click **OK**.

4. Repeat the previous step to select additional service actions.

5. Do the following to create a service action entity:

   a. Click **New** to create a service action entity.

      The Service Action Wizard opens.

   b. Enter an action or select a previously defined action from the list (for example, ADD, MOD, DEL, or QUERY).

   c. Enter a name for the entity (for example, SUBSCRIBER, GSM-SUBSCRIBER, ROUTE, TRUNK, or LINE).

   d. (Optionally) Deselect the **Use recommended name and location** to manually edit the name of the entity and browse for a location.

   e. Click **Finish**.

      In the Studio Projects view, the new service action entity appears in the **Service Actions** folder.

6. Repeat the previous step to create additional service actions.

7. In the Service Bundle editor, locate one or more service actions.

   You can reorder the service actions by highlighting the service actions and using up and down arrows in the editor.

   Additionally, you can manually enter interface parameters or import them from a sample upstream XML document. See "Mapping Upstream Interface Parameters to Service Action Parameters" and "Importing Upstream Parameters" for more information.

8. When prompted to create upstream interface parameters based on service action labels, select **Yes** to automatically create upstream interface parameters based on the service action labels of the service actions that you have selected.

   The parameters appear in the **Upstream Interface** tab. A mapping between each upstream parameter and each service action parameter is created automatically.

9. Navigate to the **SA Parameter Map** tab to view service action parameters. You can map the service actions to service bundles.

This feature is useful if you do not have sufficient information on the upstream parameters (such as the exact parameter labels that are provided on the incoming XML). This feature assumes that the parameter labels required by the service action are the same as those on the upstream order. The selected service actions appear in the Service Bundle Details area of the Service Bundle editor with their associated attributes (sequence, service action name, condition, label, value, expression, vendor, technology, software load and action).

Related Topics

Understanding Service Bundles

Creating Service Bundles

Understanding Service Action Spawning Conditions

Understanding Upstream Interface Parameters

Mapping Upstream Interface Parameters to Service Action Parameters

Understanding Lookups

Understanding Service Action Spawning Conditions

Service action spawning conditions govern whether a service action executes. When you map service bundles to service actions, by default the service actions are assigned a spawning condition of *Always*, meaning that no conditions will prevent them from running. The conditions associated with the service action must evaluate to *True* for the service action to be spawned.

The labels used in the evaluation of the service action spawning condition are those that are derived as a result of the upstream parameters to service action parameter mapping. Use the service action labels in your spawning expression.

On the Service Bundle editor **Service Actions** tab, you can define the following spawning conditions for service actions:

- **Always**: the service action is always spawned for this service bundle.
- **Equals**: the service action is spawned only if the specified parameter has a particular value.
- **Defined**: the service action is spawned only if the specified parameter is present.
- **Not Defined**: the service action is spawned only if the specified parameter is not present.

You can optionally define a logical expression that is used with one of the delivered spawning conditions, where ASAP will execute the service action if both the out-of-the-box expression and your user-defined expression evaluate to true. You can define a logical expression using a number of criteria. The range of options available allows a service action to be executed if the service action parameter value is within a set range of values, or if the service action parameter is greater than, less than, or equal to, a specified value. You can combine multiple expressions using an AND or OR operator.

The following sections describe how to formulate logical expressions.

Logical Expression Components

If more complicated spawning logic is required (that is, the out-of-the-box spawning capability is not sufficient), enable the **Include Expression** check box on the Service

Bundle editor **Service Actions** tab and define a logical expression in the associated text box.

The logical expression mechanism works in cooperation with one of the out-of-the-box spawning expressions. For example, if you have specified a service action condition of **Equals** and have created a logical expression, both of these conditions must evaluate to true in order for the service action to be executed. If you want only the logical expression to trigger the execution of the service action, then use the action condition of **Always** and specify a logical expression.

The following operators can be used to define logical expressions:

Operators	Description
AND, OR, NOT	Operators used to create compound expressions. For example: `(KEY < 7214) AND (PORT > 4000)`
ISDEF, NOTDEF	Operators that can be used with single parameters. For example: ■ `(NOTDEF KEY)` ■ `(ISDEF IMSI) AND (NOTDEF IMSI)` **Note:** Always use brackets with these operators.
>, <, >=, =<, =, !=	The parameter values must be able to be converted to integers. For example: `(KI > 10)`
LIKE, !LIKE	String operators. The strings being evaluated must be placed inside quotation marks. It is also possible to use wildcard syntax such as the % (to specify one or more characters) and ? (acts on a single character). For example: ■ `(NE_ID !LIKE "HLR-2727")` ■ `(TECH LIKE "CS?K") AND (SFTWR LIKE "SNO%")` **Note:** Always use brackets with these operators.

