Oracle® Communications Unified Inventory Management

Cartridge Guide

Release 8.0.1
G50254-01
April 2026

ORACLE

Oracle Communications Unified Inventory Management Cartridge Guide, Release 8.0.1

G50254-01

Contents

About This Content

1 Overview

About Cartridges 1
- Understanding Cartridge Dependencies 2

Base Cartridges 2
- Deploying Base Cartridges 3
- Opening Base Cartridges 3

Required Cartridges 3
- About the Domain Model Cartridge 4
- About the MDS Cartridge 5

Sample Cartridges and Cartridge Packs 5
- Examples 5
- Technology Solutions 5
- Downloading Sample Cartridges 6
- Using Sample Cartridges and Cartridge Packs 7

2 Managing Cartridges and Cartridge Project Archives

Opening Cartridge Project Archives in Design Studio 1

Deploying Cartridges 1
- Using the Cartridge Deployer Tool 2
  - Deploying Cartridges with the Cartridge Deployer Tool 2
  - Viewing Deployed Cartridges 5
  - Using Cartridge Deployment Log for Analyzing Cartridge Deployment Errors 6

About Cartridge Bundles 6
- Creating Cartridge Bundles 6
- Automating Cartridge Bundling 7
- Resolving Problems Related to Multiple Deployments 9
  - Purging Metadata Version History by Using Oracle Fusion Middleware Control 9
  - Purging Metadata Version History by Using the WLST Script 10

Guidelines and Recommended Cartridge and Cartridge Pack Deployment 10

3 Upgrading and Extending Cartridges and Cartridge Packs

How UIM Data Is Affected by Cartridge Changes 1
Leading Practices for Extending Cartridges 2
- Extending Cartridges By Adding New Content in Separate Cartridges 2
- Renaming and Modifying Cartridges 3
  - Considerations When Modifying Cartridge Content After Entity Data Creation 4
  - Actions to Take After Renamed Cartridge Has Been Deployed 5
- Tracking Cartridge Dependencies 5
- Using Ruleset Stubs to Customize Validation and Allocation 6
- Use Manual Versioning to Update Specifications 7
- Use Database Snapshots to Allow Rollbacks of Upgrades 7
- Special Considerations for Configuration Versions 7
- Adopt a Systematic Naming Policy 8

Recommended Cartridge Life Cycle 8

4 Common Content

CommonManager 1
ResourceManager 2
ServiceManager 3

5 Base Cartridge Reference

Base Specifications Cartridge 1
- Business Interaction Specifications 1
- Connectivity Specifications 1
- Custom Involvement Specifications 2
- Inventory Group Type Specifications 2
- IP Resource Specifications 2
- Managed Project Specifications 3
- Network Specifications 3
- Property Location Specifications 3

Base Measurement Entities 4
- Capacity Types 4
- Measurement Types 4
- Units of Measure 4

Phone Management Specifications and Rulesets 5
- Characteristics 5
- Rulesets 6

Telephone Number Specifications and Rulesets 6
- Entity Specifications 6

Telephone Number Specifications 7
Rulesets 9
Base Technologies Cartridge 10
Connectivity Functions 10
Technology Types 14
Rate Codes 15
Connectivity Signal Termination Point Specifications 20
Base Tags Cartridge 28

6 Base Rulesets and Extension Points

Base Rulesets 1
Base Ruleset Descriptions 1
Base Extension Points 3
Base Extension Point Descriptions 3

7 NFV Orchestration Base Cartridges

NFV Orchestration Base Specifications Cartridge 1
Business Interaction Specification 1
Custom Object Specification 1
Device Interface Specification 3
Inventory Group 4
IP Address Resource Extension 4
Network 5
Network Address Domain 5
Place 5
Extension Point 6
Enabled Extension Point 7
Rulesets 8
NFV Orchestration Base Tags Cartridge 9
Tags 9

About This Content

This guide provides information about how you use cartridges and cartridge packs with Oracle Communications Unified Inventory Management (UIM). The guide contains the background and introductory information about the cartridges and cartridge packs, information about the contents of cartridge packs, instructions about deploying cartridges and opening cartridge project archives, instructions about bundling cartridges, information about extending and upgrading cartridges and cartridge packs, and reference information about base cartridges.

Audience

This guide is intended for:

- Administrators who deploy cartridges in test and production environments
- Business analysts who require information about solutions can be implemented and deployed
- Developers and modelers who design and implement solutions
- Developers and modelers who customize and extend cartridge packs

This guide assumes that you have a working knowledge of UIM.

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.

Conventions

The following text conventions are used in this document.

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1 Overview

This chapter introduces Oracle Communications Unified Inventory Management (UIM) cartridges and cartridge packs. It also introduces the concept of extending UIM by using them.

> **Note**
>
> - The **UIM_SDK_Home** represents the directory of **UIM_SDK.zip** when unpacked from Oracle Software Delivery Cloud.
> - In traditional environment, the SDK contents are available as part of **UIM_Home**. To refer any files, you can use **UIM_SDK_Home** or **UIM_Home** directories.
> - In UIM cloud native environment, **UIM_Home** is not accessible in Kubernetes worker nodes. To refer any files, use **UIM_SDK_Home** instead of **UIM_Home**.

About Cartridges

You can extend UIM functionality dynamically without rebuilding the application. You do so by deploying *cartridges* into the application. A cartridge is a collection of entity specifications, characteristics, rulesets, and code that is defined in an Oracle Communications Service Catalog and Design - Design Studio project. The project is compiled into a deployable JAR file known as a cartridge.

When you deploy the cartridge into UIM, the data it contains becomes available in the application. For example, if you deploy a cartridge that includes several different types of logical device specifications, those specifications become available for creating logical device entities in UIM.

Oracle supplies cartridges in various ways to extend UIM.

- **Samples:** Sample cartridges and cartridge packs provides specifications, extensions, rulesets, and code that you can use as-is or as a starting point for your own development, testing, and experimentation. Some Oracle sample cartridge packs address particular technology domains, such as Carrier Ethernet, GSM/3GPP, and Cable TV. These cartridge packs can accelerate your deployment of a solution by supplying all or most of the required entity specifications. See "Sample Cartridges and Cartridge Packs" for more information.
- **Base cartridges:** Base cartridges provide fundamental capabilities and features required by cartridge packs and other cartridges. Cartridge packs and other cartridges are dependent on the data supplied by base cartridges. See "Base Cartridges" for more information.
- **Required cartridges:** Some cartridges are required to be open in the Design Studio workplace when you develop content for UIM. See "Required Cartridges" for more information.

The names of all Oracle-supplied cartridges, whether they are supplied in a cartridge pack, as base cartridges, or as samples, begin with *ora_uim* or *OracleComms*.

In addition to the cartridges supplied by Oracle, you can also create your own custom cartridges in Design Studio. For example, you can create custom specifications and rulesets that are specific to your business and technology, then deploy them to UIM.

Many cartridges, including those supplied by Oracle in cartridge packs and base cartridges, have accompanying ZIP archive files containing the Design Studio project data used to compile the cartridge. You can import the ZIP files into Design Studio to open the corresponding project for review or extension. The archive file is included in the larger cartridge pack ZIP file.

Understanding Cartridge Dependencies

Complete solutions in UIM include multiple cartridges with relationships to each other. For example, cartridge packs are divided into multiple cartridges, each providing a particular set of capabilities. Some cartridges can even be used in multiple cartridge packs. This modularity means that you can design any number of solutions with no rework required for shared capabilities.

These relationships create dependencies between cartridges. For example, suppose you include a specification from Cartridge A as a specification option in a configuration item in Cartridge B. In this situation, Cartridge B is dependent on Cartridge A. You must deploy Cartridge A before Cartridge B. Larger numbers of cartridges obviously lead to more complex dependencies.

Design Studio displays these dependencies in the **Dependency** tab of Inventory Project editors and prevents circular dependencies. UIM enforces these dependencies when you deploy cartridges. You see an error if you try to deploy a cartridge before all of its dependent cartridges have been deployed.

Base Cartridges

Base cartridges provide fundamental capabilities and features required by cartridge packs and inventory entities. Each base cartridge provides a different set of contents depending on its purpose. For the example, the base measurements cartridge provides measurement types, units of measure, and capacity types that you can use to define pipe capacity in UIM.

Cartridge packs and other cartridges require the content in the base cartridges and therefore have dependencies on them. Both UIM and Design Studio enforce these dependencies:

- You must open the necessary base cartridges in Design Studio in your workspace before you can compile cartridges dependent on them.
- You must deploy the necessary base cartridges to UIM before you deploy cartridge packs and cartridges dependent on them.

Although not all base cartridges are required by all cartridge packs, Oracle recommends that you deploy the base cartridges before deploying cartridge packs or other cartridges into UIM.

See "Base Cartridge Reference" for detailed information about the contents of the following base cartridges:

- Base Specifications cartridge: ora_uim_basespecifications
- Base Measurements cartridge: ora_uim_basemeasurements
- Base Phone Management cartridge: ora_uim_basephone_mgmt
- Telephone Number cartridges:
  - ora_uim_us_tn

  - ora_uim_canada_tn
  - ora_uim_norway_tn
  - ora_uim_uk_tn
  - ora_uim_saudi_arabia_tn
- Base Technologies cartridge: ora_uim_basetechnologies

See "Base Rulesets and Extension Points " for detailed information about the contents of the following base cartridges:

- Base Extension Points cartridge: ora_uim_baseextpts
- Base Rulesets cartridge: ora_uim_baserulesets

See "NFV Orchestration Base Cartridges " for detailed information about the contents of the following base cartridges:

- NFV Orchestration Base Specifications cartridge: OracleComms_NSO_BaseCartridge
- NFV Orchestration Base Tags cartridge: OracleComms_NSO_BaseTags

Deploying Base Cartridges

Base cartridges are packaged as JAR files and are part of the UIM installation. They are located in the *UIM_home*/**cartridges**/**base** directory while installing a traditional UIM environment. For UIM cloud native environment, the base cartridges are located in the *UIM_SDK_Home*/**cartridges**/**base** directory.

The base cartridges are independent of each other except for the country-specific Telephone Number cartridges, which are dependent on the Base Phone Management cartridge. You must first deploy the Base Phone Management cartridge to be able to deploy any of the country-specific Telephone Number cartridges into UIM.

You deploy base cartridges into UIM in the same manner as other cartridges. See "Deploying Cartridges" for more information.

Opening Base Cartridges

You must open base cartridges in Design Studio when you work with other cartridges that are dependent on them. You can also make additive extensions to base cartridges in separate cartridges. See *SCD Design Studio Modeling Inventory* and *UIM Developer's Guide* for information about opening files in Design Studio.

> **Note**
>
> You should not unseal or modify base cartridges in Design Studio. See "Upgrading and Extending Cartridges and Cartridge Packs " for information about additively extending cartridges.

Required Cartridges

In addition to the base cartridges, Oracle supplies two cartridges that are required to develop UIM cartridges in Design Studio:

- ora_uim_model

- ora_uim_mds

The projects in these two cartridges must be open in your Design Studio workplace before you can compile a UIM cartridge.

> **Note**
>
> Unlike base cartridges, ora_uim_mds and ora_uim_model are not deployed to UIM. They are required only in Design Studio.

The required cartridges for UIM traditional and cloud native are available in the following locations:

- **UIM traditional:**

  UIM_home/**cartridges**/**base** directory

- **UIM cloud native:**

  UIM_SDK_Home/**cartridges**/**base** directory

See *SCD Design Studio Modeling Inventory* for information about open cartridges in Design Studio.

> **Note**
>
> The information in the required cartridges reflects data that is installed in UIM. It can change when a UIM patch or upgrade is installed. Check with a system administrator to ensure that you use the current version of these cartridges.

About the Domain Model Cartridge

The Domain Model cartridge (ora_uim_model) is a read-only Design Studio project that represents the entities, data elements, and relationships defined in the UIM model. The ora_uim_model cartridge must be open in Design Studio when you create or use UIM cartridges.

The ora_uim_model cartridge includes the inventory entities on which UIM specifications are based. For example, the model cartridge includes an entity called BusinessInteraction. When you create a Business Interaction specification in Design Studio, you are extending the BusinessInteraction entity to define a new kind of business interaction. You can see the entities from which specifications are extended in the **Extends** tab of Specification editor.

You must import and open the ora_uim_model cartridge to make the entities in the corresponding project available in Design Studio.

You use the Design Studio Inventory Entity editor to review the inventory entities included in the ora_uim_model project. The editor displays the data elements defined for the entity and (when applicable) the parent entity from which the entity inherits data elements. This editor is read-only and is for informational purposes only.

About the MDS Cartridge

The MDS cartridge (ora_uim_mds) is a read-only Design Studio project that represents the contents of work areas and sections included in the UIM user interface. This information is used to populate the **Pages/Panels** field in **Layout** tabs of Design Studio specification editors. The ora_uim_mds cartridge must be open in Design Studio when you create or use UIM cartridges.

Sample Cartridges and Cartridge Packs

Oracle provides sample cartridges that you can use for testing and as the basis of your own development. There are two groups of samples available in two ways: examples and technology solutions.

Examples

The following three UIM sample cartridges provide working examples of specific UIM features. You can find them on the computer where you installed UIM in the *UIM_Home*/**cartridges**/**sample** directory for traditional UIM or in the *UIM_SDK_Home*/**cartridges**/**sample** directory for UIM cloud native.

- ora_uim_servicetopology_sample

  The UIM topology is a graphical representation of the spatial relationships and connectivity among your related inventory entities. The ora_uim_servicetopology_sample cartridge contains characteristics, specifications, extension points, and rulesets that collectively provide a working example of extending the topology of a specific service.

- ora_uim_pathanalysis_sample

  UIM can use an algorithm to determine the path between entities. This algorithm is called the path analysis. The ora_uim_pathanalysis_sample cartridge contains characteristics, specifications, extension points, and rulesets that collectively provide a working example of extending the path analysis of a specific service.

- ora_uim_geocoder_sample

  Geocoding is the process of associating geographic coordinates, such as latitude and longitude, with other geographic data, such as street addresses or postal codes. The ora_uim_geocoder_sample cartridge contains characteristics, specifications, extension points, and rulesets that collectively provide a working example for geocoding a US address.

> **Note**
>
> To use the ora_uim_geocoder_sample cartridge, you must license Oracle Spatial 11g. In addition, you must have geocoding data in Oracle Spatial format, such as that purchased from a third-party vendor like NAVTEQ.

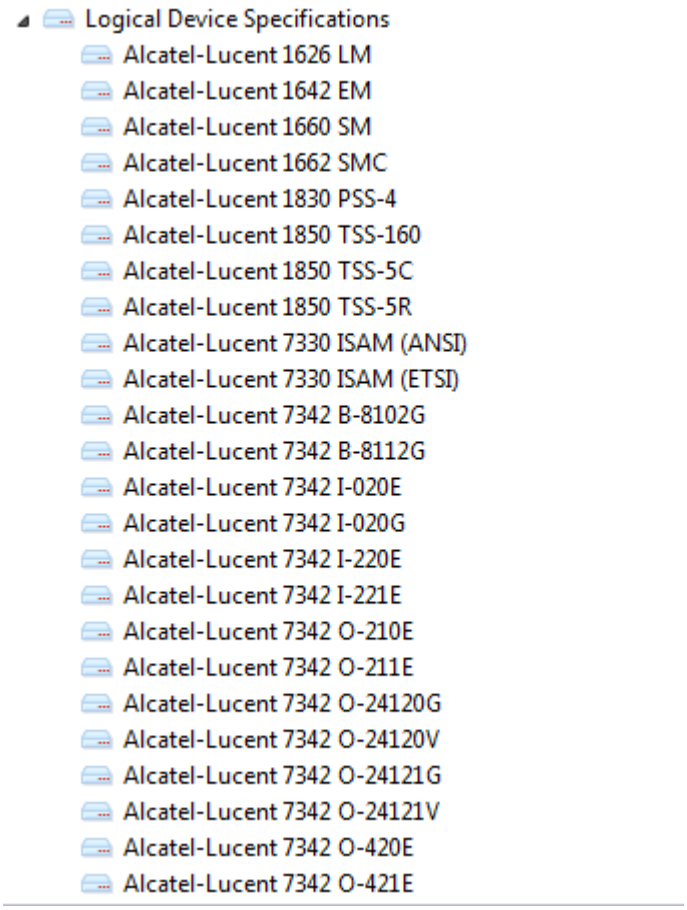
Technology Solutions

UIM provides sample cartridges and cartridge packs that address specific technology domains. Some have full documentation. For example, the UIM Carrier Ethernet cartridge includes specifications and other artifacts that enable you to implement a Carrier Ethernet solution.

UIM also provides sample federation cartridge packs that enable communication, cooperation, and data exchange with external systems. See "Overview" in *UIM Developer's Guide* for more information about federation and the federation cartridge packs.

The Devices sample cartridge pack includes specifications for many types of equipment and devices. The cartridge pack includes cartridges for specific manufacturers as well as for generic equipment. Device specifications are manufacturer- and model-specific. For example, Figure 1-1 shows some of the Logical Device sample specifications provided for Alcatel-Lucent.

Figure 1-1 Alcatel-Lucent Logical Device Specifications



Downloading Sample Cartridges

The sample cartridges and cartridge packs are available for download as part of the UIM software from the Oracle Software Delivery Cloud at:

https://edelivery.oracle.com/

The sample cartridge packs and their documentation are included in the **OracleComms_UIM_CartridgePacks.zip** file.

- Federation Data Domain cartridge pack
- Federation Protocol cartridge pack

- Devices cartridge pack
- UIM Carrier Ethernet cartridge
- UIM Packet cartridge
- UIM DSL cartridge
- Cable TV cartridge pack
- Consumer VoIP cartridge pack
- GSM 3GPP cartridge pack
- L2 VPN cartridge pack
- Metro Ethernet cartridge pack
- MPLS L3 VPN cartridge pack
- Channelized Connectivity cartridge pack
- Mobile GSM cartridge pack

Using Sample Cartridges and Cartridge Packs

To use sample cartridges and cartridge packs, you can:

- Deploy the sample into a UIM test environment to analyze the data and give you ideas for implementing a customized version in a production environment.
- Import the sample cartridge into Oracle Communications Service Catalog and Design - Design Studio to analyze the characteristics, specifications, extension points, and rulesets to understand how to implement a customized version in a production environment.

Sample cartridges all have dependencies on base cartridges, such as the ora_uim_baseextpts cartridge. The individual cartridges in cartridge packs also have dependencies on each other.