Logical expressions are limited to a length of 255 characters.

Logical Expression Operational Order

To specify the order of operations, use as many parentheses as needed; for example:

```
((A < 8) OR ((NOTDEF B) AND ((C != 3) OR (NOT (D = 9)))))
```

Logical Expression Error Conditions

There are two error conditions that may arise as a result of the use of logical expressions:

- Using the integer operator when the input parameter cannot be converted to an integer (all SRT parameters are strings initially)
- The specified parameter label is not present

In such cases, the evaluation of an expression will fail and a SYS_ERR message will be generated in the diagnostics. However, this error does not fail the work order but simply results in the exclusion of this particular atomic action from being spawned.

Note: The syntax of complex expressions is not currently checked by Oracle Communications Design Studio but is checked by ASAP at run time.

Related Topics

Defining Service Action Spawning Logic

Understanding Service Bundles

Creating Service Bundles

Associating Service Actions with the Service Bundle

Understanding Upstream Interface Parameters

Mapping Upstream Interface Parameters to Service Action Parameters

Understanding Lookups

Defining Service Action Spawning Logic

In the Service Bundle editor **Service Actions** tab, select the service action for which you want to define spawning logic.

Define the conditions as required by selecting an out-of-the-box condition (**Always, Equals, Defined, Not Defined**). Optionally, you can also define your own logical expression by selecting **Include Expression** and providing an expression. See "Understanding Service Action Spawning Conditions" for more information.

Related Topics

Understanding Service Action Spawning Conditions

Understanding Upstream Interface Parameters

After the initial translation has occurred, the contents of messages sent from upstream systems are analyzed by the SRT at run time. After initial translation, the service bundle configuration enables the SRT to analyze the contents of messages from upstream. To accomplish this, you map the parameters that the messages contain to those required by the service actions that have been associated with the service bundle.

> **Note:** Some upstream interface parameters may not be used in the provisioning process. Upstream interface parameters can be mapped to lookups for further processing, for example to split parameters apart or to concatenate them together.

To map upstream interface parameters to service action parameters, you add the parameters to the Upstream Interface list:

- Automatically based on service action labels. When you associate service actions with the service bundle, Design Studio asks you whether you want to use the service action labels to populate the upstream interface parameter list. If you elect to do so, Design Studio places the parameters in the parameter list on the Service Bundle editor **Upstream Interface** tab.
- Manually in the Service Bundle editor. If you know what the upstream labels will be but do not have a sample XML document that can be imported, the parameters can be added in manually.
- Import the upstream parameters from a sample upstream service request (in XML format). Using this method, Design Studio runs a transformation against the upstream XML file and extracts the interface parameters. The XSLT that

implements the transformation must first be created by a technical resource such as a developer. See "Configuring Translations" for more information.

Related Topics

Defining Upstream Interface Parameters Manually

Importing Upstream Parameters

Service Bundle Editor Editor Tab

Understanding Service Bundles

Creating Service Bundles

Associating Service Actions with the Service Bundle

Understanding Service Action Spawning Conditions

Understanding Upstream Interface Parameters

Mapping Upstream Interface Parameters to Service Action Parameters

Understanding Lookups

Defining Upstream Interface Parameters Manually

In the Service Bundle editor you can manually add upstream interface parameters.

To define upstream interface parameters manually:

1. On the Service Bundle editor, click the **Upstream Interface** tab, and then click **Add** in the Service Bundle Details area.
2. In the **Upstream Interface Details** tab, enter the following information:
   - In the **Name** field, enter the name of the upstream parameter.
   - In the **Default** field, enter the default value of the upstream parameter. If the upstream parameter does not have a value, the default is supplied to ASAP.
   - In the **Data Type** field, enter the parameter's data type.
   - Select **Multi Instance** if the specified label is the stem of a series of parameters. This allows for a dynamic number of instances and possible use with ASAP's indexing capability for spawning multiple atomic actions.
   - Enable **Required** if this upstream parameter is required.

Related Topics

Understanding Upstream Interface Parameters

Importing Upstream Parameters

In the Service Bundle editor you can import upstream parameters from a sample upstream service request.