You must open the base cartridges and other cartridges on which a sample has dependencies in Design Studio before making changes or compiling. Similarly, you must deploy all cartridges on which a sample cartridge has dependencies into UIM before deploying the sample.

> **Note**
>
> Oracle recommends that you deploy all of the base cartridges into UIM.

For instructions on how to import a sample cartridge into Design Studio, see *SCD Design Studio Modeling Inventory*. When sample cartridges are imported into Design Studio, the project within the cartridge is compiled. If compiler errors are present, you must configure the project library list. See *UIM Developer's Guide* for more information. See "Deploying Cartridges" for more information about deploying cartridges to UIM test environments.

To ensure data consistency, you must observe certain guidelines and best practices when extending and upgrading sample cartridges. See "Upgrading and Extending Cartridges and Cartridge Packs " for more information.

Sample cartridges are sealed to protect them from accidental modification. Oracle recommends that you leave them sealed and make additive changes rather than direct modifications.

2

Managing Cartridges and Cartridge Project Archives

This chapter explains how you work with cartridges and cartridge bundles in Oracle Communications Unified Inventory Management (UIM). It also provides information about the cartridge project archives that you use in Oracle Communications Service Catalog and Design - Design Studio.

Opening Cartridge Project Archives in Design Studio

Many cartridges, such as those in cartridge packs, are accompanied by cartridge project archives that you can import into Design Studio for review or extension. The project archives are ZIP files included in the larger cartridge pack ZIP file.

For example, in the Consumer VoIP cartridge pack, each of the deployable cartridge files, such as **ora_uim_service_location_cartproj-*.jar** is accompanied by a corresponding project archive file, such as **ora_uim_service_location_cartproj-*.zip**. (In the actual file names, the asterisk is replaced by a build number.)

See *SCD Design Studio Modeling Inventory* for information about importing project archive files into Design Studio. See "Upgrading and Extending Cartridges and Cartridge Packs " and *UIM Developer's Guide* for information about how you can extend cartridge projects in Design Studio.

Deploying Cartridges

You can deploy cartridges into traditional UIM in the following ways. However, you can deploy cartridges into a UIM cloud native environment from Design Studio or using CMT. See the **Deploying UIM Cartridges** section in *UIM Cloud Native Deployment Guide* for more information on deploying cartridges in UIM cloud native environment.

- From Design Studio. You can deploy cartridges and cartridge packs interactively from Design Studio to test environments. Design Studio enables you to manage cartridges in the test environment consistently, manage common test environment connection parameters across the design team, and compare cartridge version and build numbers in the development environment with those of the cartridges deployed in the test environment. See *SCD Design Studio Modeling Inventory* for more information.
- By using the Design Studio Cartridge Management Tool (CMT). The CMT enables you to automate cartridge deployment. You can use the CMT to deploy cartridges into both test and production UIM environments. You can also use it to deploy cartridges into cluster environments. The CMT enables you to deploy cartridge pack bundles that include multiple individual cartridges. See "Creating, Packaging, and Distributing Plug-in Projects" in *SCD Developer's Guide* for more information about the CMT, and see "About Cartridge Bundles" for more information about cartridge bundles.
- By using the UIM Cartridge Deployer Tool (CDT). The UIM CDT is a GUI-based tool that enables you to deploy cartridge projects. The Oracle Universal Installer installs the CDT as part of the UIM installation process. The CDT enables you to deploy cartridge pack bundles that include multiple individual cartridges. You can use the CDT to deploy

cartridges into both test and production UIM environments. You can also use it to deploy cartridges into cluster environments. See "Using the Cartridge Deployer Tool" and "About Cartridge Bundles" for more information.

You must ensure the following to successfully deploy a cartridge:

- All required base cartridges are deployed into UIM before deploying other cartridges. Oracle recommends that you deploy all base cartridges even if they are not immediately required. See "Base Cartridges" for more information
- The user deploying the cartridge must have **create table** privilege on the UIM database schema. Characteristics within a cartridge require additional database tables and this user privilege is required.

Using the Cartridge Deployer Tool

The Cartridge Deployer Tool is a component of UIM. The Oracle Universal Installer installs the Cartridge Deployer Tool as part of the installation process in a location selected by the customer. See "Unified Inventory Management Installation Overview " in *UIM Installation Guide* for more information.

Deploying Cartridges with the Cartridge Deployer Tool

The Cartridge Deployer Tool does not validate cartridge dependencies. If any selected cartridge is dependent upon another cartridge, you must ensure that the cartridge upon which the dependency is based is already deployed or is selected first in the list for deployment.

> ⚠ **Caution**
>
> Before you start deploying cartridges using the Cartridge Deployer Tool, ensure that the WebLogic administration console is not locked for editing.

To deploy cartridges:

1. Navigate to *CDT_Home*/**CartridgeDeployerClients/CartridgeDeployer** directory.

   where *CDT_Home* is the directory where the Cartridge Deployer Tool was installed.
2. Run the following command:

   ./runCartridgeDeployer.sh

   The Cartridge Deployer Welcome page appears.
3. Select the **Deploy Cartridge** option and click **Next**.

   The Select Cartridge Type page appears.
4. From the Cartridge Type list, select **Inventory** and click **Next**.

   The Cartridge Location page appears.
5. Click **Browse** to search for and select the cartridges you want to deploy.

   You can select multiple cartridges from a single directory by holding down the **Ctrl** key.

> **Note**
>
> The customized file browser shows only predefined cartridge extensions. UIM supports cartridges with only the **.jar** extension.

6. Click **Next**.

   The Configure Deployment Queue page appears.

7. View the details of the selected cartridges and click **Next**.

   > **Note**
   >
   > To add Deploy property or Model property, under **Details** for that cartridge, right-click on **Properties** and select the respective options for related menus.

   The WebLogic Connection Information page displays.

8. Do the following:

   a. In the **Host name or IP address** field, enter the host name or IP address of the WebLogic Administration Server.

   b. In the **Port number** field, enter the port number of the WebLogic Administration Server.

   c. Select **Use SSL (if enabled) while connecting to WebLogic Admin server**.

   d. (Optional) In the **Keystore location** field, enter the path or click **Browse** to search for the Keystore location. Leave this field blank if no Keystore location exists.

   e. In the **CMWS User** field, enter the user name of the Cartridge Management Web service (CMWS) user.

   > **Note**
   >
   > Use your WebLogic administrator or CMWS user name.
   >
   > The CMWS user is a WebLogic server user belonging to the Cartridge_Management_Webservice group. The Cartridge_Management_Webservice group is also a member of the Administrators group.

   f. In the **Password** field, enter the password for the CMWS user.

   > **Note**
   >
   > Use your WebLogic administrator user password.

   g. Click **Next**.

   The Select WebLogic Target page displays.

9. In the list, select the managed server where the CMWS is deployed and click **Next**.

If the Secure Sockets Layer (SSL) is not configured correctly, the following message displays:

```
SSL Handshaking failed. You can proceed without SSL by unchecking SSL options
on the bottom of this screen.
```

> **Note**
>
> The SLL handshake fails when the Cartridge Deployer Tool connects to the CMWS using HTTPS.

10. If you receive this message, click **OK** in the message dialog box, and deselect the **Use SSL (if enabled) while connecting to Cartridge Management Web service** check box located at the bottom of the page.
11. Click **Next**.

    The Review Deployment page displays.
12. Review and confirm your selections and click **Next**.

    The Cartridge Deployment page displays.

> **Note**
>
> The Cartridge Deployer Tool rejects cartridges if a higher version already exists. You can view rejected cartridges in the **Cartridges rejected for this deployment session** list.

13. Click **Deploy**.

    You can view the deployment progress.

    Logs returned by the Cartridge Deployer Tool are displayed at the end of each cartridge deployment operation. The logs display whether the cartridge deployment succeeds or fails. In case of failure, the UIM server log includes detailed information about the specific error.

> **Note**
>
> If the UIM server goes down during cartridge deployment, the cartridge is recovered after the UIM server is up again, or during the next cartridge deployment session, with the cartridge deployment request showing Failed.

If a cartridge requires redeployment only, the CMWS will redeploy automatically. For more information on redeployment, see the WebLogic server administration console Help.

If a cartridge requires a redeployment and a restart, then you must deploy the cartridge, redeploy the **inventory.ear** file using the WebLogic administration server console, and then restart the server.

> **Note**
>
> If you deploy a cartridge that contains Java code, the Cartridge Deployer automatically redeploys the **inventory.ear** file. In addition, you must manually redeploy the **custom.ear** file by using the WebLogic administration server console.

Viewing Deployed Cartridges

To view deployed cartridges:

1. Locate the *UIM_home*/**CartridgeDeployer** directory.
2. From a command line, run the following command to run the Cartridge Deployer Tool executable:

   ./runCartridgeDeployer.sh

   The Cartridge Deployer Welcome page displays.
3. Select the **View Deployed Cartridges** option, and click **Next**.

   The Select Cartridge Type page displays.
4. Select **Inventory** in the Cartridge Type list, and click **Next**.

   The WebLogic Connection Information page displays.
5. Do the following:
   a. In the **Host name or IP address** field, enter the host name or IP address of the WebLogic Administration Server.
   b. In the **Port number** field, enter the port number of the WebLogic Administration Server.
   c. In the **CMWS User** field, enter the user name of the Cartridge Management Web service (CMWS) user.

      > **Note**
      >
      > Use your WebLogic administrator user name.
      >
      > The CMWS user is a WebLogic server user belonging to the administrators group.

   d. In the **Password** field, enter the password for the CMWS user.

      > **Note**
      >
      > Use your WebLogic administrator user password.

   e. Click **Next**.

   The Select WebLogic Target page displays.
6. Select the WebLogic targets where the Cartridge Management Web service is installed and click **Next**.

> **Note**
>
> In some cases, the WebLogic targets may be different from where UIM is installed.

If SSL is not configured correctly, the following message displays:

SSL Handshaking failed. You can proceed without SSL by unchecking SSL options on the bottom of this screen.

> **Note**
>
> The SLL handshake fails when the Cartridge Deployer Tool connects to the CMWS using HTTPS.

7. Click **OK** in the message dialog box, and deselect the **Use SSL (if enabled) while connecting to Cartridge Management Web service** check box located at the bottom of the page.
8. Click **Next**.

   The Deployed Cartridges page displays.

   This page lists all of the cartridges that are currently deployed in UIM.

Using Cartridge Deployment Log for Analyzing Cartridge Deployment Errors

At the end of each cartridge deployment operation, the Cartridge Deployer Tool returns logs that display whether the cartridge deployment succeeds or fails. In case of any failure, detailed information about the specific error can be viewed from either the UIM server log or the **cartridgeDeployment** log.

You can find the *serverName*_**uim_cartridgeDeployment.log** file located under the *Domain_Home***/uim/logs** directory. Where, *serverName* is the host name for UIM and *Domain_Home* is the domain root directory.

About Cartridge Bundles

Cartridges can be bundled into a single deployable file. Deploying cartridge bundles offer two important advantages over deployable individual cartridges one at a time:

- The cartridges are deployed as a unit, so only a single redeployment of the **inventory.ear** file in the WebLogic server administration console is required.
- The cartridge bundle can be configured to ensure that cartridges are installed in the correct order to ensure that dependencies are met.

Cartridge bundles are JAR files that contain individual cartridges. For example, the cartridge bundle file for the Consumer VoIP cartridge pack (xxxx.jar) includes a number of individual cartridge files, such as yyyy.jar, zzzz.jar, and so on.

Creating Cartridge Bundles

In addition to the cartridge bundles provided by Oracle (such as those in cartridge packs), you can create your own bundles. For example, if you create a domain solution for your business in Design Studio, you can create a cartridge bundle to make deployment easier.

The Cartridge Deployer Tool determines the appropriate order in which the cartridges are deployed, based on the dependencies of each cartridge.

If the bundle of cartridges is successfully deployed, the message *your-bundle-name* **has been installed** appears. The list of cartridges that are contained in the bundle is displayed in the Installed Cartridges table, in the Cartridge Management page.

To create a cartridge bundle:

1. Create a directory called **META-INF/cartridges/**.
2. Copy the cartridge JAR files you want to bundle to the **META-INF/cartridges** directory.
3. Create a **MANIFEST.MF** file and include the following properties:

   Bundle-Name: *unique_name_for_this_bundle*

   Bundle-Version: *5-digit_version_number*

   Bundle-BuildTime: *yyy-mm-dd hh:mm:ss*

   Bundle-BuildNumber: *build_number*

   Bundle-Category: Inventory

   Bundle-Vendor: InventoryCartridge (Optional)

   Bundle-RequiredExecutionEnvironment: Oracle-Communications/Inventory-7.2.0, JavaSE-1.6 (Optional)

   Bundle-RequiredTargetVersion: 7.2.0 (Optional)

   For example:

   ```
   Manifest-Version: 1.0
   Ant-Version: Apache Ant 1.7.1
   Created-By: 20.9-b04 (Sun Microsystems Inc.)
   Bundle-Name: cableTV
   Bundle-Version: 7.2.2.0.0
   Bundle-BuildTime: 2012-10-05 11:07:28
   Bundle-BuildNumber: 1
   Bundle-Category: Inventory
   ```

4. Place the file into the **META-INF** directory.
5. Navigate to the parent folder of **META-INF**.
6. Open a command window.
7. Run the following command:

   ```
   jar cfM your-bundle-name.jar *.*
   ```

   where *your-bundle-name* is the name of the bundle to be created.

Automating Cartridge Bundling

The process of bundling multiple cartridges can be automated using Ant targets. Example 2-1 shows the **cartbuild.xml** file, which defines Ant targets that you can modify to bundle your own cartridges.

To modify the XML, and to run the Ant target defined within the XML:

1. Modify the following property name values:

- **bundle.name:** Set the value to the name of the cartridge that will contain the bundled cartridges. Do not include **.jar** in the name of the cartridge: when the cartridge is created is it automatically appended with **.jar**.
- **workspace.dir:** Set the value to path of your Eclipse workspace.
- **bundle.dir:** Set the value to path where the created cartridge is to reside.

2. Within the **copyfiles** Ant target, modify the <fileset> and <include> elements to reflect the names of the individual cartridges that you are bundling.
3. From a command prompt, navigate to the directory that contains the XML file and enter the following to run the Ant target:

```
ant -f cartbuild.xml
```

This command assumes that individual cartridges specified in the **copyfiles** Ant target were already built within Design Studio, and that you did not change the name of the XML file.

Example 2-1 cartbuild.xml

```
<project name="cartridgebundle" default="all" basedir=".">
    <property name="bundle.name" value="cart-bundle"/>
    <property name="workspace.dir" value="c:/eclipse_workspace"/>
    <property name="bundle.dir" value="d:/javatest/workspace/bundle"/>
    <property name="meta.dir" value="${bundle.dir}/META-INF"/>
    <property name="cartridges.dir" value="${meta.dir}/cartridges"/>

    <target name="clean"
     description="Deletes the bundle directory">
        <delete dir="${bundle.dir}"/>
    </target>

    <target name="makefolders"
     description="Create folders to contain temporaryfiles and target bundle jar">
        <mkdir dir="${bundle.dir}"/>
        <mkdir dir="${meta.dir}"/>
        <mkdir dir="${cartridges.dir}"/>
    </target>

    <target name="copyfiles" depends="makefolders"
     description="Copy selected cartridge jars to the temporary folder">
        <copy todir="${cartridges.dir}">
            <fileset dir="${workspace.dir}/custom-cartridge-1/cartridgeBin">
                <include name="custom-cartridge-1.jar"/>
            </fileset>
            <fileset dir="${workspace.dir}/ custom-cartridge-2/cartridgeBin">
                <include name="custom-cartridge-2.jar"/>
            </fileset>
        </copy>
    </target>

    <target name="makejar" depends="copyfiles"
     description="Create the target bundle jar">
        <jar compress="true" jarfile="${bundle.dir}/${bundle.name}.jar">
            <fileset dir="${bundle.dir}">
                <include name="**/*"/>
            </fileset>
        </jar>
    </target>

    <target name="clean.meta" depends="makejar"
```

```
     description="Deletes the temporary META-INF directory">
        <delete dir="${meta.dir}"/>
    </target>

    <target name="all"
     depends="clean, makefolders, copyfiles, makejar, clean.meta"
     description="Clean bundle folders and build the target bundle jar">
    </target>

</project>
```

Resolving Problems Related to Multiple Deployments

After you have deployed a number of cartridges to UIM, you may see error messages and be prevented from additional deployments. To resolve this problem, you must purge the DEV_MDS.MDS_COMPONENTS database table using one of the methods described below. This database table contains the metadata version history.

Purging Metadata Version History by Using Oracle Fusion Middleware Control

You can purge the metadata version history by using Fusion Middleware Control. You can also enable automatic purging of the data. See *Oracle Fusion Middleware Administrator's Guide* for information about launching and using Fusion Middleware Control.

To purge metadata version history by using the Fusion Middleware Control:

1. In the Fusion Middleware Control Farm home page, expand **Application Deployments** and select **oracle.communications.inventory**.
2. From the **Application Deployment** menu, choose **MDS Configuration**.

   The MDS Configuration page is displayed.
3. In the Purge section, enter a number and select the unit of time in the **Purge all unlabeled past versions older than** field.

   For example, enter **3** and select **days**.
4. Click **Purge**.
5. In the **Confirmation** dialog box, click **Close**.

To enable automatic purging in Fusion Middleware Control:

1. In the Fusion Middleware Control Farm home page, navigate to the WebLogic domain and select it.
2. From the **WebLogic Domain** menu, choose **System MBean Browser**.

   The System MBean Browser page appears.
3. Expand **Application Defined MBeans**, then **oracle.adf.share.config**, then **Server: name**, then **Application: oracle.communications.inventory**, then **ADFConfig**, then **ADFConfig, and ADFConfig.Select MDSAppConfig**.

   The Application Defined MBeans page appears.
4. For **AutoPurgeTimeToLive**, enter a value in seconds.
5. Click **Apply**.

Purging Metadata Version History by Using the WLST Script

You can purge metadata version history by using the **purgeMetadata** command in the WLST script. You must run the script from the appropriate Oracle home. Do not run the script from the WebLogic Server home. The script is located in the following directory.

*WL_Home***/common/bin**

You specify the documents to be purged by using the **olderThan parameter**, specifying the number of seconds. The following example purges documents older than 100 seconds.

```
purgeMetadata(application='oracle.communications.inventory',server='AdminServer,
olderThan=100)
```

Guidelines and Recommended Cartridge and Cartridge Pack Deployment

If unexpected issues occur while you deploy a cartridge read the message that displays to troubleshoot the cause of the issue.

Common deployment errors occur for the following reasons:

- The Design Studio project from which the cartridge was built has an error. Check the Problems view in Design Studio for error information, and then correct any problems before redeploying the cartridge into UIM. See *SCD Design Studio Modeling Inventory* for more information.
- The UIM database data model does not match the version of the data model that Design Studio used to build the cartridge.

  If you build a cartridge that relies on features found in an older version of the data model, deploying the cartridge into a UIM instance with a newer data model can result in deployment errors. Ensure that the data model that you use is the same version in both Design Studio and UIM.
- The UIM version is not compatible with the version of Design Studio used to build the cartridge.

  In this situation, the UIM cartridge reader encounters information it cannot handle and throws an error. Make sure you use the version of Design Studio and the Inventory plug-in that corresponds to your UIM version. If necessary, rebuild cartridges with the correct Design Studio version. You can see the Design Studio target version in the **Target Version** field of the **inventoryCartridge** file for the project.

  Design Studio operates without a connection to the UIM database, so it is possible to create cartridges that are structured correctly in Design Studio but do not deploy properly into UIM. Ensure that the target UIM environment is configured with the correct schema, parameters, and base data that match the version of Design Studio being used.
- The cartridge being deployed is dependent on another cartridge that has not yet been deployed.

  Correct this by deploying the base or parent cartridge and then redeploying the cartridge that is dependent on the parent cartridge. A message displays telling you which cartridges are required if you try to deploy a child cartridge without the base.

In some cases, cartridges can be deployed without errors but still cause inconsistencies or unexpected behavior in UIM. These issues can be avoided by following the recommended

guides for upgrading and extending cartridges and cartridge packs. See "Upgrading and Extending Cartridges and Cartridge Packs " for more information.

3

Upgrading and Extending Cartridges and Cartridge Packs

This chapter provides information on upgrading and extending cartridges and cartridge packs to include entity specifications, characteristics, rulesets, and other artifacts to extend the content and functionality of Oracle Communications Unified Inventory Management (UIM). You can create your own cartridges and cartridge packs in Design Studio or obtain samples from Oracle. Both cartridges you create and those supplied by Oracle can be upgraded or extended after they are deployed.

> **Note**
>
> Do not unseal or modify the base cartridges that are supplied with UIM.

When you upgrade or extend a cartridge, there is the possibility of unexpected results. To ensure data consistency, you must follow the guidelines and leading practices provided in this chapter.

> **Note**
>
> Redeploy the cartridges with Drools based rulesets before you run any of the Drools based rulesets. This is a one time activity after UIM upgrade.

How UIM Data Is Affected by Cartridge Changes

When you deploy a new version of a cartridge into UIM, existing specifications, characteristics, and rulesets are overwritten with the content in the new version. UIM uses names as unique identifiers. So when a specification, characteristic, or ruleset is deployed, it overwrites any existing version with the same name. Only the latest version is maintained in UIM. The origin of the overwriting specification, characteristic, or ruleset is irrelevant. It can come from a new version of the original cartridge or from an entirely separate cartridge.

Entity data from an earlier version of a cartridge is *not* changed when you deploy an updated cartridge, however. For example, if you created Logical Device entities based on a specification called Customer Edge Router, the entity data for those logical devices continues to exist in its original form when a new version of the specification is deployed following a cartridge upgrade.

But because a new version of the specification has replaced the old, the new version is used to display entities in UIM, even when the entities were created with an older version of the specification. As a result, there can be mismatches between the data and the new specification used to display it. Some data may no longer be visible in UIM, even though it still exists in the database. Similarly, data for characteristics introduced in the new version of a specification will be absent for entities created with the previous version.

For example, suppose the old version of the Customer Edge Router specification included a characteristic called Route Distinguisher Group, which has been removed from the new version. When the new version is installed, the data that populated that characteristic continues to exist in the database but is not displayed.

If the new version of the Customer Edge specification includes the characteristic Management IP Address and populated automatically by a ruleset when an entity is created, existing entities based on the old version of the specification will not include that information.

Leading Practices for Extending Cartridges

The section includes guidelines that you should follow to ensure that your extensions can be maintained even when a cartridge or cartridge pack is upgraded. Although these guidelines are oriented toward Oracle-supplied cartridges, many of the same considerations apply to cartridges that you create yourself.

There are two basic approaches to extending a cartridge or cartridge pack:

- Making additive changes in separate cartridges
- Modifying the content of renamed cartridges (“clone and own")

In general, it is preferable to make additive changes. Because you are not modifying cartridges directly, the risk of complications during upgrade is reduced. See "Extending Cartridges By Adding New Content in Separate Cartridges". There are circumstances when you must unseal, rename, and modify cartridges, however. See "Renaming and Modifying Cartridges".

> **Note**
>
> You should make sure to deploy only tested, stable extensions to your production environment, especially when you are modifying cartridges.

Extending Cartridges By Adding New Content in Separate Cartridges

In many cases, you can extend a cartridge or cartridge pack by creating a new cartridge that includes specifications and other artifacts that supplement the default content.

Because artifacts created in this way are separate from and named differently from the default contents, they will be preserved when you update the cartridge pack to a new version.

Inter-cartridge relationships make it possible to add separate content that is closely tied to the original content. You can establish relationships between specifications in one cartridge and specifications in another cartridge. In this scenario, you do not need to unseal or modify the original content of a cartridge pack. See *SCD Design Studio Modeling Inventory* for more information about implementing relationships among specifications in different cartridges.

Inter-cartridge relationships create dependencies between the cartridges. See "Tracking Cartridge Dependencies" for more information about dependencies. If you create an inter-cartridge relationship, deploy the cartridges, and subsequently redeploy or update the cartridge that includes the relationship target, the relationship is broken. You must re-establish it.

When you develop content in separate cartridges, you must be careful to name the corresponding Design Studio project uniquely to avoid conflicts. The project name must be unique in the workspace and among all cartridges that will be deployed to UIM.

The following list includes examples of extensions you can make additively:

- You can add new global rules in separate cartridge to define new behaviors. Global rules are triggered at an extension point that applies to all specifications.
- You can establish an assignment or reference from a configuration item in one cartridge to a specification in a separate cartridge by creating an upward reference from the specification.
- You can use an upward reference to associate a new configuration specification in a separate cartridge to a configuration-enabled specification in another cartridge. For example, you can associate a new Service Configuration specification that you create with a Service specification in an Oracle-supplied sample cartridge.
- You can use upward reference to associate related specifications in a separate cartridge to specifications in a another cartridge. For example, you can associate a new Device Interface specification that you create to a Logical Device specification defined in a sample cartridge.
- You can add Inventory Group specifications in separate cartridge that you can use to group entities from other cartridges. For example, you could create inventory groups that you use to group MMS Server and Voice Mail Server entities for the GSM 3GPP sample cartridge pack.

Renaming and Modifying Cartridges

Some types of extensions cannot be achieved by additive changes in separate cartridges. In these situations, you must unseal the affected cartridge, rename it, and then modify the necessary artifacts.

Renaming the cartridge before modifying it is a good practice because it differentiates the modified cartridge from the original. When you deploy the cartridge to UIM, for example, you can tell at a glance that it is the modified version.

See *SCD Design Studio Modeling Inventory* for detailed instructions about unsealing and renaming cartridges.

The following list includes examples of situations when you need to modify cartridge contents

- Adding a characteristic or attribute to an existing specification. The characteristic can be in a separate cartridge, but associating the characteristic to the specification requires unsealing the cartridge in which the specification is defined.
- Removing a characteristic or attribute from an existing entity.
- Changing existing Java logic.
- Adding a new configuration item to a configuration specification.
- Deleting a configuration item from a configuration specification. This option may require the database to be cleaned out, or the specification may need to be versioned.
- Adding a new specification-based ruleset (as opposed to a global ruleset).
- Updating an existing specification-based ruleset.

Observe the following guidelines when you unseal and rename a cartridge:

- Keep a backup copy of the original cartridge.
- To avoid naming conflicts among specifications, characteristics and other artifacts, do not deploy the original cartridge to UIM or open it in Design Studio at the same time as the renamed version.

- Ensure that all cartridges on which the sample cartridge has a dependency are open in Design Studio before you rename it. See "Tracking Cartridge Dependencies" for more information.

Considerations When Modifying Cartridge Content After Entity Data Creation

You should consider the possible consequences of modifying the content of a cartridge after it has been deployed to UIM and after entity data has been created.

Modifying Artifacts

Making changes to existing specifications, characteristics, and other artifacts can cause issues when entity data has already been created. Problems can occur with the cartridge deployment or with UIM data.

For example, changing the type of a characteristic can cause difficulty with updating values assigned to it. Suppose you use Design Studio to change an existing required characteristic from a text field to a list. You deploy the new version of the cartridge successfully. When you open an entity that uses this characteristic in UIM, it displays its original value. If you try to modify the entity, however, this value will be lost because UIM forces you to set a value based on its new definition as a list with a limited set of values.

Deleting Artifacts

Deleting a specification, characteristic, or ruleset in a new version of a cartridge does not delete data that has been created in UIM based on it.

For example, if you delete a specification in Design Studio after entities based on it have been created in UIM, neither the specification itself nor the entities based on it are removed when you deploy the new version of the cartridge.

It is possible to inactivate a specification from UIM, but this requires that you delete all entities based on it and all entities related to those entities. Only then can you inactivate the specification. See "Getting Started with Unified Inventory Management" in UIM Help for instructions about deleting entities and specifications.

For example, if you try to entirely inactivate a specification called PE Router without first deleting entities based on it, you see an error message. You must search for and delete all logical devices based on PE Router. If any of these devices includes device interfaces, they must also be deleted. Characteristics included in the specification do not need to be deleted.

Unlike specifications, which are not removed in UIM when they are deleted in a cartridge, relationships between entities in UIM *are* removed when those relationships are deleted in a new version of a cartridge. For example, suppose the CE Router specification and the CE Router Interface specification are related to each other. As a result you can create logical devices and interfaces based on that relationship in UIM.

If you deploy new versions of the specifications that lack this relationship, the relationship is also removed from UIM entities based on the specifications.

Renaming Specifications and Characteristics

If you rename a characteristic or specification in Design Studio and include the renamed versions in an updated cartridge that you deploy into UIM, the new versions of the artifacts do not replace the old ones. Because UIM uses the specification or characteristic name as the unique identifier, the renamed version appears to be new and therefore does not overwrite the existing version. This can result in issues with the display of data.

If you use Design Studio to rename a specification after entities based on it have been created in UIM, deploying the upgraded cartridge results in the following:

- A new specification with the new name is created in UIM.
- The original specification remains in UIM.

If you use Design Studio to rename a characteristic that is used in entities already created in UIM, deploying the upgraded cartridge results in the following:

- The original characteristic is removed from the specification.
- The renamed characteristic is added to the specification.
- The data from the original characteristic remains in the database but does not appear in the new characteristic.

For characteristics, you can change the label without changing the name. The label is the text that is displayed in UIM. So if you need to change the displayed text but leave existing data in place, you can change the characteristic label. See *SCD Design Studio Modeling Inventory* for more information.

Actions to Take After Renamed Cartridge Has Been Deployed

If you cannot avoid renaming and redeploying a cartridge that has already been deployed under its original name, there are steps you should take to avoid potential problems.

> **Note**
>
> The information in this section is appropriate only for test and development environments. In a production environment, you should only deploy stable and tested extensions.

Problems arise because redeploying a cartridge with a new name after it has previously been deployed results in duplicate Java classes in UIM with the same package name. Errors can result when these classes are called, for example, from a ruleset.

The recommended approach to avoid these errors when a cartridge is renamed after it has already been deployed is to reinstall the UIM server and clear the database (essentially a fresh install with a clean database).

An alternative but less reliable method is available in the WebLogic Server Admin Console. In the Deployments screen, examine the **uim_custom_lib.ear** file in the current version of the **UIM/app/uim_custom_lib/uim_custom_lib**_*version* directory.

Stop the server, remove the original cartridge JAR file, remove the server temporary files, and then restart the server.

Tracking Cartridge Dependencies

Cartridges can have complex dependencies that you need to track. For example, suppose you include a specification from Cartridge A as a specification option in a configuration item in Cartridge B. In this situation, Cartridge B is dependent on Cartridge A. You must deploy Cartridge A before Cartridge B. Larger numbers of cartridges obviously lead to more complex dependencies.

Design Studio displays these dependencies in the **Dependency** tab of Inventory Project editors and prevents circular dependencies. UIM enforces these dependencies when you

deploy cartridges. You see an error if you try to deploy a cartridge before all of its dependent cartridges have been deployed.

You should be aware of dependencies when you extend cartridges. For example, if you deploy an updated version of Cartridge A, you know that there is a possible impact on Cartridge B.

Extensions may also affect the order in which you deploy cartridges. For example, if you create a cartridge to supplement a cartridge pack, dependencies will probably mean that you have to deploy it last.

See "Unified Inventory Management System Administration Overview" in *UIM System Administrator's Guide* and "Overview" in *UIM Developer's Guide* for information about grouping and deploying multiple cartridges.

Using Ruleset Stubs to Customize Validation and Allocation

Some cartridges include ruleset stubs that enable you to customize the behavior of entities without changing specifications. These ruleset stubs make it possible to customize validation and allocation functionality.

A ruleset stub is an empty ruleset that performs no operations by default. The full stub implementation includes the ruleset itself (**.ruleset** file), a rule file (**.drl** or **.groovy** file), an association to a specific extension point, and a reference to the ruleset extension point from a particular specification.

To implement your customized behavior, you define a ruleset that has the same name as the stub. You include the ruleset in a cartridge and deploy it into UIM after deploying the original cartridge. The customized ruleset overwrites the empty original.

For example, a cartridge could include a ruleset called AUTO_ALLOCATE_SERVICE_CONFIG that is associated with an extension point called AUTO_ALLOCATE_SERVICE_CONFIG_Ext. This extension point is in turn included in a Service Configuration specification.

To take advantage of the stub, you use Design Studio to create a cartridge that includes a ruleset also named AUTO_ALLOCATE_SERVICE_CONFIG.

> **Note**
>
> Design Studio requires that all specifications and rulesets be uniquely named within a workspace. The original cartridge must not be open in the workspace (even if it is sealed) when you define the replacement ruleset.

This ruleset includes the auto-allocation logic you want to implement. When you deploy the cartridge into UIM, the customized ruleset replaces the original, meaning that your logic is used for auto-allocation in entities based on the relevant Service Configuration specification.

If you deploy a new version of the original cartridge to UIM, the stub ruleset overwrites the customized version. You can re-deploy the cartridge with the customized ruleset to restore the desired functionality.

See *UIM Developer's Guide* and *SCD Design Studio Modeling Inventory* for more information about rulesets.

Use Manual Versioning to Update Specifications

There may be cases where you want to provide a new version of an existing specification. For example, suppose you have been using a specification for a physical device and the manufacturer comes out with a new model of the device. Changing the original specification runs all of the risks associated with modifications, so you should copy the specification, assign it a name that indicates its status, and introduce the required changes. For example, if you have an existing Physical Device specification called LRI_PERouter, you could define an updated specification called LRI_PERouter-v2.

If related specifications and characteristics need to change along with the parent specification, you can manually version them, too. For example, if a Logical Device specification has an associated Device Interface specification, you can define a version of the Device Interface specification to reflect the update to the logical device.

Use Database Snapshots to Allow Rollbacks of Upgrades

When you deploy a new version of or make significant content changes to a cartridge, you should save a snapshot of the database. This enables you to roll back the upgrade if the new content causes unexpected or undesirable effects.

It is not necessary to create a snapshot for simple changes such as adding a small number of new specifications.

You should also make sure to maintain backup copies of original cartridges so that you can redeploy content if necessary.

You should also back up important data that is stored in the UIM file system, such as Java code and the cartridge deployment history. See "Unified Inventory Management System Administration Overview" in *UIM System Administrator's Guide* and "Overview" in *UIM Developer's Guide* for information about these files.

Special Considerations for Configuration Versions

There are some special considerations involving entity configurations that you should keep in mind while planning upgrades to cartridges. See "About Unified Inventory Management" in *UIM Concepts* and "Getting Started with Unified Inventory Management" in *UIM Help* for information about using configurations.

As you create new configuration versions for a parent entity in UIM, you create a record of the state of the entity at various points in time. If you need to update a configuration specification, you have to choose between the potential for requiring transformation of data and the loss of configuration version continuity.

- If maintaining continuity of configuration versions is not essential, you can follow the standard practice recommended for entities: set an end date for the configuration specification and define a new one containing the updates. This option is suitable for in-progress configurations. Because associations to configuration specifications cannot be changed after configurations have been created, however, you will need to recreate the affected parent entities and then create the first configuration version using the new configuration specification.
- If maintaining the continuity of configuration versions is important, you should modify the configuration specification rather than replace it. Such modifications result in the kinds of issues discussed in "Considerations When Modifying Cartridge Content After Entity Data Creation", however: the data in older configuration versions will be displayed from the

perspective of the modified specification. If you need to access data that would otherwise not be visible as a result of the modifications, you need to transform the data to conform to the new specification. This may be disruptive of in-progress configurations and require tightly synchronized changes to upstream systems.

Adopt a Systematic Naming Policy

Because specifications and characteristics must be uniquely named in a Design Studio workspace and in a UIM database, you should adopt a careful and systematic naming convention. If you are not careful about naming, you run the risk of overwriting an existing specification or characteristic or of having one of your specifications or characteristics overwritten when a cartridge is deployed.

For example, assume that a Logical Device specification called PE Router is included in an Oracle sample cartridge. If you build your own cartridge that includes an identically named specification, the specification in the sample will be overwritten when you deploy your cartridge.

Follow these guidelines to avoid naming issues:

- Use specification and characteristic names that are unique and not generic. For example, you could include an identifying prefix to all specifications and characteristics you create.
- Do not rename characteristics and specifications after entities based on them have been created in UIM. If you want to stop using a specification or characteristic of a particular name and use a similar one with a different name, you should create the new version and set an end date for the old version in Design Studio. After deploying the cartridge containing the changes, migrate entity data as necessary.

Recommended Cartridge Life Cycle

This section outlines the recommended life cycle of a Oracle-supplied sample cartridge or cartridge packK.