To import upstream parameters from a sample upstream service request:

1. On the Service Bundle editor, click the **Upstream Interface** tab, and then click **Import** in the Service Bundle Details area.

   The Import Interface Parameter Wizard appears.
2. In the **Sample XML** field, identify the XML file that represents a sample upstream service request.

3. In the **Stylesheet** field, identify the XML stylesheet (XSLT) that transforms the service request into SRT format.
4. Click **OK**.

   Design Studio parses the service request and adds the upstream parameters to the **Upstream Interface** tab.

Related Topics

Understanding Upstream Interface Parameters

Mapping Upstream Interface Parameters to Service Action Parameters

You map upstream interface parameters to service action parameters to enable the SRT to map the incoming data from the translated order to the service actions selected to run for the service bundle.

Note: When importing an SRT cartridge, error messages may appear indicating that some service bundles are not synchronous with the service actions they reference. This can happen when some service action parameters have not been mapped in the service bundle. If this occurs, you can synchronize the parameters in the Service Bundle editor. See "Synchronizing SRT Cartridge Parameters" for more information. See "Importing Activation SRT Cartridges from SAR Files" for more information about importing an SRT cartridge.

To map an upstream interface parameter to a service action parameter (method 1):

1. On the Service Bundle editor, click the **SA Parameter Map** tab, and increase the size of the **Source Type** field by dragging the slider to the right.

   You can also double-click the tab to increase the editor area.
2. For each service action parameter listed in the **SA Parameter Map** tab, select the parameter and change the value in the **Source Type** field (in the Service Bundles Details area) as follows:
   - If the source of the service action parameter comes from an upstream system, select **Interface** as the source type). Select an upstream parameter and drag it to the corresponding service action parameter. The upstream parameter label appears in the Source column.
   - If the source of the service action parameter comes from a lookup, select **Lookup** as the source type. Select a lookup result and drag it to the corresponding service action parameter. The lookup output label appears in the Source column.

To map an upstream interface parameter to a service action parameter (method 2):

1. On the Service Bundle editor, click the **SA Parameter Map** tab, and select from the list a service action parameter that you want to map to an upstream parameter.
2. In the Parameter Map Detail area, specify one of the following values in the **Mapping Type** field:
   - Select **Override** to impose a specific value on the service action parameter. If you select this option, you must specify a default value in the **Default** field. The default value is used if the incoming parameter value is not provided.

- Select **Interface** to map the service action parameter to an upstream parameter. If you select this option, you must specify the upstream source parameter and an optional default value. In the **Source** field, identify the upstream source parameter that you want to map to the service action parameter. If required, specify a default value for this parameter in the **Default** field. The default value is used if the incoming parameter value is not provided.
- Select **Lookup** to map the service action parameter to a lookup. If you select this option, you must identify the lookup that is to be performed by either selecting the appropriate lookup from the **Lookup** list, or by clicking **Open** to define the lookup in the Lookup editor. See "Configuring Lookups" for instructions on defining lookups. In the **Lookup Output Parameter** field, select the output parameter that is provided by the lookup. If required, specify a default value for this parameter in the **Default** field. The default value is used if the lookup does not generate a value for this parameter.

3. Verify that all parameters in the service action parameters list are valid.

   A parameter can become invalid if an atomic action parameter has been removed after the service bundle has referenced the service action to which the atomic action is mapped. A red circle with an X appear next to invalid parameters.

   It is recommended that you investigate whether the invalid parameters were changed or removed in error. If they are unnecessary for the service bundle, select these parameters and click **Remove Invalid**. You can optionally add a service action parameter by clicking **Add**, or clear the mappings by clicking **Clear**.

4. Save your changes.

Related Topics

Synchronizing SRT Cartridge Parameters

Understanding Service Bundles

Creating Service Bundles

Associating Service Actions with the Service Bundle

Understanding Service Action Spawning Conditions

Understanding Upstream Interface Parameters

Mapping Upstream Interface Parameters to Service Action Parameters

Understanding Lookups

Synchronizing SRT Cartridge Parameters

When importing SRT cartridges, error messages may appear indicating that some service bundles are not synchronous with the service actions they reference. This can happen when some service action parameters have not been mapped in the service bundle. If this occurs, you can synchronize the parameters in the Service Bundle editor.