1. You download the sample cartridge or cartridge pack.
2. If necessary, you develop the new specifications, rulesets, and other artifacts required by your business. See "Leading Practices for Extending Cartridges" for more information.

   For example, you can define vendor-specific specifications for the particular networking equipment that you use. You package these extensions in separate cartridges that you deploy after deploying the sample cartridge pack.
3. You deploy the sample cartridge or cartridge pack along with your extensions to it.

   Cartridge packs typically comprise several cartridges with dependencies on each other. You must deploy the cartridges in a specific order. If you are deploying cartridges by using a cartridge bundle, the deployment order is handled automatically.
4. You create inventory entities based on the sample and your extensions.
5. At a later date, Oracle upgrades the sample cartridge or cartridge pack.
6. You evaluate the enhancements included in the upgrade and plan any changes to your extensions that are required to take advantage of the new capabilities.
7. You deploy the upgraded cartridge or cartridge pack.

   Because your extensions followed the recommended guidelines, the upgrade does not interfere with data you created using the previous version.
8. You deploy any new extensions of the sample along with updates to existing extensions.

9. You create inventory entities based on the upgraded sample and extensions.

4

Common Content

This chapter describes the utility methods included in the Oracle Communications Unified Inventory Management (UIM) common cartridge (ora_uim_common). This cartridge provides modeling capabilities that are independent of the technology domain.

The common cartridge includes utility methods that provide operations across multiple cartridge packs. Refer to the Javadoc for detailed information about the utility methods.

The utility methods in the cartridge are bundled into the following classes:

- CommonManager
- ResourceManager
- ServiceManager

The **ora_uim_common.jar** provides a library that resolves references from the **ora_uim_common** cartridge. You use this library to develop new domain cartridges or to enhance other cartridges. Save this library in the same location as the path you defined for the UIM_LIB variable in Oracle Communications Service Catalog and Design - Design Studio. See *SCD Design Studio Modeling Inventory* for more information.

CommonManager

The CommonManager class represents the common interface for the use of a cartridge pack.

Table 4-1 describes the utility methods included in the CommonManager class.

Table 4-1 Utility Methods in CommonManager

Method Signature	Description
boolean checkPartyServiceRel (ServiceConfigurationVersion scv, Party party)	Checks whether the service is associated with a party.
boolean checkPartyServiceRel (ServiceConfigurationVersion scv, PartyType subscriberParam)	Checks whether the service is associated with a party. For this method, the input parameter is PartyType.
void commitAndStartTransaction()	Commits an open transaction.
GeographicAddress createAddress (GeographicAddressType addressType, java.lang.String specName)	Creates a geographic address based on the address, city, and state information.
GeographicLocation createGeoLocationForServiceAddress (GeographicAddress address)	Creates a service location for the given service address.
<E extends Persistent> E createLiveEntity (Persistent entity)	Creates an entity in live context.
Party createParty (PartyType partyType, java.lang.String partySpecName, java.lang.String roleSpecName)	Creates a party based on the name, specification, and account number information.
java.util.List<GeographicAddress> findAddress (GeographicAddressType addressType)	Finds a service address based on the address, city, and state information.

Table 4-1 (Cont.) Utility Methods in CommonManager

Method Signature	Description
java.util.List<InventoryGroup> findAndValidateInventoryGroup (java.lang.String inventoryGroupName, java.lang.String specificationName)	Finds all inventory groups based on the name of the inventory group and its specification.
Specification findAndValidateSpecification (java.lang.Class specClass, java.lang.String specificationName)	Finds a specification based on the name of the specification and validates it. This method requires the name of the specification class as an input parameter.
Specification findAndValidateSpecification (java.lang.String specificationName)	Finds a specification based on the name of the specification and validates it.
java.util.List<InventoryGroup> findInventoryGroup (java.lang.String inventoryGroupName, java.lang.String specificationName)	Finds all the inventory groups based on the inventory group name and specification.
GeographicLocation findLocationForServiceAddress (GeographicAddress serviceAddress)	Finds a service location for the given service address. If a service location is not found, it returns null.
java.util.List<Party> findParty (PartyType partyType)	Finds a party based on the party name, specification, and account number information.
java.util.List<InventoryGroup> findServingAreaForServiceAddress (GeographicAddress serviceAddress, java.lang.String placeCharName, java.lang.String invGroupSpecName)	Finds inventory groups that are defined for a serving area.
GeographicAddress getAddress (GeographicAddressType addressType, java.lang.String specName)	Finds an address entity based on the given parameter type. When an address is not found, it creates one address with the given information in the parameter type.
GeographicLocation getLocation (GeographicAddress address)	Finds a geographical location entity based on the given address. When a location is not found, it creates a location and relates it to the address.
Party getParty (PartyType partyType, java.lang.String partySpecName, java.lang.String roleSpecName)	Finds a party entity based on the given parameter type. When a party is not found, it creates one party with the given information in the party parameter.
CharValue makeCharValue (characteristicExtensible, java.lang.String charSpecName, java.lang.String value)	Creates a characteristic value.
void validateInventoryGroupName (InventoryGroup inventoryGroup, java.lang.String specificationName)	Checks whether an inventory group exists for an instance of this specification.

ResourceManager

The ResourceManager class represents the common interface for resource management of a cartridge pack.

Table 4-2 describes the utility methods included in the ResourceManager class.

Table 4-2 Utility Methods in ResourceManager

Method Signature	Description
PhysicalDevice createPhysicalDevice (java.lang.String physicalDeviceName, java.lang.String specificationName, java.util.Collection<PhysicalDeviceCharacteristic> pdCharacteristics)	Creates a physical device based on name and specification.

Table 4-2 (Cont.) Utility Methods in ResourceManager

Method Signature	Description
PhysicalDevice createPhysicalDevice (java.lang.String physicalDeviceName, java.lang.String specificationName, java.lang.String serialNumber, java.lang.String physicalAddress, java.lang.String physicalLocation, java.util.Collection<PhysicalDeviceCharacteristic> pdCharacteristics)	Creates a physical device based on name and specification. The hard attributes serial number, physical address, and physical location are additional input parameters for this method.
java.util.List<LogicalDevice> findAndValidateLogicalDevice (java.lang.String logicalDeviceName, java.lang.String specificationName)	Finds all the logical devices based on the name of the logical device and its specification.
java.util.List<PhysicalDevice> findAndValidatePhysicalDevice (java.lang.String physicalDeviceName, java.lang.String specificationName)	Finds all the physical devices based on the name of the physical device and its specification and validates them.
java.util.List<PhysicalDevice> findAndValidatePhysicalDeviceByPhysicalAddress (java.lang.String physicalAddress, java.lang.String specificationName)	Finds all the physical devices based on the physicalAddress characteristic and its specification.
java.util.List<LogicalDevice> findLogicalDevice (InventoryGroup group, java.lang.String specName)	Finds all the logical devices in the given inventory group.
java.util.List<LogicalDevice> findLogicalDevice (java.lang.String logicalDeviceName, java.lang.String specificationName)	Finds all the logical devices based on the name of the logical device and its specification name.
java.util.List<LogicalDeviceAccount> findLogicalDeviceAccountByChar (java.lang.String charName, java.lang.String charValue, java.lang.String specName, LogicalDevice ld)	Finds all logical device accounts based on a characteristic. If a logical device account is not found, it returns either null or an empty list.
java.util.List<LogicalDevice> findLogicalDeviceByChar (java.lang.String charName, java.lang.String charValue, java.lang.String specName)	Finds all logical devices based on a characteristic.
java.util.List<PhysicalDevice> findPhysicalDevice (java.lang.String physicalDeviceName, java.lang.String specificationName)	Finds all the physical devices based on the name of a physical device and the specification.
java.util.List<PhysicalDevice> findPhysicalDeviceByPhysicalAddress (java.lang.String physicalAddress, java.lang.String specificationName)	Finds physical devices in the inventory based on the physical address (MAC address) and specification.
void validateLogicalDeviceName (LogicalDevice logicalDevice, java.lang.String specificationName)	Checks whether a logical device already exists for an instance of this specification.
void validatePhysicalDeviceName (PhysicalDevice physicalDevice, java.lang.String specificationName)	Checks whether a physical device already exists for an instance of this specification.

ServiceManager

The ServiceManager class represents the common interface for the service and service configuration management of a cartridge pack.

Table 4-3 describes the methods included in the ServiceManager class.

Table 4-3 Utility Methods in ServiceManager

Method Signature	Description
addChildConfigItem (ServiceConfigurationVersion scv, ServiceConfigurationItem parentItem, java.lang.String childItemName)	Adds a child configuration item in the given configuration version.
addChildConfigItem (ServiceConfigurationVersion scv, java.lang.String parentItemName, java.lang.String childItemName)	Adds a child configuration item in the given configuration version. For this method, the name of the parent service configuration item, rather than the configuration item itself, is passed as an input parameter.
checkItemAssignedReferenced (ServiceConfigurationVersion scv, ServiceConfigurationItem item)	Checks whether the configuration item has an assignment or reference.
checkItemAssignedReferenced (ServiceConfigurationVersion scv, java.lang.String itemName)	Checks whether the configuration item has an assignment or reference. For this method, the name of the service configuration item, rather than the configuration item itself, is passed as an input parameter.
findServiceConfigItemByName (ServiceConfigurationVersion scv, java.lang.String itemName)	Finds service configuration items by the name of a configuration item in the given service configuration version.
getAllChildServiceConfigurationItems (java.util.List<ServiceConfigurationItem> items)	Retrieves all the levels of child configuration items for the given configuration items.
getAllChildServiceConfigurationItemsMatching (ServiceConfigurationItem item, java.lang.String matchingName)	Retrieves the child configuration items for the given configuration item that match with the given name.
getServiceConfigurationItems (ServiceConfigurationVersion scv, Assignment assignment)	Finds the service configuration items based on the assigned resources.
getServiceConfigurationItems (ServiceConfigurationVersion scv, ConfigurationReference reference)	Finds the service configuration items based on the referenced resources.
getServiceConfigurationVersion (Service svc, java.lang.String configSpecName)	Finds whether a service configuration version that is in progress exists before a new version can be created.
relateServiceToParty (ServiceConfigurationVersion scv, Party party, java.lang.String roleSpec)	Relates a party to the service.
relateServiceToParty (ServiceConfigurationVersion scv, PartyType partyType, java.lang.String partySpecName, java.lang.String roleSpec)	Relates a party to the service. For this method, the input parameter is PartyType.
updateServiceConfigurationVersion (ServiceConfigurationVersion scv)	Updates the service configuration version for the given configuration version and returns the updated version.

5

Base Cartridge Reference

This chapter provides reference information about the Oracle Communications Unified Inventory Management (UIM) base cartridges. Base cartridges provide fundamental data that is referenced by other cartridges and cartridge packs.

The chapter is organized by cartridge, with each section documenting the contents of a cartridge or group of cartridges.

- Base Specifications Cartridge
- Base Measurement Entities
- Phone Management Specifications and Rulesets
- Telephone Number Specifications and Rulesets
- Base Technologies Cartridge

Base Specifications Cartridge

The Base Specifications cartridge (ora_uim_basespecifications) includes a variety of specifications designed for particular purposes.

Business Interaction Specifications

Table 5-1 lists and describes the Business Interaction specifications included in the Base Specifications cartridge.

Table 5-1 Business Interaction Specifications

Specification	Description
Service Order	Defines the inventory actions that are required to fulfill a service order. This specification contains only default data elements.

Connectivity Specifications

Table 5-2 lists and describes the Connectivity specifications included in the Base Specifications cartridge.

In releases prior to UIM 7.3, you could not create multiple different Connectivity specifications. Instead, there was a single specification provided in the cartridge. All entities created in UIM were based on that specification and you could modify it only by adding entity level characteristics.

For backward compatibility, that specification still exists but has been renamed from TDM Facility to Channelized Facility.

Table 5-2 Connectivity Specifications

Specification	Description
Channelized Facility	Represents a channelized transport that is a carrier of other connectivities. A Channelized Facility provides channels, that can be assigned to the connectivities (riders) that the transport carries.

Custom Involvement Specifications

Table 5-3 lists and describes the Custom Involvement specifications included in the Base Specifications cartridge.

Table 5-3 Custom Involvement Specifications

Specification	Description
Manages (Oracle Provided)	Represents an association in which one resource manages another.
Preconfigure	Represents an association created in advance among entities. Preconfigured resources share the same life cycle: when one of the resources is reserved or assigned, the other is automatically reserved or assigned with it. For example, in a DSL scenario with telephone service, you can an involvement to preconfigure a telephone number, switch port, and cable pair.

Inventory Group Type Specifications

Table 5-4 lists and describes the Inventory Group Type specifications included in the Base Specifications cartridge.

Table 5-4 Inventory Group Type Specifications

Specification	Description
Flow Identifier Resource Pool	When associated with an Inventory Group specification, defines the inventory group as a resource pool for flow identifiers such as VLAN IDs, VPLS IDs, and so on.
IP Address Resource Pool	When associated with an Inventory Group specification, defines the inventory groups as a resource pool for IP addresses.

IP Resource Specifications

Table 5-5 lists and describes the IP Resource specifications included in the Base Specifications cartridge.

Table 5-5 IP Resource Specifications

Specification	Description
IPv4 Address Base	Represents an IPv4 address resource that can be assigned to a service configuration item.
IPv4 Network Base	Represents an IPv4 network, which is the foundation of IP address management. An IPv4 network contains one or more IPv4 subnets, each of which contains IPv4 addresses or ranges of IPv4 addresses.

Table 5-5 (Cont.) IP Resource Specifications

Specification	Description
IPv4 Subnet Base	Represents an IPv4 subnet, which is used in IP address management. An IPv4 subnet is part of an IPv4 network. An IPv4 subnet can be partitioned into smaller, more manageable subnets, each of which contains IPv4 addresses or ranges of IPv4 addresses.
IPv6 Address Base	Represents an IPv6 address resource that can be assigned to a service configuration item.
IPv6 Network Base	Represents an IPv6 network, which is the foundation of IP address management. An IPv6 network contains one or more IPv4 subnets, each of which contains IPv6 addresses or ranges of IPv6 addresses.
IPv6 Subnet Base	Represents an IPv6 subnet, which is used in IP address management. An IPv6 subnet is part of an IPv6 network. An IPv6 subnet can be partitioned into smaller, more manageable subnets, each of which contains IPv6 addresses or ranges of IPv6 addresses.

Managed Project Specifications

Table 5-6 lists and describes the Managed Project specification included in the Base Specifications cartridge.

Table 5-6 Managed Project Specifications

Specification	Description
Base Project	Defines the entity used to create Project entities in UIM.

Network Specifications

Table 5-7 lists and describes the Managed Project specification included in the Base Specifications cartridge.

Table 5-7 Network Specifications

Specification	Description
SONET Network (Oracle Provided)	Defines a network based on the SONET technology. SONET networks created in versions earlier than UIM 7.3 are automatically assigned to this specification during the upgrade process.
SDH Network (Oracle Provided)	Defines a network based on the SDH technology. SDH networks created in versions earlier than UIM 7.3 are automatically assigned to this specification during the upgrade process.

Property Location Specifications

Table 5-8 lists and describes the Property Location specifications included in the Base Specifications cartridge. Unlike most other specifications, you cannot create multiple different Property Location specifications. Instead, you use entity-level characteristics to modify the single specification provided in the cartridge. All entities created in UIM based on that specification share those characteristics. See *SCD Design Studio Modeling Inventory* for more information about entity-level characteristics.

Table 5-8 Property Locations Specifications

Specification	Description
Property Location	Represents a piece of land with defined legal boundaries. Property location is the lowest-level element in the natural hierarchy that identifies a location (country, state/province, municipality, property location). Property location is used in defining network locations for channelized connectivity.

Base Measurement Entities

This section contains information about the entities included in the Base Measurements cartridge (ora_uim_basemeasurements). The Base Measurements cartridge provides measurement types, units of measure, and capacity types to support the definition of signal structures and pipe capacity in UIM.

For example, the Base Measurements cartridge provides a Bit Rate measurement type; a set of related units of measure including bps, Kbps, Mbps, and Gbps; and a Bandwidth capacity type. See *UIM Concepts* for more information about measurement entities.

Bandwidth is the only type of capacity that can be used by default. You can use extension points and rulesets to extend the application to handle requirements that are specific to another type of entity or technology. For more information on extension points and rulesets, see *UIM Developer's Guide*.

Capacity Types

Table 5-9 lists and describes the capacity types included in the Base Measurements cartridge.

Table 5-9 Capacity Types

Specification	Description
Bandwidth	Identifies the capacity type used to measure the capacity of pipes. Bandwidth is measured by bit rate. Entities such as pipes provide and consume bandwidth at a bit rate that you specify. For example, a T3 pipe can provide 43 Mbps, for example, and a T1 pipe can consume 1.544 Mbps of that capacity.

Measurement Types

Table 5-10 lists and describes the measurement types included in the Base Measurements cartridge.

Table 5-10 Measurement Types

Specification	Description
Bit Rate	Identifies the unit of measure for the Bandwidth capacity type

Units of Measure

Table 5-11 lists and describes the units of measure included in the Base Measurements cartridge.

Table 5-11 Units of Measure Specifications

Specification	Description
Bits Per Second	Denotes the number of bits of data that can be transferred over a communications channel in one second
Gigabits Per Second	Denotes the number of gigabits of data that can be transferred over a communications channel in one second
Kilobits Per Second	Denotes the number of kilobits of data that can be transferred over a communications channel in one second
Megabits Per Second	Denotes the number of megabits of data that can be transferred over a communications channel in one second

Phone Management Specifications and Rulesets

This section contains information about the characteristics and rulesets included in the Base Phone Management cartridge. The Base Phone Management cartridge provides characteristics and rulesets that help distinguish telephone numbers based on type. The cartridge also enables UIM to manage the assignment state of telephone numbers, including ported telephone numbers. This ability enables UIM to support telephone number portability features.

Characteristics

The Base Phone Management cartridge provides characteristics, including a telephone number type characteristic that provides values to identify the following types of telephone numbers:

- Owned: Telephone numbers owned by a service provider
- Ported in: Telephone numbers owned by a different service provider but used by a customer after subscribing to the current service provider
- Ported out: Telephone numbers owned by the original service provider but used by a customer after subscribing to a different service provider
- Toll free: Telephone numbers for which the service provider, instead of the customer, pays for the usage charges

In addition to the characteristics, this cartridge includes rulesets that manage the state transitions of telephone numbers based on the telephone number type characteristic.

Table 5-12 lists and describes the characteristics included in the Base Phone Management cartridge.

Table 5-12 Characteristics

Specification	Description
Block Availability	Indicates whether a particular telephone number block is available for assignment
Block AvailabilityDate	Identifies the date after which a particular telephone number block will be available for assignment
Block Indicator	Identifies a telephone number block
Responsible Provider	Identifies the primary provider responsible for providing the telephone service

Table 5-12 (Cont.) Characteristics

Specification	Description
TN Country Code	Identifies the originating country of a mobile telephone number
TN Type	Indicates the type of telephone number: toll free, owned, ported in, or ported out
Winback	Identifies that a mobile telephone number has been provisioned as a Winback

Rulesets

Table 5-13 lists the rulesets and extension points included in the Base Phone Management cartridge.

Table 5-13 Rulesets in the Base Phone Management Cartridge

Rulesets	Functionality
SNAPBACK_TRANSITION	Manages the transition of a telephone number in Ported state to the Unassigned state. A telephone number is assigned the Ported Out state when the customer carries the number along to another service provider. However, it is still owned by the original service provider even though it is made unavailable for assignments within the system. When the customer relinquishes the number, this ruleset transitions the number from the Ported Out state to the Unassigned state and makes it available for reassignment within UIM.

Telephone Number Specifications and Rulesets

The Telephone Number Management cartridges model generic telephone number management services, including country-specific telephone number formatting and telephone number blocks. These services can be used in any telephony domain.

There are five separate cartridges for different countries, which you can use in any combination:

- ora_uim_us_tn
- ora_uim_canada_tn
- ora_uim_norway_tn
- ora_uim_uk_tn
- ora_uim_saudi_arabia_tn

Entity Specifications

The base telephone number cartridges provide telephone number specifications that model services used for country-specific telephone number formatting and managing telephone number blocks. These generic services may be used in any telephony domain, such as POTS or Mobile using any user-defined service configuration specification.

The telephone number specifications provide a template to indicate how UIM handles country-specific telephone number formats. The specifications include the Telephone Number Type characteristic that allows you to implement telephone number portability.

The Telephone Number Type characteristic differentiates telephone numbers a service provider owns from those that are ported in or ported out. UIM differentiates aging and state transition business logic for telephone numbers based on the value of this characteristic. You can deploy whichever base telephone number cartridges your business requires. For example, if you need to provision a telephone service for subscribers in the US and Canada, you need not deploy telephone number cartridges pertaining to Norway.

Telephone Number Specifications

Table 5-14 lists and describes the Telephone Number specifications included in the base telephone number cartridges.

Table 5-14 Telephone Number Specifications

Specification	Description
canadaTelephoneNumber	Defines telephone numbers for Canada; includes the following characteristics: • **TN Type**: Defines the type of telephone number; for example, owned by the service provider or ported; Valid values include Toll Free, Ported In, Ported Out, Owned; Can have a null value • **Responsible Provider**: The service provider of the telephone service • **TN Country Code**: Defines the international dialing code for Canada. The default value is **1**
canadaTelephoneNumberBlock	Defines blocks of Canadian telephone numbers; primarily used as a shortcut for including telephone numbers in inventory groups Includes the following characteristics: • **Block Indicator**: Groups telephone numbers into distinct blocks using a country-specific format; This grouping allows specialized rules to be written for telephone numbers that belong to different blocks. • **Responsible Provider**: The service provider that has received the particular telephone number from the regulatory agency of the country • **Block Availability**: Allows the provider to manage the telephone number isolation; You may customize rules to allow telephone number assignments from a block whose availability is set to Yes. • **Block Availability Date**: Date for which a particular telephone number block is available for assignment to subscribers
norwayTelephoneNumber	Defines telephone numbers for Norway; includes the following characteristics: • **TN Type**: Defines the type of telephone number; for example, owned by the service provider or ported telephone numbers; Valid values include Toll Free, Ported In, Ported Out, Owned; Can have a null value • **Responsible Provider**: The service provider that provides the telephone service • **TN Country Code**: Defines the international dialing code for Norway. The default value is **47**
norwayTelephoneNumberBlock	Defines blocks of Norwegian telephone numbers; primarily used as a shortcut for including telephone numbers in inventory groups Includes the following characteristics: • **Block Indicator**: Groups telephone numbers into distinct blocks using a country-specific format; This grouping allows specialized rules to be written for telephone numbers that belong to different blocks. • **Responsible Provider**: The service provider that has received the particular telephone number from the regulatory agency of the country • **Block Availability**: Allows the provider to manage the telephone number utilization; You may customize rules to allow telephone number assignments from a block whose availability is set to Yes • **Block Availability Date**: Date for which a particular telephone number block is available for assignment to subscribers

Table 5-14 (Cont.) Telephone Number Specifications

Specification	Description
saudiArabiaTelephoneNumber	Defines telephone numbers for Saudi Arabia; includes the following characteristics: • **TN Type**: Defines the type of telephone number; for example, owned by the service provider or ported telephone numbers; Valid values include Toll Free, Ported In, Ported Out, Owned; Can have a null value • **Responsible Provider**: The service provider that provides the telephone service • **TN Country Code**: Defines the international dialing code for Saudi Arabia. The default value is **966**
saudiArabiaTelephoneNumberBlock	Defines blocks of Saudi telephone numbers; primarily used as a shortcut for including telephone numbers in inventory groups Includes the following characteristics: • **Block Indicator**: Groups telephone numbers into distinct blocks using a country-specific format; This grouping allows specialized rules to be written for telephone numbers that belong to different blocks. • **Responsible Provider**: The service provider that has received the particular telephone number from the regulatory agency of the country • **Block Availability**: Allows the provider to manage the telephone number uilization; You may customize rules to allow telephone number assignments from a block whose availability is set to Yes • **Block Availability Date**: Date for which a particular telephone number block is available for assignment to subscribers
ukTelephoneNumber	Defines telephone numbers for the UK; includes the following characteristics: • **TN Type**: Defines the type of telephone number; for example, owned by the service provider or ported telephone numbers; Valid values include Toll Free, Ported In, Ported Out, Owned; Can have a null value • **Responsible Provider**: The service provider that provides the telephone service • **TN Country Code**: Defines the international dialing code for the UK. The default value is **44**
ukTelephoneNumberBlock	Defines blocks of UK telephone numbers; primarily used as a shortcut for including telephone numbers in inventory groups. Includes the following characteristics: • **Block Indicator**: Groups telephone numbers into distinct blocks using a country-specific format; This grouping allows specialized rules to be written for telephone numbers that belong to different blocks. • **Responsible Provider**: The service provider that has received the particular telephone number from the regulatory agency of the country. • **Block Availability**: Allows the provider to manage the telephone number utilization; You may customize rules to allow telephone number assignments from a block whose availability is set to **Yes**. • **Block Availability Date**: Date for which a particular telephone number block is available for assignment to subscribers.
usTelephoneNumber	Defines telephone numbers for the US; includes the following characteristics: • **TN Type**: Defines the type of telephone number; for example, owned by the service provider or ported telephone numbers; Valid values include Toll Free, Ported In, Ported Out, Owned; Can have a null value. • **Responsible Provider**: The service provider that provides the telephone service. • **TN Country Code**: Defines the international dialing code for the US. The default value is **1**.

Table 5-14 (Cont.) Telephone Number Specifications

Specification	Description
usTelephoneNumberBlock	Defines blocks of US telephone numbers; primarily used as a shortcut for including telephone numbers in inventory groups. Includes the following characteristics: • **Block Indicator**: Groups telephone numbers into distinct blocks using a country-specific format; This grouping allows specialized rules to be written for telephone numbers that belong to different blocks; Differentiates US telephone number instances with a format of seven digits as per the US format. • **Responsible Provider**: The service provider that has received the particular telephone number from the regulatory agency of the country. • **Block Availability**: Allows the provider to manage the telephone number utilization; You may customize rules to allow telephone number assignments from a block whose availability is set to **Yes**. • **Block Availability Date**: Date for which a particular telephone number block is available for assignment to subscribers.

Rulesets

Table 5-15 lists the rulesets and extension points included in the Telephone Number cartridges.

Table 5-15 Rulesets in the Telephone Number Cartridges

Rulesets	Functionality
TELEPHONE_NUMBER_BLOCK_FORMATTING_USTN	Defines the telephone number format and length as NNN-NNN-N for a US telephone number block.
TELEPHONE_NUMBER_FORMATTING_USTN	Contains rules for formatting telephone numbers based on service type and customer location The functionality of this ruleset may be customized for US telephone numbers
TELEPHONE_NUMBER_BLOCK_FORMATTING_CANADATN	Defines the telephone number format and length as MMM-MMM-N for a Canada telephone number block.
TELEPHONE_NUMBER_FORMATTING_CANADATN	Contains rules for formatting telephone numbers based on service type and customer location The functionality of this ruleset may be customized for Canada telephone numbers
TELEPHONE_NUMBER_BLOCK_FORMATTING_NORWAYTN	Contains rules for formatting blocks of telephone numbers based on service type and customer location The functionality of this ruleset may be customized for Norway telephone numbers
TELEPHONE_NUMBER_BLOCK_FORMATTING_UKTN	Contains rules for formatting blocks of telephone numbers based on service type and customer location The functionality of this ruleset may be customized for UK telephone numbers
TELEPHONE_NUMBER_FORMATTING_UKTN	Contains rules for formatting telephone numbers based on service type and customer location The functionality of this ruleset may be customized for UK telephone numbers

Table 5-15 (Cont.) Rulesets in the Telephone Number Cartridges

Rulesets	Functionality
TELEPHONE_NUMBER_BLOCK_FORMATTING_SATN	Contains rules for formatting blocks of telephone numbers based on service type and customer location The functionality of this ruleset may be customized for Saudi Arabia telephone numbers
TELEPHONE_NUMBER_FORMATTING_SATN	Contains rules for formatting telephone numbers based on service type and customer location The functionality of this ruleset may be customized for Saudi Arabia telephone numbers

Base Technologies Cartridge

This section includes information about the specifications and other artifacts included in the Base Technologies cartridge (ora_uim_basetechnologies). This cartridge supplies channelized connectivity features in UIM.

See "About Unified Inventory Management" in *UIM Concepts* and "Getting Started with Unified Inventory Management" in *UIM Help* for more information.

Connectivity Functions

Connectivity functions define the parts played by entities assigned to particular combinations of rate codes and technologies. Connectivity functions are divided into Facility and Service categories depending on their purpose.

Table 5-16 lists the connectivity functions included in the Base Technologies cartridge.

Table 5-16 Connectivity Functions

Specification	Description	Category
10xDS0	A Frame Relay facility supported by 10 DS0 channels.	Facility
10xDS1	A Frame Relay facility supported by 10xDS1 channels.	Facility
10xE0	A Frame Relay facility supported by 10xE0 channels.	Facility
10xE1	A Frame Relay facility supported by 10xE1 channels.	Facility
11xDS0	A Frame Relay facility supported by 11 DS0 channels.	Facility
11xE0	A Frame Relay facility supported by 11xE0 channels.	Facility
12xDS0	A Frame Relay facility supported by 12 DS0 channels.	Facility
12xDS1	A Frame Relay facility supported by 12xDS1 channels.	Facility
12xE0	A Frame Relay facility supported by 12xE0 channels.	Facility
12xE1	A Frame Relay facility supported by 12xE1 channels.	Facility
13xDS0	A Frame Relay facility supported by 13 DS0 channels.	Facility
13xE0	A Frame Relay facility supported by 13xE0 channels.	Facility
14xDS0	A Frame Relay facility supported by 14 DS0 channels.	Facility
14xDS1	A Frame Relay facility supported by 14xDS1 channels.	Facility
14xE0	A Frame Relay facility supported by 14xE0 channels.	Facility

Table 5-16 (Cont.) Connectivity Functions

Specification	Description	Category
15xDS0	A Frame Relay facility supported by 15 DS0 channels.	Facility
15xE0	A Frame Relay facility supported by 15xE0 channels.	Facility
16xDS0	A Frame Relay facility supported by 16 DS0 channels.	Facility
16xDS1	A Frame Relay facility supported by 16xDS1 channels.	Facility
16xE0	A Frame Relay facility supported by 16xE0 channels.	Facility
17xDS0	A Frame Relay facility supported by 17 DS0 channels.	Facility
17xE0	A Frame Relay facility supported by 17xE0 channels.	Facility
18xDS0	A Frame Relay facility supported by 18xDS0 channels.	Facility
18xDS1	A Frame Relay facility supported by 18xDS1 channels.	Facility
18xE0	A Frame Relay facility supported by 18xE0 channels.	Facility
20xDS1	A Frame Relay facility supported by 20xDS1 channels.	Facility
22xDS1	A Frame Relay facility supported by 22xDS1 channels.	Facility
2xDS0	A Frame Relay facility supported by 2 DS0 channels.	Facility
2xDS1	A Frame Relay facility supported by 2xDS1 channels.	Facility
2xE0	A Frame Relay facility supported by 2xE0 channels.	Facility
2xE1	A Frame Relay facility supported by 2xE1 channels.	Facility
3xDS0	A Frame Relay facility supported by 3 DS0 channels.	Facility
3xE0	A Frame Relay facility supported by 3xE0 channels.	Facility
4xDS0	A Frame Relay facility supported by 4 DS0 channels.	Facility
4xDS1	A Frame Relay facility supported by 4xDS1 channels.	Facility
4xE0	A Frame Relay facility supported by 4xE0 channels.	Facility
4xE1	A Frame Relay facility supported by 4xE1 channels.	Facility
5xDS0	A Frame Relay facility supported by 5 DS0 channels.	Facility
5xE0	A Frame Relay facility supported by 5xE0 channels.	Facility
6xDS0	A Frame Relay facility supported by 6 DS0 channels.	Facility
6xDS1	A Frame Relay facility supported by 6xDS1 channels.	Facility
6xE0	A Frame Relay facility supported by 6xE0 channels.	Facility
6xE1	A Frame Relay facility supported by 6xE1 channels.	Facility
7xDS0	A Frame Relay facility supported by 7 DS0 channels.	Facility
7xE0	A Frame Relay facility supported by 7xE0 channels.	Facility
8xDS0	A Frame Relay facility supported by 8 DS0 channels.	Facility
8xDS1	A Frame Relay facility supported by 8xDS1 channels.	Facility
8xE0	A Frame Relay facility supported by 8xE0 channels.	Facility
8xE1	A Frame Relay facility supported by 8xE1 channels.	Facility
9xDS0	A Frame Relay facility supported by 9 DS0 channels.	Facility
9xE0	A Frame Relay facility supported by 9xE0 channels.	Facility
ADSL	A DSL facility operating at an ADSL rate.	Facility

Table 5-16 (Cont.) Connectivity Functions

Specification	Description	Category
ADSL2	A DSL facility operating at an ADSL2 rate.	Facility
ADSL2+	A DSL facility operating at an ADSL2+ rate.	Facility
CDSL	A DSL facility operating at a CDSL rate.	Facility
DIA	A Diameter service.	Service
E0	An E-carrier transmission facility operating at an E0 rate.	Facility
E1	An E-carrier transmission facility operating at an E1 rate.	Facility
E2	An E-carrier transmission facility operating at an E2 rate.	Facility
E3	An E-carrier transmission facility operating at an E3 rate.	Facility
E4	An E-carrier transmission facility operating at an E4 rate.	Facility
E5	An E-carrier transmission facility operating at an E5 rate.	Facility
G.Lite	A DSL facility operating at a G.Lite rate.	Facility
GE1	An Ethernet or MPLS facility operating at a GE1 rate.	Facility
GE10	An Ethernet or MPLS facility operating at a GE10 rate.	Facility
GE100	An Ethernet or MPLS facility operating at a GE100 rate.	Facility
GE40	An Ethernet or MPLS facility operating at a GE40 rate.	Facility
HDSL	A DSL facility operating at an HDSL rate.	Facility
IDSL	A DSL facility operating at an IDSL rate.	Facility
IPTV	An IPTV service.	Service
J0	A J-carrier transmission facility operating at a J0 rate.	Facility
J1	A J-carrier transmission facility operating at a J1 rate.	Facility
J2	A J-carrier transmission facility operating at a J2 rate.	Facility
J2ALT	A J-carrier transmission facility operating at a J2alt rate.	Facility
J3	A J-carrier transmission facility operating at J3 rate.	Facility
J4	A J-carrier transmission facility operating at a J4 rate.	Facility
J5	A J-carrier transmission facility operating at a J5 rate.	Facility
LL	A Leased Line service.	Service
ME10	An Ethernet, ATM, or MPLS facility operating at an ME10 rate.	Facility
ME100	An Ethernet, ATM, or MPLS facility operating at an ME100 rate.	Facility
OC01	A synchronous digital-transmission facility that transmits OC-1 signals (51.840 Mbps) on optical media in a SONET network.	Facility
OC03	A synchronous digital-transmission facility that transmits OC-3 signals (155.520 Mbps) on optical media in a SONET network.	Facility
OC12	A synchronous digital-transmission facility that transmits OC-12 signals (622.080 Mbps) on optical media in a SONET network.	Facility
OC192	A synchronous digital-transmission facility that transmits OC-192 signals (9,953.230 Mbps) on optical media in a SONET network.	Facility
OC3072	A synchronous digital-transmission facility that transmits OC-3072 signals (159,252.480 Mbps) on optical media in a SONET network.	Facility
OC48	A synchronous digital-transmission facility that transmits OC-48 signals (2,488.320 Mbps) on optical media in a SONET network.	Facility