To synchronize parameters after importing an SRT cartridge:

1. In the Activation SRT Project editor, click the **Properties** tab, and then click **Unsealed** to unseal the SRT cartridge project.
2. In the Studio Projects view, right-click the Service Bundle entity icon and select **Properties**.

   The Properties dialog box appears.

3. In the Properties dialog box, select the Read-write status option and click **OK**.
4. Open the Service Bundle editor.

   A dialog box prompts you to synchronize the Service Bundle parameters.
5. Click **OK**.
6. Rebuild the project.

Related Topics

Mapping Upstream Interface Parameters to Service Action Parameters

Understanding Lookups

Lookups are required when upstream systems fail to send all of the required information needed to model service bundles (for example, the data is not fully assigned, has incorrect labels or values, and so forth). Generally, a developer is required to create the code (java or stored procedure) for the lookup. Solution designers, however, can configure attributes of the lookups using Design Studio.

Lookups are executed by the SRT during the processing of an upstream XML service request. Lookups can be used to look up additional parameters that are required for activation of services, to format parameters so that they are compatible with service action parameters, and so forth. The lookup library contains custom lookups that are used by the SRT only. You may want to create lookups in the following situations:

- The upstream system is unable to provide a network element identifier. As opposed to using ASAP's internal support for network element routings (for example, ID routing, user defined routing, and so forth), you can create a lookup to accept an identifier (such as a MSISDN or IMSI in the case of mobile services, or DN or LEN in the case of PSTN services) and return one or more network element identifiers. You can implement this lookup as a stored procedure or java method to retrieve the data from a database table.
- Several parameters need to be concatenated or one parameter needs to be split apart in order to match the parameters required by one or more service actions. No database interaction is required as the algorithm is implemented inside a java method or java script.

After a technical resource creates the lookup (the stored procedure or java code that implements the lookup logic), you can define the lookup in Design Studio by describing its attribute, such as its name and input/output parameters. Existing lookups are available for use and appear in Service Bundle editor **SA Parameter Map** tab where they can be mapped to service action parameters. It also possible to create a series of lookups that feed parameters to each other as well.

Prior to deploying your Activation SRT cartridge to the environment, you package the required lookups in addition to service bundles and other element types.

Related Topics

Understanding Service Bundles

Creating Service Bundles

Associating Service Actions with the Service Bundle

Understanding Service Action Spawning Conditions

Understanding Upstream Interface Parameters

Mapping Upstream Interface Parameters to Service Action Parameters

Understanding Lookups

Packaging Activation SRT Cartridges

Service Bundle Editor

Use the Service Bundle editor to associate service actions with the service bundle and associate upstream parameters with service actions.

You can create the Service Bundle editor using the Service Bundle Wizard. In the Studio Projects view, double-click a service bundle entity to open the editor.

When working with the Service Bundle editor, see the following topics:

- Service Bundle Editor Editor Tab
- Service Bundle Editor Blueprint Tab

Service Bundle Editor Editor Tab

Use the **Editor** tab to not only associate service action entities to the service bundle, but also to add additional service action and upstream parameters for the service bundle.

Service Bundle Properties

Field	Use
Description	Specify a description for the Service Bundle editor.
Product Label and **Product Value**	Select a product label and specify a corresponding product value. These values map the upstream order (that contains the label and value) to this service bundle. During run time, the mapping enables SRT to determine the service bundle to execute based on the incoming XML document.
Include order data in response	Select to pass a more comprehensive response back upstream (beyond just the simple success or failure response) for this service bundle.

When working with the **Editor** tab, see the following topics:

- Service Bundle Editor Service Actions Tab
- Service Bundle Editor Upstream Interface Tab
- Service Bundle Editor SA Parameter Map Tab

Related Topics

Service Bundle Editor

Creating Service Bundles

Modeling Service Bundles

Service Bundle Editor Service Actions Tab

Use the **Service Actions** tab to associate service action entities with the service bundle and define conditions to spawn the service actions.

Service Bundle Details

Field	Use
Remove	Click to remove a service action from this tab.
Add	Click to add a service action to this tab.