Table 5-16 (Cont.) Connectivity Functions

Specification	Description	Category
OC768	A synchronous digital-transmission facility that transmits OC-768 signals (39,813.120 Mbps) on optical media in a SONET network.	Facility
ODU0	A OTN Optical connectivity operating at a ODU0 Rate.	Facility
ODU1	A OTN Optical connectivity operating at a ODU1 Rate.	Facility
ODU2	A OTN Optical connectivity operating at a ODU2 Rate.	Facility
ODU2e	A OTN Optical connectivity operating at a ODU2e Rate.	Facility
ODU3	A OTN Optical connectivity operating at a ODU3 Rate.	Facility
ODU4	A OTN Optical connectivity operating at a ODU4 Rate.	Facility
OM1	A WDM facility operating at an OM1 rate.	Facility
OM10	A WDM facility operating at an OM10 rate.	Facility
OM11	A WDM facility operating at an OM11 rate.	Facility
OM12	A WDM facility operating at an OM12 rate.	Facility
OM16	A WDM facility operating at an OM16 rate.	Facility
OM160	A WDM facility operating at an OM160 rate.	Facility
OM18	A WDM facility operating at an OM18 rate.	Facility
OM3	A WDM facility operating at an OM3 rate.	Facility
OM32	A WDM facility operating at an OM32 rate.	Facility
OM4	A WDM facility operating at an OM4 rate.	Facility
OM40	A WDM facility operating at an OM40 rate.	Facility
OM70	A WDM facility operating at an OM70 rate.	Facility
OM8	A WDM facility operating at an OM8 rate.	Facility
OM80	A WDM facility operating at an OM80 rate.	Facility
OM96	A WDM facility operating at an OM96 rate.	Facility
OTU1	A OTN Optical connectivity operating at a OTU1 Rate.	Facility
OTU2	A OTN Optical connectivity operating at a OTU2 Rate.	Facility
OTU3	A OTN Optical connectivity operating at a OTU3 Rate.	Facility
OTU4	A OTN Optical connectivity operating at a OTU4 Rate.	Facility
RADSL	A DSL facility operating at a RDSL rate.	Facility
SDSL	A DSL facility operating at an SDSL rate.	Facility
SHDSL	A DSL facility operating at an SHDSL rate.	Facility
SM0	A digital optical TDM facility operating at a rate of STM-0.	Facility
SM01	A digital optical TDM facility operating at a rate of STM-1.	Facility
SM04	A digital optical TDM facility operating at a rate of STM-4.	Facility
SM1024	A digital optical TDM facility operating at a rate of STM-1024.	Facility
SM16	A digital optical TDM facility operating at a rate of STM-16.	Facility
SM256	A digital optical TDM facility operating at a rate of STM-256.	Facility
SM64	A digital optical TDM facility operating at a rate of STM-64.	Facility

Table 5-16 (Cont.) Connectivity Functions

Specification	Description	Category
ST01	A synchronous digital-transmission facility that transmits STS-1 signals (51.840 Mbps) on non-optical media in a SONET network.	Facility
ST03	A synchronous digital-transmission facility that transmits STS-3 signals (155.520 Mbps) on non-optical media in a SONET network.	Facility
T0	A T-carrier transmission facility operating at a DS0 rate.	Facility
T1	A T-carrier transmission facility operating at a DS1 rate.	Facility
T2	A T-carrier transmission facility operating at a DS2 rate.	Facility
T3	A T-carrier transmission facility operating at a DS3 rate.	Facility
T4	A T-carrier transmission facility operating at a DS4 rate.	Facility
T5	A T-carrier transmission facility operating at a DS5 rate.	Facility
VC11	A digital optical TDM facility operating at a rate of VC-11.	Facility
VC12	A digital optical TDM facility operating at a rate of VC-12.	Facility
VC2	A digital optical TDM facility operating at a rate of VC-2.	Facility
VC3	A digital optical TDM facility operating at a rate of VC-3.	Facility
VC4	A digital optical TDM facility operating at a rate of VC-4.	Facility
VDSL	A DSL facility operating at a VDSL rate.	Facility
VDSL2	A DSL facility operating at a VDSL2 rate.	Facility
VOIP	A VOIP service.	Service
VPN	A VPN service.	Service

Technology Types

Technology types define the technologies used by connectivities and other entities. Table 5-17 lists the technology type specifications.

Table 5-17 Technology Types

Specification	Description
ATM	A telecommunications technology that can carry a range of user traffic, including voice, data, and video signals. ATM is an abbreviation for Asynchronous Transfer Mode.
DSL	(Digital Subscriber Line) A family of technologies that are used to provide internet access by transmitting digital data over telephone lines.
E-Carrier	A TDM standard that is used in most parts of the world aside from the US, Canada, and Japan. Similar to and developed from T-Carrier.
Ethernet	A family of computer networking technologies for local area networks (LANs) and metropolitan area networks (MANs).
Frame Relay	A wide area network technology that specifies the physical and logical link layers of digital telecommunications channels using a packet switching methodology.
J-Carrier	A TDM standard that is used in Japan. Similar to and developed from T-Carrier.

Table 5-17 (Cont.) Technology Types

Specification	Description
MPLS	(Multi-Protocol Label Switching) DA technology in which data is directed from one network node to the next based on short path labels rather than long network addresses. MPLS can encapsulate packets of various network protocols. including T1/E1, ATM, Frame Relay, and DSL.
OTN	(Optical Transport Network) A series of standards created to combine the benefits of SONET/SDH with the bandwidth-expanding capabilities of WDM. This combination makes it possible to build more network functionality into optical networks. OTN is able to carry many types of data, including 100 GigE signals.
SONET	A set of multiplexing protocols that transfer digital bit streams over optical fiber using lasers or LEDs. Commonly used in the US and Canada. Similar to and derived from SDH.
SDH	A set of multiplexing protocols that transfer digital bit streams over optical fiber using lasers or LEDs. Commonly used outside the US and Canada. SDH is an abbreviation of Synchronous Digital Hierarchy.
T-Carrier	A time-division multiplexing (TDM) standard that is used in the US and Canada.
Undefined	Used when the technology is not specified.
WDM	(Wave Division Multiplexing) A fiber-optic technology in which a number of optical carrier signals are multiplexed onto a single optical fiber by using different wavelengths of laser light.

Rate Codes

Rate codes are a shorthand way to assign the signal type and capacity to a channelized connectivity entity or logical device. Rate codes are used for validation and other purposes during connectivity design. Table 5-18 lists the rate codes included in the Base Technologies cartridge.

Table 5-18 Rate Codes

Specification	Description
100GigE	Ethernet or MPLS signal with a nominal bit rate of 100000 Mbps.
100M	ATM, Ethernet, or MPLS signal with a nominal bit rate of 100 Mbps.
10GigE	Ethernet or MPLS signal with a nominal bit rate of 10000 Mbps.
10M	ATM, Ethernet, or MPLS signal with a nominal bit rate of 10 Mbps.
10xDS0	Frame Relay fractional signal with a nominal bit rate of 6.4 Mbps supported by 10 DS0 channels.
10xDS1	Frame Relay fractional signal with a nominal bit rate of 15.44 Mbps supported by 10xDS1 channels.
10xE0	Frame Relay fractional signal with a nominal bit rate of 6.4 Mbps supported by 10xE0 channels.
10xE1	Frame Relay fractional signal with a nominal bit rate of 20.48 Mbps supported by 10xE1 channels.
11xDS0	Frame Relay fractional signal with a nominal bit rate of 0.74 Mbps supported by 11 DS0 channels.
11xE0	Frame Relay fractional signal with a nominal bit rate of 0.74 Mbps supported by 11xE0 channels.

Table 5-18 (Cont.) Rate Codes

Specification	Description
12xDS0	Frame Relay fractional signal with a nominal bit rate of 0.768 Mbps supported by 12 DS0 channels.
12xDS1	Frame Relay fractional signal with a nominal bit rate of 18.528 Mbps supported by 12xDS1 channels.
12xE0	Frame Relay fractional signal with a nominal bit rate of 0.768 Mbps supported by 12xE0 channels.
12xE1	Frame Relay fractional signal with a nominal bit rate of 24.576 Mbps supported by 12xE1 channels.
13xDS0	Frame Relay fractional signal with a nominal bit rate of 0.832 Mbps supported by 13 DS0 channels.
13xE0	Frame Relay fractional signal with a nominal bit rate of 0.832 Mbps supported by 13xE0 channels.
14xDS0	Frame Relay fractional signal with a nominal bit rate of 0.896 Mbps supported by 14 DS0 channels.
14xDS1	Frame Relay fractional signal with a nominal bit rate of 21.616 Mbps supported by 14xDS1 channels.
14xE0	Frame Relay fractional signal with a nominal bit rate of 0.896 Mbps supported by 14xE0 channels.
15xDS0	Frame Relay fractional signal with a nominal bit rate of 0.960Mbps supported by 15 DS0 channels.
15xE0	Frame Relay fractional signal with a nominal bit rate of 0.960Mbps supported by 15xE0 channels.
16xDS0	Frame Relay fractional signal with a nominal bit rate of 1.024Mbps supported by 16 DS0 channels.
16xDS1	Frame Relay fractional signal with a nominal bit rate of 24.704 Mbps supported by 16xDS1 channels.
16xE0	Frame Relay fractional signal with a nominal bit rate of 1.024 Mbps supported by 16xE0 channels.
17xDS0	Frame Relay fractional signal with a nominal bit rate of 1.088 Mbps supported by 17 DS0 channels.
17xE0	Frame Relay fractional signal with a nominal bit rate of 1.088 Mbps supported by 17xE0 channels.
18xDS0	Frame Relay fractional signal with a nominal bit rate of 1.152 Mbps supported by 18 DS0 channels.
18xDS1	Frame Relay fractional signal with a nominal bit rate of Mbps 27.792 supported by 18xDS1 channels.
18xE0	Frame Relay fractional signal with a nominal bit rate of 1.152 Mbps supported by 18xE0 channels.
1GigE	Ethernet or MPLS signal with a nominal bit rate of 1000 Mbps.
20xDS1	Frame Relay fractional signal with a nominal bit rate of 30.880 Mbps supported by 20xDS1 channels.
22xDS1	Frame Relay fractional signal with a nominal bit rate of 33.968 Mbps supported by 22xDS1 channels.
2xDS0	Frame Relay fractional signal with a nominal bit rate of 0.128 Mbps supported by 2 DS0 channels.

Table 5-18 (Cont.) Rate Codes

Specification	Description
2xDS1	Frame Relay fractional signal with a nominal bit rate of 3.088 Mbps supported by 2xDS1 channels.
2xE0	Frame Relay fractional signal with a nominal bit rate of 0.128 Mbps supported by 2xE0 channels.
2xE1	Frame Relay fractional signal with a nominal bit rate of 4.096 Mbps supported by 2xE1 channels.
3xDS0	Frame Relay fractional signal with a nominal bit rate of 0.192 Mbps supported by 3 DS0 channels.
3xE0	Frame Relay fractional signal with a nominal bit rate of 0.192 Mbps supported by 3xE0 channels.
40GigE	Ethernet or MPLS signal with a nominal bit rate of 40000 Mbps.
4xDS0	Frame Relay fractional signal with a nominal bit rate of 0.256 Mbps supported by 4 DS0 channels.
4xDS1	Frame Relay fractional signal with a nominal bit rate of 6.176 Mbps supported by 4xDS1 channels.
4xE0	Frame Relay fractional signal with a nominal bit rate of 0.256 Mbps supported by 4xE0 channels.
4xE1	Frame Relay fractional signal with a nominal bit rate of 8.192 Mbps supported by 4xE1 channels.
5xDS0	Frame Relay fractional signal with a nominal bit rate of 0.320 Mbps supported by 5 DS0 channels.
5xE0	Frame Relay fractional signal with a nominal bit rate of 0.320 Mbps supported by 5xE0 channels.
6xDS0	Frame Relay fractional signal with a nominal bit rate of 0.384 Mbps supported by 6 DS0 channels.
6xDS1	Frame Relay fractional signal with a nominal bit rate of 9.264 Mbps supported by 6xDS1 channels.
6xE0	Frame Relay fractional signal with a nominal bit rate of 0.384 Mbps supported by 6 YY channels.
6xE1	Frame Relay fractional signal with a nominal bit rate of 12.288 Mbps supported by 6xE1 channels.
7xDS0	Frame Relay fractional signal with a nominal bit rate of 0.448 Mbps supported by 7 DS0 channels.
7xE0	Frame Relay fractional signal with a nominal bit rate of 0.448 Mbps supported by 7xE0 channels.
8xDS0	Frame Relay fractional signal with a nominal bit rate of 0.512 Mbps supported by 8 DS0 channels.
8xDS1	Frame Relay fractional signal with a nominal bit rate of 12.352 Mbps supported by 8xDS1 channels.
8xE0	Frame Relay fractional signal with a nominal bit rate of 0.512 Mbps supported by 8xE0 channels.
8xE1	Frame Relay fractional signal with a nominal bit rate of 16.384 Mbps supported by 8xE1 channels.
9xDS0	Frame Relay fractional signal with a nominal bit rate of 0.576 Mbps supported by 9 DS0 channels.

Table 5-18 (Cont.) Rate Codes

Specification	Description
9xE0	Frame Relay fractional signal with a nominal bit rate of 0.576Mbps supported by 9xE0 channels.
ADSL	DSL signal with a nominal bit rate of 7.000 Mbps.
ADSL2	DSL signal with a nominal bit rate of 24.000 Mbps.
ADSL2+	DSL signal with a nominal bit rate of 8.000 Mbps.
CDSL	DSL signal with a nominal bit rate of 1.000 Mbps.
DS0	T-Carrier or Frame Relay signal with a capacity of 0.064 Mbps.
DS1	T-Carrier, Frame Relay, or ATM signal with a capacity of 1.544 Mbps.
DS2	T-Carrier signal with a capacity of 6.312 Mbps.
DS3	T-Carrier, Frame Relay, or ATM signal with a capacity of 44.736 Mbps.
DS4	T-Carrier signal with a capacity of 274.176 Mbps.
DS5	T-Carrier signal with a capacity of 274.176 Mbps 400.352.
E0	E-Carrier or Frame Relay signal with a capacity of 0.064 Mbps.
E1	E-Carrier, Frame Relay, or ATM signal with a capacity of 2.048 Mbps.
E2	E-Carrier signal with a capacity of 8.448 Mbps.
E3	E-Carrier, Frame Relay, or ATM signal with a capacity of 34.368 Mbps.
E4	E-Carrier signal with a capacity of 139.264 Mbps.
E5	E-Carrier signal with a capacity of 565.148 Mbps.
G.Lite	DSL signal with a nominal bit rate of 16.000 Mbps.
HDSL	DSL signal with a nominal bit rate of 2.048 Mbps.
IDSL	DSL signal with a nominal bit rate of 0.148 Mbps.
J0	J-Carrier signal with a capacity of 0.064 Mbps.
J1	J-Carrier signal with a capacity of 1.544 Mbps.
J2	J-Carrier signal with a capacity of 6.312 Mbps.
J2alt	J-Carrier signal with a capacity of 7.786 Mbps.
J3	J-Carrier signal with a capacity of 32.064 Mbps.
J4	J-Carrier signal with a capacity of 97.728 Mbps.
J5	J-Carrier signal with a capacity of 565.148 Mbps.
OC1	SONET signal with a capacity of 51.840 Mbps.
OC12	SONET, Frame Relay, or ATM signal with a capacity of 622.080 Mbps.
OC192	SONET, Frame Relay, or ATM signal with a capacity of 9953.280 Mbps.
OC3	SONET, Frame Relay, or ATM signal with a capacity of 155.520 Mbps.
OC3072	SONET signal with a capacity of 159252.480 Mbps.
OC48	SONET, Frame Relay, or ATM signal with a capacity of 2488.320 Mbps.
OC768	SONET or MPLS signal with a capacity of 39813.120 Mbps.
ODU0	OTN signal with a nominal bit rate of 1244.160 Mbps.
ODU1	OTN signal with a nominal bit rate of 2498.775 Mbps.
ODU2	OTN signal with a nominal bit rate of 10037.273 Mbps.

Table 5-18 (Cont.) Rate Codes

Specification	Description
ODU2e	OTN signal with a nominal bit rate of 10312.500 Mbps.
ODU3	OTN signal with a nominal bit rate of 40319.219 Mbps.
ODU4	OTN signal with a nominal bit rate of 104794.445 Mbps.
OM1	WDM signal with a nominal bit rate of 10000 Mbps.
OM10	WDM signal with a nominal bit rate of 100000 Mbps.
OM11	WDM signal with a nominal bit rate of 110000 Mbps.
OM12	WDM signal with a nominal bit rate of 120000 Mbps.
OM16	WDM signal with a nominal bit rate of 160000 Mbps.
OM160	WDM signal with a nominal bit rate of 1600000 Mbps.
OM18	WDM signal with a nominal bit rate of 180000 Mbps.
OM3	WDM signal with a nominal bit rate of 30000 Mbps.
OM32	WDM signal with a nominal bit rate of 320000 Mbps.
OM4	WDM signal with a nominal bit rate of 40000 Mbps.
OM40	WDM signal with a nominal bit rate of 400000 Mbps.
OM70	WDM signal with a nominal bit rate of 700000 Mbps.
OM8	WDM signal with a nominal bit rate of 80000 Mbps.
OM80	WDM signal with a nominal bit rate of 800000 Mbps.
OM96	WDM signal with a nominal bit rate of 960000 Mbps.
OTU1	OTN signal with a nominal bit rate of 2666.057 Mbps.
OTU2	OTN signal with a nominal bit rate of 10709.224 Mbps.
OTU3	OTN signal with a nominal bit rate of 43018.41 Mbps.
OTU4	OTN signal with a nominal bit rate of 111809.973 Mbps.
RADSL	DSL signal with a nominal bit rate of 2.200 Mbps.
SDSL	DSL signal with a nominal bit rate of 2.048 Mbps.
SHDSL	DSL signal with a nominal bit rate of 4.624 Mbps.
STM0	SDH signal with a capacity of 51.840 Mbps.
STM1	SDH or ATM signal with a capacity of 155.520 Mbps.
STM1024	SDH signal with a capacity of 159252.480 Mbps.
STM16	SDH, MPLS, or ATM signal with a capacity of 2488.320 Mbps.
STM256	SDH or MPLS signal with a capacity of 39813.120 Mbps.
STM4	SDH or ATM signal with a capacity of 622.080 Mbps.
STM64	SDH, MPLS, or ATM signal with a capacity of 9953.280 Mbps.
STS1	SONET signal with a capacity of 51.840 Mbps.
STS12	SONET signal with a capacity of 622.080 Mbps.
STS192	SONET signal with a capacity of 9953.280 Mbps.
STS3	SONET signal with a capacity of 155.520 Mbps.
STS3072	SONET signal with a capacity of 159252.480 Mbps.

Table 5-18 (Cont.) Rate Codes

Specification	Description
STS48	SONET signal with a capacity of 2488.320 Mbps.
STS768	SONET signal with a capacity of 39813.120 Mbps.
VC11	SDH signal with a capacity of 1.728 Mbps.
VC12	SDH signal with a capacity of 2.304 Mbps.
VC2	SDH signal with a capacity of 6.912 Mbps.
VC3	SDH signal with a capacity of 48.960 Mbps.
VC4	SDH signal with a capacity of 150.336 Mbps.
VC4-1024	SDH signal with a capacity of 153944.064 Mbps.
VC4-16	SDH signal with a capacity of 2405.376 Mbps.
VC4-256	SDH signal with a capacity of 38486.016 Mbps.
VC4-4	SDH signal with a capacity of 601.344 Mbps.
VC4-64	SDH signal with a capacity of 9621.504 Mbps.
VDSL	DSL signal with a nominal bit rate of 55.000 Mbps.
VDSL2	DSL signal with a nominal bit rate of 100.000 Mbps.
VT1.5	SONET signal with a capacity of 1.728 Mbps.
VT2	SONET signal with a capacity of 2.304 Mbps.
VT6	SONET signal with a capacity of 6.912 Mbps.

Connectivity Signal Termination Point Specifications

Connectivity Signal Termination Point specifications define the signals for facilities, channels, and processing signals. These specifications define the layers and capacity of a signal structure. A signal structure definition is composed of two or more Connectivity Signal Termination Point specifications in a hierarchy.

The Connectivity Signal Termination Point specifications in the Base Technologies cartridge help define the UIM signal architecture, which is used to define the signal structures of channelized connectivity. See "About Unified Inventory Management" in *UIM Concepts* for more information about channelized connectivity. Table 5-19 lists the Connectivity Signal Termination Point specifications included in the Base Technologies cartridge.

Table 5-19 Connectivity Signal Termination Point Specifications

Specification	Description
100GigE	Defines a packet optical signal with 100GigE rate code.
100M	Defines a packet electrical signal with a 100M rate code.
10GigE	Defines a packet optical signal with 10GigE rate code.
10M	Defines a packet electrical signal with a 10M rate code.
10xDS0	Defines a PDH electrical signal with a 10xDS0 rate code.
10xDS1	Defines a PDH electrical signal with a 10xDS1 rate code.
10xE0	Defines a PDH electrical signal with a 10xE0 rate code.

Table 5-19 (Cont.) Connectivity Signal Termination Point Specifications

Specification	Description
10xE1	Defines a PDH electrical signal with a 10xE1 rate code.
11xDS0	Defines a PDH electrical signal with a 11xDS0 rate code.
11xE0	Defines a PDH electrical signal with a 11xE0 rate code.
12xDS0	Defines a PDH electrical signal with a 12xDS0 rate code.
12xDS1	Defines a PDH electrical signal with a 12xDS1 rate code.
12xE0	Defines a PDH electrical signal with a 12xE0 rate code.
12xE1	Defines a PDH electrical signal with a 12xE1 rate code.
13xDS0	Defines a PDH electrical signal with a 13xDS0 rate code.
13xE0	Defines a PDH electrical signal with a 13xE0 rate code.
14xDS0	Defines a PDH electrical signal with a 14xDS0 rate code.
14xDS1	Defines a PDH electrical signal with a 14xDS1 rate code.
14xE0	Defines a PDH electrical signal with a 14xE0 rate code.
15xDS0	Defines a PDH electrical signal with a 0.960Mbps supported by 15xDS0 rate code.
15xE0	Defines a PDH electrical signal with a 0.960Mbps supported by 15xE0 rate code.
16xDS0	Defines a PDH electrical signal with a 1.024Mbps supported by 16xDS0 rate code.
16xDS1	Defines a PDH electrical signal with a 16xDS1 rate code.
16xE0	Defines a PDH electrical signal with a 16xE0 rate code.
17xDS0	Defines a PDH electrical signal with a 17xDS0 rate code.
17xE0	Defines a PDH electrical signal with a 17xE0 rate code.
18xDS0	Defines a PDH electrical signal with a 18xDS0 rate code.
18xDS1	Defines a PDH electrical signal with a Mbps 27.792 supported by 18xDS1 rate code.
18xE0	Defines a PDH electrical signal with a 18xE0 rate code.
1GigE	Defines a packet optical signal with 1GigE rate code.
20xDS1	Defines a PDH electrical signal with a 20xDS1 rate code.
22xDS1	Defines a PDH electrical signal with a 22xDS1 rate code.
2xDS0	Defines a PDH electrical signal with a 2xDS0 rate code.
2xDS1	Defines a PDH electrical signal with a 2xDS1 rate code.
2xE0	Defines a PDH electrical signal with a 2xE0 rate code.
2xE1	Defines a PDH electrical signal with a 2xE1 rate code.
3xDS0	Defines a PDH electrical signal with a 3xDS0 rate code.
3xE0	Defines a PDH electrical signal with a 3xE0 rate code.
40GigE	Defines a packet optical signal with 40GigE rate code.
4xDS0	Defines a PDH electrical signal with a 4xDS0 rate code.
4xDS1	Defines a PDH electrical signal with a 4xDS1 rate code.
4xE0	Defines a PDH electrical signal with a 4xE0 rate code.
4xE1	Defines a PDH electrical signal with a 4xE1 rate code.
5xDS0	Defines a PDH electrical signal with a 5xDS0 rate code.

Table 5-19 (Cont.) Connectivity Signal Termination Point Specifications

Specification	Description
5xE0	Defines a PDH electrical signal with a 5xE0 rate code.
6xDS0	Defines a PDH electrical signal with a 6xDS0 rate code.
6xDS1	Defines a PDH electrical signal with a 6xDS1 rate code.
6xE0	Defines a PDH electrical signal with a 6 YY channels.
6xE1	Defines a PDH electrical signal with a 6xE1 rate code.
7xDS0	Defines a PDH electrical signal with a 7xDS0 rate code.
7xE0	Defines a PDH electrical signal with a 7xE0 rate code.
8xDS0	Defines a PDH electrical signal with a 8xDS0 rate code.
8xDS1	Defines a PDH electrical signal with a 8xDS1 rate code.
8xE0	Defines a PDH electrical signal with a 8xE0 rate code.
8xE1	Defines a PDH electrical signal with a 8xE1 rate code.
9xDS0	Defines a PDH electrical signal with a 9xDS0 rate code.
9xE0	Defines a PDH electrical signal with a 9xE0 rate code.
ADSL	Defines a packet electrical signal with a ADSL rate code.
ADSL2	Defines a packet electrical signal with a ADSL2 rate code.
ADSL2+	Defines a packet electrical signal with a ADSL2+ rate code.
AU-3	Defines an administrative unit processing signal with a VC3 rate code. Can be multiplexed from AUG-1 as up to 3 channels or from STM-0 as 1 channel.
AU-4	Defines an administrative unit processing signal with a VC4 rate code. Can be multiplexed from AUG-1 as 1 channel.
AU-4-1024c	Defines an administrative unit processing signal with a VC4-1024 rate code. Can be multiplexed from AUG-1024 as 1 channel.
AU-4-16c	Defines an administrative unit processing signal with a VC4-16 rate code. Can be multiplexed from AUG-16 as 1 channel.
AU-4-256c	Defines an administrative unit processing signal with a VC4-256 rate code. Can be multiplexed from AUG-256 as 1 channel.
AU-4-4c	Defines an administrative unit processing signal with a VC4-4 rate code. Can be multiplexed from AUG-4 as 1 channel.
AU-4-64c	Defines an administrative unit processing signal with a VC4-64 rate code. Can be multiplexed from AUG-64 as 1 channel.
AUG-1	Defines an administrative unit group processing signal with a VC4 rate code. Can be multiplexed from AUG-4 as up to 4 channels or from STM-1 as 1 channel.
AUG-1024	Defines an administrative unit group processing signal with a VC4-1024 rate code. Can be multiplexed from STM-1024 as 1 channel.
AUG-16	Defines an administrative unit group processing signal with a VC4-16 rate code. Can be multiplexed from AUG-64 as up to 4 channels or from STM-16 as 1 channel.
AUG-256	Defines an administrative unit group processing signal with a VC4-256 rate code. Can be multiplexed from AUG-1024 as up to 4 channels or from STM-256 as 1 channel.
AUG-4	Defines an administrative unit group processing signal with a VC4-4 rate code. Can be multiplexed from AUG-16 as up to 4 channels. Can be multiplexed from STM-4 as 1 channel.
AUG-64	Defines an administrative unit group processing signal with a VC4-64 rate code. Can be multiplexed from AUG-256 as up to 4 channels or from STM-64 as 1 channel.

Table 5-19 (Cont.) Connectivity Signal Termination Point Specifications

Specification	Description
C-11	Defines a container signal with a VC11 rate code. Maps to a VC-11 signal.
C-12	Defines a container signal with a VC12 rate code. Maps to a VC-12 signal
C-2	Defines a container signal with a VC2 rate code. Maps to a VC2 signal.
C-3	Defines a container signal with a VC3 rate code. Maps to a VC-3 signal.
C-4	Defines a container signal with a VC4 rate code. Maps to a VC-4 signal.
C-4-1024c	Defines a container signal with a VC4-1024 rate code. Maps to a VC-4-1024c signal.
C-4-16c	Defines a container signal with a VC4-16 rate code. Maps to a VC-4-16c signal.
C-4-256c	Defines a container signal with a VC4-256 rate code. Maps to a VC-4-256c signal.
C-4-4c	Defines a container signal with a VC4-4 rate code. Maps to a VC-4-4c signal.
C-4-64c	Defines a container signal with a VC4-64 rate code. Maps to a VC-4-64c signal.
CDSL	Defines a packet electrical signal with a CDSL rate code.
DS0	Defines a PDH electrical signal with a DS0 rate code. Cannot be used to define facility signals. Can be multiplexed from DS1 as up to 24 channels.
DS1	Defines a PDH electrical signal with a DS1 rate code. Can be multiplexed from DS2 as up to 4 channels. Can be multiplexed from DS3 as up to 28 channels. Supported by VT-1.5SPE and C-11 signals.
DS2	Defines a PDH electrical signal with a DS2 rate code. Can be multiplexed from DS3 as up to 7 channels. Supported by VT-6SPE and C-2 signals.
DS3	Defines a PDH electrical signal with a DS3 rate code. Can be multiplexed from DS4 as up to 6 channels. Supported by STS-1SPE and C-3 signals.
DS4	Defines a PDH electrical signal with a DS4 rate code.
E0	Defines a PDH electrical signal with a E0 rate code. Can be multiplexed from E1 as up to 30 channels.
E1	Defines a PDH electrical signal with a E1 rate code. Can be multiplexed from E2 as up to 4 channels. Can be multiplexed from E3 as up to 16 channels. Supported by VT-2SPE and C-12 signals.
E2	Defines a PDH electrical signal with a E2 rate code. Can be multiplexed from E3 as up to 4 channels.
E3	Defines a PDH electrical signal with a E3 rate code. Can be multiplexed from E4 as up to 4 channels. Supported by STS-1SPE and C-3 signals.
E4	Defines a PDH electrical signal with a E4 rate code. Can be multiplexed from E5 as up to 4 channels.Supported by STS-3cSPE and C-4 signals.
E5	Defines a PDH electrical signal with a E5 rate code.
G.Lite	Defines a packet electrical signal with a G.Lite rate code.
HDSL	Defines a packet electrical signal with a HDSL rate code.
IDSL	Defines a packet electrical signal with a IDSL rate code.
J0	Defines a PDH electrical signal with a J0 rate code. Can be multiplexed from J1 as up to 24 channels.
J1	Defines a PDH electrical signal with a J1 rate code. Can be multiplexed from J2 as up to 4 channels. Supported by J2alt as up to 5 channels. Supported by VT-1.5SPE and C-11 signals.
J2	Defines a PDH electrical signal with a J2 rate code. Can be multiplexed from J3 as up to 5 channels. Supported by VT-6SPE and C-2 signals.
J2alt	Defines a PDH electrical signal with a J2alt rate code. Can be multiplexed from J3 as up to 4 channels.
J3	Defines a PDH electrical signal with a J3 rate code. Can be multiplexed from J4 as up to 3 channels. Supported by STS-1SPE and C-3 signals.
J4	Defines a PDH electrical signal with a J4 rate code. Can be multiplexed from J5 as up to 4 channels.

Table 5-19 (Cont.) Connectivity Signal Termination Point Specifications

Specification	Description
J5	Defines a PDH electrical signal with a J5 rate code.
OC-1	Defines an optical signal with a n OC1 rate code.
OC-12	Defines an optical signal with an OC12 rate code.
OC-192	Defines an optical signal with an OC192 rate code.
OC-3	Defines an optical signal with an OC3 rate code.
OC-3072	Defines an optical signal with an OC3072 rate code.
OC-48	Defines an optical signal with an OC48 rate code.
OC-768	Defines an optical signal with a OC768 rate code.
ODU0	Defines an optical signal with an ODU0 rate code.
ODU1	Defines an optical signal with an ODU1 rate code.
ODU2	Defines an optical signal with an ODU2 rate code.
ODU2e	Defines an optical signal with an ODU2e rate code.
ODU3	Defines an optical signal with an ODU3 rate code.
ODU4	Defines an optical signal with an ODU4 rate code.
OM1	Defines an optical signal with an OM1 rate.
OM10	Defines an optical signal with an OM10 rate. Can be multiplexed to 10 OM1 channels.
OM11	Defines an optical signal with an OM11 rate. Can be multiplexed to 11 OM1 channels.
OM12	Defines an optical signal with an OM12 rate. Can be multiplexed to: • 3 OM4 channels • 4 OM3 channels • 12 OM1 channels
OM16	Defines an optical signal with an OM16 rate. Can be multiplexed to: • 2 OM8 channels • 4 OM4 channels • 16 OM1 channels
OM160	Defines an optical signal with an OM160 rate. Can be multiplexed to: • 2 OM80 channels • 4 OM40 channels • 5 OM32 channels • 10 OM16 channels • 16 OM10 channels • 20 OM8 channels • 40 OM4 channels • 160 OM1 channels
OM18	Defines an optical signal with an OM18 rate. Can be multiplexed to: • 6 OM3 channels • 18 OM1 channels
OM3	Defines an optical signal with an OM3 rate. Can be multiplexed to 3 OM1 channels.

Table 5-19 (Cont.) Connectivity Signal Termination Point Specifications

Specification	Description
OM32	Defines an optical signal with an OM32 rate. Can be multiplexed to: • 2 OM16 channels • 4 OM8 channels • 8 OM4 channels • 32 OM1 channels
OM4	Defines an optical signal with an OM4 rate. Can be multiplexed to 4 OM1 channels.
OM40	Defines an optical signal with an OM40 rate. Can be multiplexed to 4 OM10 channels or 10 OM4 channels.
OM70	Defines an optical signal with an OM70 rate. Can be multiplexed to: • 7 OM10channels • 70 OM1 channels
OM8	Defines an optical signal with an OM8 rate. Can be multiplexed to 2 OM4 channels or 8 OM1 channels.
OM80	Defines an optical signal with an OM80 rate. Can be multiplexed to: • 5 OM16 channels • 10 OM8 channels • 20 OM4 channels • 80 OM1 channels
OM96	Defines an optical signal with an OM96 rate. Can be multiplexed to: • 3 OM32 channels • 6 OM16 channels • 8 OM12 channels • 12 OM8 channels • 24 OM4 channels • 32 OM3 channels • 96 OM1 channels
OTU1	Defines an optical signal with an OTU1 rate code.
OTU2	Defines an optical signal with an OTU2 rate code.
OTU3	Defines an optical signal with an OTU3 rate code.
OTU4	Defines an optical signal with an OTU4 rate code.
RADSL	Defines a packet electrical signal with a RADSL rate code.
SDSL	Defines a packet electrical signal with a SDSL rate code.
SHDSL	Defines a packet electrical signal with a SHDSL rate code.
STM-0	Defines an STM optical signal with an STM0 rate code.
STM-1	Defines an STM optical signal with an STM1 rate code.
STM-1024	Defines an STM optical signal with an STM1024 rate code.
STM-16	Defines an STM optical signal with an STM16 rate code.
STM-256	Defines an STM optical signal with an STM256 rate code.
STM-4	Defines an STM optical signal with an STM4 rate code.
STM-64	Defines an STM optical signal with an STM64 rate code.
STS -192	Defines an STS electrical signal with an STS192 rate code. Can be multiplexed from OC-3072 as up to 16 channels, from OC-768 as up to 4 channels, from OC-192 as 1 channel, from STS-3072 as up to 16 channels, or from STS-768 as up to 4 channels.
STS-1	Defines an STS electrical signal with an STS1 rate code.

Table 5-19 (Cont.) Connectivity Signal Termination Point Specifications

Specification	Description
STS-1	Defines an STS processing signal with an STS1 rate code. Can be multiplexed in the following ways: • From OC-3072 as up to 3072 channels • From OC-768 as up to 768 channels • From OC-192 as up to 192 channels • From OC-48 as up to 48 channels • From OC-12 as up to 12 channels • From OC-3 as up to 3 channels • From OC-1 as 1 channel • From STS-3072 as up to 3072 channels • From STS-768 as up to 768 channels • From STS-192 as up to 192 channels • From STS-48 as up to 48 channels • From STS-12 as up to 12 channels • From STS-3 as up to 3 channels • From STS-3_STP as up to 3 channels
STS-1 SPE	Defines an SPE electrical signal with an STS1 rate code. Aligns with STS-1.Aligns with STS-1_STP.
STS-12	Defines an STS processing signal with an STS12 rate code. Can be multiplexed in the following ways: • From OC-3072 as up to 256 channels • From OC-768 as up to 64 channels • From OC-192 as up to 16 channels • From OC-48 as up to 4 channels • From OC-12 as 1 channel • From STS-3072 as up to 256 channels • From STS-768 as up to 64 channels • From STS-192 as up to 16 channels • From STS-48 as up to 4 channels
STS-12c SPE	Defines an SPE electrical signal with an STS12 rate code. Aligns with STS-12.
STS-192c SPE	Defines an SPE electrical signal with an STS192 rate code. Aligns with STS-192.
STS-3	Defines an STS processing signal with an STS3 rate code. Can be multiplexed in the following ways: • From OC-3072 as up to 1024 channels • From OC-768 as up to 256 channels • From OC-192 as up to 64 channels • From OC-48 as up to 16 channels • From OC-12 as up to 4 channels • From OC-3 as 1 channel • From STS-3072 as up to 1024 channels • From STS-768 as up to 256 channels • From STS-192 as up to 64 channels • From STS-48 as up to 16 channels • From STS-12 as up to 4 channels
STS-3	Defines an STS electrical signal with an STS3 rate code.
STS-3072	Defines an STS electrical signal with an STS3072 rate code. Can be multiplexed from OC-3072 as 1 channel.
STS-3072c SPE	Defines an SPE electrical signal with an STS3072 rate code. Aligns with STS-3072.
STS-3c SPE	Defines an SPE electrical signal with an STS3 rate code. Aligns with STS-3.Aligns with STS-3_STP.