Service Action Condition

Field	Use
Always	Select for ASAP to always spawn this service action for this service bundle. **Always** is selected by default when you add a service action to the Service Bundle editor.
Equals	Select for ASAP to spawn this service action only if the specified service action parameter, in the **Parameter Label** field, is defined on the service action and has a parameter value as defined in the **Parameter Value** field.
Defined	Select for ASAP to spawn this service action only if the service action parameter specified in the **Parameter Label** field is defined on the service action.
Not Defined	Select for ASAP to spawn this service action only if the service action parameter specified in the **Parameter Label** field is not defined on the service action.
Parameter Label	Select a label name from the list.
Parameter Value	Specify a value in the field. This field is available when you select the **Equals** button.
Include Expression	Define a logical expression using a number of criteria. The range of options available allows a service action to be executed if the service action parameter value is within a set range of values, or if the service action parameter is greater than, less than, or equal to, a specified value. You can combine multiple expressions using an AND or OR operator. See "Understanding Service Action Spawning Conditions" for more information.

Related Topics

Service Bundle Editor

Service Bundle Editor Upstream Interface Tab

Service Bundle Editor SA Parameter Map Tab

Associating Service Actions with the Service Bundle

Understanding Service Action Spawning Conditions

Service Bundle Editor Upstream Interface Tab

Use the **Upstream Interface** tab to modify the upstream parameter values. Apart from the upstream parameters added automatically, you can further add upstream parameters manually on this tab.

Service Bundle Details

Field	Use
Add	Click to manually add upstream interface parameters.

Field	Use
Remove	Click to remove upstream interface parameters.
Import	Click to import the upstream parameters from a sample upstream service request (in XML format).
Export Schema	Click to export the service bundle interface to an **.xsd** file and save it to your machine.

When working with the **Upstream Interface** tab, see the following topics:

- Details Tab
- Upstream Parameter Map Tab
- Lookup Map Tab
- Advanced Tab

Related Topics

Service Bundle Editor

Service Bundle Editor Service Actions Tab

Service Bundle Editor SA Parameter Map Tab

Understanding Upstream Interface Parameters

Details Tab

Use the **Details** tab to view and edit the parameters.

Field	Use
Name	Modify the name (label) of a parameter to match a parameter that will come from upstream.
Default	Specify an optional default value for an upstream parameter. If the upstream parameter does not have a value, the default is supplied to ASAP.
Description	Modify or provide a description for a parameter.
Data Type	Select to modify the data type for a parameter. Valid types include integer, string, boolean, and so on.
Required	Select if this upstream parameter is required.
Multi Instance	Select if the specified label is the stem of a series of parameters. This allows for a dynamic number of instances and possible use with ASAP's indexing capability for spawning multiple atomic actions.

Related Topics

Understanding Upstream Interface Parameters

Upstream Parameter Map Tab

Lookup Map Tab

Advanced Tab

Upstream Parameter Map Tab

Use the **Upstream Parameter Map** tab to display the mapping between the selected upstream interface parameter and service action label (service action parameters) associated with the service bundle.

Related Topics

Details Tab

Lookup Map Tab

Advanced Tab

Lookup Map Tab

Use the **Lookup Map** tab to display how the selected upstream interface parameter is used in different lookups for further data processing. For example in a mobile scenario, for a given MSISDN value, the SRT may need to perform a database lookup to determine which HLR to send the work order to or possibly split the MSISDN into country code, operator code, and the number parameters for use with a particular service action. See "Configuring Lookups" for information on creating lookups.

Related Topics

Understanding Lookups

Details Tab

Upstream Parameter Map Tab

Advanced Tab

Advanced Tab

Use the **Advanced** tab to specify complex data types for the selected upstream interface parameter that are different than those available in the **Details** tab.

In the **XML Schema** field, enter the complex data type for the selected upstream interface parameter.

Related Topics

Understanding Upstream Interface Parameters

Related Topics

Upstream Parameter Map Tab

Lookup Map Tab

Service Bundle Editor SA Parameter Map Tab

Use the **SA Parameter Map** tab to associate upstream parameters with service action parameters. This association enables SRT to map the incoming data from the translated order to the selected service action to run for the service bundle.

Field	Use
Source Type	Select a parameter in the Name grid and modify the value to any one of the following: ■ **Interface**: Select if the source of the service action parameter comes from an upstream system. ■ **Lookup**: Select if the source of the service action parameter comes from a lookup.
Name	Specify a name for an upstream parameter and click **Add**.
Mapping Type	Select any one of the following: ■ **Override**: Select to impose a specific value on the service action parameter. If you select this option, you must specify a default value in the **Default** field. The default value is used if the incoming parameter value is not provided. ■ **Interface**: Select to map the service action parameter to an upstream parameter. In the **Source** field, identify the upstream source parameter that you want to map to the service action parameter. If required, specify a default value for this parameter in the **Default** field. The default value is used if the incoming parameter value is not provided. ■ **Lookup**: Select to map the service action parameter to a lookup. From the Lookup list, select an appropriate lookup or click **Open** to define the lookup in the Lookup editor. See "Configuring Lookups" for more information. In the **Lookup Output Parameter** field, select the output parameter that is provided by the lookup. If required, specify a default value for this parameter in the **Default** field. The default value is used if the lookup does not generate a value for this parameter.
Add	Click to add a service action parameter to this tab.
Remove	Click to remove a user defined parameter from the service action parameter list.
Remove Invalid	Click to remove any unnecessary invalid parameter (a red circle with an X appears next to an invalid parameter) for the service bundle.
Clear	Click to clear the editable contents of the service action parameter such as **Mapping Type**, **Source** and **Default**.

Related Topics

Service Bundle Editor

Service Bundle Editor Service Actions Tab

Service Bundle Editor Upstream Interface Tab

Mapping Upstream Interface Parameters to Service Action Parameters

Service Bundle Editor Blueprint Tab

Use the **Blueprint** tab to view the generated documentation for the service bundle entity. This tab is read only.

Related Topics

Service Bundle Editor

Service Bundle Editor Editor Tab

Modeling Service Bundles

5 Packaging and Deploying ASAP SRT Cartridges

To package an Activation SRT cartridge and deploy it to the environment, you must complete the steps in the following topics:

- Packaging Activation SRT Cartridges
- Deploying Cartridge Projects

Packaging Activation SRT Cartridges

Packaging enables you to control what elements to include in the SRT cartridge SAR file that you will deploy to an Oracle Communications ASAP environment. Packaging enables you to control what elements are deployed to an ASAP environment. For example, you want to exclude from deployment new lookups that are not yet ready to be deployed and tested. Or, you may have service bundles that you want to deploy to specific ASAP environments.

To package an Activation SRT cartridge:

1. Select **Studio**, then select **Show Design Perspective** to display the SRT cartridges.

   For each cartridge, higher level information is available and can be edited within the Project editor.

2. In the Studio Projects view, double-click the Project entity to open it in the Project editor.

3. Select the **Packaging** tab.

   The cartridge packaging instructions information appears, with a list of all types of elements that can be packaged inside the cartridge. By default, everything within the cartridge project will be included when the cartridge is built.

4. To specify which elements should be included for any of the element types (service bundles, lookups, translations, and so forth):

   a. Select the element type to display all elements of this type in the Entity column.

   b. Deselect **Include all from Project**.

      All elements are cleared from the list.

   c. Click **Select** to display the Service Bundles Selection dialog box.

   d. Select the service bundles that you want to include when packaging the cartridge and click **OK**.

The specified service bundles appear in the Service Bundles Entity column.

5. Select **File**, then select **Save** to save the changes.

   The cartridge is rebuilt and contains the specified elements.

Related Tasks

Understanding Java Libraries in Design Studio

6 Modeling Translations

You must configure translations for upstream systems that send information to Oracle Communications ASAP prior to (or in parallel with) modeling service bundles and before deploying Activation SRT cartridges to the ASAP environment.

Translations enable the SRT to place orders received from upstream into a recognizable format on which subsequent processing can occur, and enable the SRT to place responses from ASAP back into a format recognizable by the upstream system.

Note: Translations are usually implemented by a developer.

Related Topics

Understanding Translations

Configuring Translations

Translation Editor

Understanding Translations

Using the Translation editor, you can identify one or more XSLT scripts to be used by the SRT component of ASAP at run time (the XSLT scripts must be created by a developer). These XSLT scripts enable an XML document to be transformed into another XML document of a different structure. For example, an XML document arriving at the SRT from an upstream system must be transformed into an XML document that conforms to the SRT service activation schema. Alternatively, an XML document arriving at the SRT from ASAP (for example a work order failure notification) must be transformed into an XML document format that is expected by an upstream system.

Translations can be implemented to:

- Translate incoming requests from an upstream system.
- Translate ASAP responses to ASAP events (such as FAILURE, COMPLETE) for return to the upstream system.
- Translate outgoing ASAP events (such as FAILURE, COMPLETE) to the upstream system.

By convention, translations are contained in a single file. It is not necessary to create additional files for different upstream systems, but you may choose to do so for organizational purposes.

Related Topics

Modeling Translations

Modeling Service Bundles

Modeling Activation SRT Cartridges

Understanding ASAP SRT Cartridges

Configuring Translations

To configure translations:

1. Select **Studio**, then select **Show Design Perspective**.
2. In the Studio Projects view, right-click and select **New**, then select **Translation.**

   Alternatively, select **Studio**, select **New,** then select **Translation.** The Studio Model Entity wizard appears.
3. Select the correct Activation SRT cartridge project for this element and enter a name for the entity.
4. Click **Browse** to open a Select Location dialog box and select a location from a list.

   This populates the **Location** field and specifies the folder under the cartridge project where the new translation will be located. If no folder exists, type in the location name, which creates a new folder with that name.

   > **Note:** Specifying a location is optional, but is recommended to organize your elements and to avoid locating elements directly below the root folder.
5. Click **Finish**.

   In the Studio Projects view, the new translation appears under your project folder, and the Translation editor appears.
6. In the Translation Details area, click **Add** to add a new translation.
7. Select the new translation that appears in the Translation Details area.
8. Define values for the fields in the Translation Map Details area.

   See "Translation Editor" for information about the Translation editor fields.
9. Save your configuration.

Related Topics

Modeling Translations

Translation Editor

Modeling Service Bundles

Modeling Activation SRT Cartridges

Understanding ASAP SRT Cartridges

Translation Editor

Use the Translation editor to configure translations. In the Studio Projects view, double-click a translation entity to open the Translation editor. You can create the editor using the Translation Wizard.

When working with the Translation editor, see the following topics:

- Translation Editor Editor Tab
- Translation Editor Blueprint Tab

Translation Editor Editor Tab

Use the **Editor** tab for defining values for translations.

Translation Details

Field	Use
Add	Click to add a row with default values for a translation.
Remove	Click to remove the row with values for a translation.

Details

Field	Use
Name	Specify a unique name for the translation.
Type	Select any one of the following: ▪ **XSLT** to indicate that an XSLT translation occurs. The XSLT file is identified in the **XSLT Script** field. ▪ **Do not forward** to indicate that it is not necessary to notify the upstream system of the event (such as a work order completion event).
Direction Type	Select any one of the following: ▪ **Incoming** to indicate that the XSLT is to be used by the SRT to translate an XML document from an upstream system (for example, Siebel) for ASAP. ▪ **Event** to indicate that the XSLT is to be used by the SRT to translate an XML document received from ASAP (for example, event information). ▪ **Outgoing** to indicate that the XSLT is to be used by SRT to translate an XML document of an event into a format accepted by the upstream system (for example Siebel).
Select	Click to select the XSLT file that implements the translation. The file is located in the *WorkspaceName/SRTCartridgeProjectName*/**Translations** folder.
Dispatch Condition Type	Select the condition under which the translation is applied to the upstream XML document. Select any one of the following conditions, that can based either on a JMSHeader or an XPath in the upstream document: ▪ **JMSHeader** to initiate the translation based on the data contained in the message properties in the JMS message. ▪ **XPath** to initiate the translation based on the data contained in the upstream XML.

Field	Use
Property Name	Displays the JMSHeader property or XPath condition (depending on the condition type) that must be met for this translation to be applied.
Property Value	Displays the JMS header property value or XPath value (depending on the condition type) that must be matched for this translation to be applied.

Query

Field	Use
Query Type	Select any one of the following conditions on which the query can be invoked: ▪ **Order Request** to query a work order. ▪ **Audit Value** to retrieve all audit trails associated with a particular work order. ▪ **Service History** to retrieve the history of a service action. ▪ **Atomic Action History** to retrieve the history of an atomic action.
Dispatch Condition Type	Select a condition type based on which a query can be invoked.
Condition Label	Specify the JMSHeader condition name or XPath condition (depending on the condition type) that must be met to invoke the query for this translation.
Condition Value	Specify the JMS header condition value or XPath value (depending on the condition type) that must be matched to invoke the query for this translation.

XPath Map

Field	Use
Parameter Name XPath	Specify the name of the XPath parameter.
Parameter Value XPath	Specify the XPath value. The SRT runs the configured XPath value and the returned name and value pairs are added to the JMS header properties.
Add	Click to add default XPath parameter name and value.

Related Topics

Translation Editor

Modeling Translations

Translation Editor Blueprint Tab

Use the **Blueprint** tab to view the generated documentation for the translation entity. This tab is read only.

Related Topics

Translation Editor

Modeling Translations