Table 5-19 (Cont.) Connectivity Signal Termination Point Specifications

Specification	Description
STS-48	Defines an STS processing signal with an STS48 rate code. Can be multiplexed in the following ways: • From OC-3072 as up to 64 channels • From OC-768 as up to 16 channels • From OC-192 as up to 4 channels • From OC-48 as 1 channel • From STS-768 as up to 16 channels • From STS-192 as up to 4 channels
STS-48c SPE	Defines an SPE electrical signal with an STS48 rate code. Aligns with STS-48.
STS-768	Defines an STS electrical signal with an STS768 rate code. Can be multiplexed from OC-3072 as up to 4 channels, from OC-768 as 1 channel, or from STS-3072 as up to 4 channels.
STS-768c SPE	Defines an SPE electrical signal with an STS768 rate code. Aligns with STS-768.
TU-11	Defines a tributary unit processing signal with a VC11 rate code. Can be multiplexed from TUG-2 as up to 4 channels.
TU-12	Defines a tributary unit processing signal with a VC12 rate code. Can be multiplexed from TUG-2 as up to 3 channels.
TU-2	Defines a tributary unit processing signal with a VC2 rate code. Can be multiplexed from TUG-2 as 1 channel.
TU-3	Defines a tributary unit processing signal with a VC3 rate code. Can be multiplexed from TUG-3 as 1 channel.
TUG-2	Defines a tributary unit group processing signal with a VC2 rate code. Can be multiplexed from TUG-3 as up to 7 channels or from VC-3 as up to 7 channels.
TUG-3	Defines a tributary unit group processing signal with a VC3 rate code. Can be multiplexed from VC-4 as up to 3 channels.
VC-11	Defines a virtual container processing signal with a VC11 rate code. Aligns with TU-11.Aligns with TU-12.
VC-12	Defines a virtual container processing signal with a VC12 rate code. Aligns with TU-12.
VC-2	Defines a virtual container processing signal with a VC2 rate code. Aligns with TU-2.
VC-3	Defines a virtual container processing signal with a VC3 rate code. Aligns with TU-3.Aligns with AU-3.
VC-4	Defines a virtual container processing signal with a VC4 rate code. Aligns with AU-4.
VC-4-1024c	Defines a virtual container processing signal with a VC4-1024 rate code. Aligns with AU-4-1024c.
VC-4-16c	Defines a virtual container processing signal with a VC4-16 rate code. Aligns with AU-4-16c.
VC-4-256c	Defines a virtual container processing signal with a VC4-256 rate code. Aligns with AU-4-256c.
VC-4-4c	Defines a virtual container processing signal with a VC4-4 rate code. Aligns with AU-4-4c.
VC-4-64c	Defines a virtual container processing signal with a VC4-64 rate code. Aligns with AU-4-64c.
VDSL	Defines a packet electrical signal with a VDSL rate code.
VDSL2	Defines a packet electrical signal with a VDSL2 rate code.
VT-1.5	Defines a virtual tributary processing signal with a VT1.5 rate code. Can be multiplexed from VT-Group as up to 4 channels.
VT-1.5 SPE	Defines an SPE electrical signal with a VT1.5 rate code. Maps to a VT-1.5 signal.
VT-2	Defines a virtual tributary processing signal with a VT2 rate code. Can be multiplexed from VT-Group as up to 3 channels.
VT-2 SPE	Defines a SPE electrical signal with a VT2 rate code. Maps to VT-2 signal.

Table 5-19 (Cont.) Connectivity Signal Termination Point Specifications

Specification	Description
VT-6	Defines a virtual tributary processing signal with a VT6 rate code. Can be multiplexed from VT-Group as 1 channel.
VT-6 SPE	Defines a SPE electrical signal with a VT6 rate code. Maps to a VT-6 signal.
VT-Group	Defines a virtual tributary group processing signal with a VT6 rate code. Can be multiplexed from STS-1SPE as up to 7 channels.

Base Tags Cartridge

The Base Tags cartridge includes specifications for tags used in UIM.

Table 5-20 Tags in the Base Tags Cartridge

Specification	Description
AutoCreateAndAssign	Defines a tag that identifies that an entity should be created and assigned to a configuration item to be automatically created when a configuration is created.

6

Base Rulesets and Extension Points

This chapter contains information about the rulesets in the Oracle Communications Unified Inventory Management (UIM) Base Rulesets cartridge and the base extension points in the Base Extensionpoints cartridge.

Base Rulesets

The Base Rulesets cartridge contains several rulesets that provide fundamental capabilities required to provision an inventory solution for specific domains, such as VoIP or POTS.

For example, the Base Rulesets cartridge provides a ruleset that identifies geographic place formats for service points. There are rulesets to validate resource availability and service configurations based on service parameters.

Use the base rulesets for reference or modify or substitute them to create custom rulesets. For example, the default telephone number format may not match the telephone number format used by the country in which you are implementing UIM. To reformat the telephone number, reuse or modify the base ruleset used to format telephone numbers.

Alternatively, you can write your own ruleset, assign it the same name as the base ruleset, and replace the base ruleset with it. See "Overview" in *UIM Developer's Guide* for more information on how to create your own rulesets.

Base Ruleset Descriptions

Table 6-1 lists and describes the basic rulesets in the Base Rulesets cartridge.

Table 6-1 Rulesets in the Base Rulesets Cartridge

Ruleset/Ruleset Extension Points	Functionality
ADDRESS_RANGE_VALIDATION	Validates address ranges for geographic addresses in China.
CONVERT_LD_SR1_TO_SR2	Converts logical device entities from the format used in the SR1 release of UIM to the format used in SR2.
CREATE_ADDRESS_CHARACTERISTIC_MAP	Returns values for address associated with characteristic names.
EncryptText	Encrypts passwords in data federation scenarios.
FIND_ADDRESS_RANGE	Distinguishes between address formats for a city or town based on incoming geographic address and retrieves associated characteristic values.
IMPORT_INVENTORY	Imports inventory data from external systems using a CSV file. You should customize the ruleset based on your business requirements before you deploy it. For example, you can customize this ruleset to import inventory data in bulk using a CSV file.
PLACE_FORMAT_IDENTIFIER	Identifies country-specific telephone number formats.

Table 6-1 (Cont.) Rulesets in the Base Rulesets Cartridge

Ruleset/Ruleset Extension Points	Functionality
RECALL_DISCONNECTED_TN	Transitions telephone numbers with a Disconnected status to the Transitional status. The disconnected numbers are transitioned if the expiry period is reached. For example, for a telephone number that was disconnected on January 1, 2009, and for which the expiry period is set for 10 days later, the transition can occur only on January 10, 2009. The expiry period is configured by the service provider. This is a global ruleset and will apply to all telephone numbers regardless of the specifications used to create them.
RESERVATION_CHECK_REDEEMER	Can be configured to either of the two values: True or False. The default value is True. Setting the value to True indicates that when a reserved resource is assigned to a service configuration item, the user needs to provide the values of the Reserved For Type and Reserved For fields. The resource cannot be redeemed if the user provides incorrect values. Setting the value to False indicates that no validation occurs to check for incorrect values of the Reserved For Type and Reserved For fields, and the resource is automatically assigned to the configuration item. This is a global ruleset.
RESERVATION_EXPIRATION	Indicates the expiration periods for Long term and Short Term reservations. The default are 10 minutes and 40 days respectively. This is a global ruleset.
SYSTEM_EXPORT	Exports database entities into other systems in XML format. This downloadable file is available in a binary ZIP file that may be imported using the SYSTEM_IMPORT ruleset.
SYSTEM_IMPORT	Imports data in XML format into UIM. This ruleset accepts a input in a binary ZIP file, which the SYSTEM_EXPORT ruleset generates.
TELEPHONE_NUMBER_FORMATTING	Changes the default telephone number length and format specified in a telephone number specification. The default length and format is NNNNNNNN. You may change the length and format to suit the requirements of your country. Use the NNN-NNN-NNNN format for US telephone numbers and NNN NN NNNN for telephone numbers in Spain.
TELEPHONE_NUMBER_GRADING	Enables you to define specific telephone number patterns. For example, you can define telephone numbers that users consider lucky or can memorize easily.
TRAIL_PIPE_TOPOLOGY_EDGE	Prevents the topology from storing the service trails of pipes and thereby ensures that they are not reflected in path analysis results. The default topology behavior requires that each individual pipe be terminated on a resource. This information is stored in a topology.
VALIDATE_ADDRESS_FOR_RANGE	Validates that a given address lies within a specified address range based on the values of the address range and address parameters.
VALIDATE_RELATE_PLACES	Validates whether the parameters for a parent place and the associated child place relate to specifications.
CREATE_SUBHOLDERS_BASED_ON_SHELF_TYPE	Creates sub-holders based on the shelf specification when a card is inserted on the shelf. UIM does not have shelf and sub-holder relationship. Sub-holder creation depends on the shelf specification name and the card name. With the corresponding extension points, you can create sub-holders when: • You create or add a card in a shelf manually. • UIM auto-creates card in a shelf as defined during the design time.

Table 6-1 (Cont.) Rulesets in the Base Rulesets Cartridge

Ruleset/Ruleset Extension Points	Functionality
POPULATE_USER_GROUP	Overwrites or updates the column **CREATEDUSERGROUP** for multiple saved searches. It takes a .txt file that contains all usernames, query id, name of the saved searches whose **CREATEDUSERGROUP** column should be modified along with the **CREATEDUSERGROUP** value. See **ora_uim_savedsearch_grpupdate cartridge\readme.txt** for more details and instructions.
FORCE_DELETE_RESOURCE	Forcefully deletes resources, even if they have active assignments or references. **Caution**: This action is irreversible. Proceed with caution, as deleting a resource is your responsibility and is performed at your own risk. See **ora_uim_ForceDeleteResource/ForceDeleteRulesetInformation.pdf** and **ora_uim_ForceDeleteResource/README.txt** for input parameters and additional details.

Base Extension Points

The Base Extensionpoints cartridge contains a set of basic extension points that may be associated with rulesets.

A ruleset is associated with an extension point that identifies the point where the ruleset is run. The extension point triggers the ruleset to run whenever the extension point is called.

Extension points are identified in the corresponding ruleset extension point specifications. Each extension point specification contains a signature of a method that calls the extension point. Whenever a service calls the method, the extension point is called and the associated ruleset is run.

For example, the ruleset extension point `MPLSL3VPN_ALLOCATE_ALL_EXT` instructs the system to run the ruleset `MPLSL3VPN_ALLOCATE_ALL` at the extension point `ServiceConfigurationManager_autoAllocateServiceConfig`.

In the Base Extension points cartridge, that extension point identifies the following point name and method signature:

```
ServiceConfigurationManager.autoAllocateServiceConfig
public abstract interface void
oracle.communications.inventory.api.service.ServiceConfigurationManager.
autoAllocateServiceConfig
(oracle.communications.inventory.api.entity.ServiceConfigurationVersion)
```

As a result, any time the `ServiceConfigurationManager.autoAllocateServiceConfig` method is called for an MPLS VPN service, the extension point in the Base Extension points cartridge is called and the `MPLSL3VPN_ALLOCATE_ALL` ruleset is run. See "Overview" in *UIM Developer's Guide* for more information on extension points.

Base Extension Point Descriptions

Table 6-2 lists and describes the extension points in the Base Extensionpoints cartridge.

Table 6-2 Ruleset Extension Points in the Base Extensionpoints Cartridge

Extension Point	Functionality
ActivityManager.createActivity	Triggered when an activity is created.
ActivityManager.updateActivity	Triggered when an activity is updated.
AddressRangeManager_findAddressRange	Determines whether a particular address format pertains to a city or town based on the incoming geographic address and retrieves the corresponding characteristic values.
AddressRangeManager_validateAddressForRange	Validates whether a specific address can be associated to a particular address range.
AssignmentManager.assignResourceToTerminationPoint	Assigns a resource to a pipe termination point. Redeems the resource reservation, if any. Refers to the following method: `AssignmentManager.assignResourceToTerminationPoint(oracle.communications.inventory.api.entity.PipeTerminationPoint, java.util.Collection, java.lang.String, java.lang.String)`
AssignmentManager_unassignResource	Enables you to unassign a resource from a service configuration when disconnecting a service.
AttachmentManager_createAttachment	Creates custom involvements between inventory items. Refers to the following method: `AttachmentManager.createCustomInvolvement(java.util.Collection)`
AttachmentManager_createAttachment_Global	Global extension point to create custom involvements between inventory items. Refers to the following method: `AttachmentManager.createCustomInvolvement(java.util.Collection)`
AttachmentManager_deleteAttachment	Deletes existing custom involvements between business entities. Refers to the following method: `AttachmentManager.deleteCustomInvolvement(java.util.Collection)`
AttachmentManager_deleteAttachment_Global	Global extension point to delete existing custom involvements between business entities. Refers to the following method: `AttachmentManager.deleteCustomInvolvement(java.util.Collection)`
AttachmentManager_updateAttachment	Modifies existing custom involvements between business entities. Refers to the following method: `AttachmentManager.updateCustomInvolvement(java.util.Collection)`
AttachmentManager_updateAttachment_Global	Global extension point to modify existing custom involvements between business entities. Refers to the following method: `AttachmentManager.updateCustomInvolvement(java.util.Collection)`
BaseConfigurationManager_approveConfigurationVersion	Changes a configuration version status to Issued status. Refers to the following method: `BaseConfigurationManager.approveConfigurationVersion(oracle.communications.inventory.api.entity.common.InventoryConfigurationVersion)`

Table 6-2 (Cont.) Ruleset Extension Points in the Base Extensionpoints Cartridge

Extension Point	Functionality
BaseConfigurationManager_assignResource	Assigns a resource to an entity configuration, redeeming the resource reservation if any. Refers to the following method. `BaseConfigurationManager.assignResource(oracle.communications.inventory.api.entity.common.InventoryConfigurationItem, oracle.communications.inventory.api.entity.common.ConsumableResource, java.lang.String, java.lang.String)`
BaseConfigurationManager_autoAllocate	Assigns resources to entityConfigurationItems based on rules. This method is called when the specOption is selected from the specification options associated with a configuration item. You can extend this to automatically assign a resource on a particular configuration item after selecting a specification option. You can extend this method for any of the three configurations: Pipe, Logical Device, and Network.
BaseConfigurationManager_autoAllocateInvConfiguration	Assigns resources automatically to the configuration items for the specified configuration version, based on rules. This method is called when the Auto assign option is selected from the Related Pages menu. You can extend this method to perform automatic assignment for an entire configuration version. You can extend this method for any of the three configuration items: Pipe, Logical Device, and Network.
BaseConfigurationManager_autoAssignResource	Assigns resources to the configuration item based on rules. You can extend this method to perform automatic assignment for the newly-created configuration items. You can extend this method for any of the three configuration items: Pipe, Logical Device, and Network.
BaseConfigurationManager_automateConfiguration	Provides an extension point for custom rulesets that automate configuration. The default implementation does nothing.
BaseConfigurationManager_cancelConfigurationVersion	Cancels a configuration version. The status to which the configuration version is transitioned varies based on the starting status: • In Progress, Designed—Canceled • Designed—Canceled • Issued—Pending Cancel • Pending Cancel—Issued
BaseConfigurationManager_completeConfigurationVersion	Completes a configuration version. The status to which the configuration version is transitioned varies based on the starting status: • Issued—Completed • Pending Cancel—Canceled
BaseConfigurationManager_issueConfigurationVersion	Changes a configuration version status to Issued.
BaseConfigurationManager_unallocateInventoryConfigurationItems	Unassigns resources from configuration items. The configuration version cannot be in a Completed or Canceled status.
BaseConfigurationManager_populateCustomProperties	Populates custom properties for consumable resources.
BaseConfigurationManager_populateCustomProperties2	Populates custom properties for reference-enabled configurations.
BaseConfigurationManager_validateInventoryConfiguration	Validates the configuration versions. This method contains a set of basic validation procedures for date validations. You can extend this method using a ruleset. This method is called when you select Validate from the Actions list. You can extend this method to add more validations for a configuration version. You can extend this method for any of the three configuration items: Pipe, Logical Device, and Network.

Table 6-2 (Cont.) Ruleset Extension Points in the Base Extensionpoints Cartridge

Extension Point	Functionality
BaseInvManager_completeCreate	Provides an extension point for validating entity creation during the completion of a business interaction.
BaseInvManager_completeDelete	Provides an extension point for validating entity deletion as part of a business interaction completion.
BaseInvManager_completeUpdate	Provides an extension point for validating entity modification as part of a business interaction completion.
BaseInvManager_validateDisassociation	Provides an extension point for validating the disassociation of items from business interactions.
BaseInvManager_validateTransfer	Provides an extension point for validation during the transfer of items from one business interaction to another.
BOMManager.populateAdditionInfoOnResource	Captures additional information about a resource while calculating a Bill of Materials (BOM).
BOMManager.populateAdditionInfoOnActivity	Captures additional information about an activity while calculating a BOM.
BusinessInteractionManager_addChildBusinessInteraction	Adds an additional child business interaction to the parent business interaction. This method is called when you associate one or more child business interactions in the Business Interaction Tree hierarchy. You can extend this method to include additional validations or business logic.
BusinessInteractionManager_cancelBusinessInteraction	Cancels one or more business interactions and restores the inventory status of the business interaction items accordingly. The context is set to Current. This method is called when you try to cancel the business interaction. You can extend this method to add more validations or business logic while canceling the business interaction. The context reverts to Current after the method has run.
BusinessInteractionManager_completeBusinessInteraction	Completes one or more business interactions and updates the inventory status of the business interaction items accordingly. This method is called when you try to complete the business interactions. You can extend this to add more validations or business logic while completing the business interactions. The context reverts to Current after the business interactions are completed.
BusinessInteractionManager_createBusinessInteractionAttachment	Creates an attachment to a business interaction.
BusinessInteractionManager_deleteBusinessInteractions	Deletes one or more business interactions. Sets the business interaction context to Current.
BusinessInteractionManager_getEntityAction	Gets the entity action during a Web service request to create a business interaction for service fulfillment.
BusinessInteractionManager_preProcessInteractionItems	Provides an extension point for sorting items during the execution of the Process Interaction Web service. You can create rulesets that ensure that items are processed in the proper order to account for dependencies among items.
BusinessInteractionManager_removeChildBusinessInteraction	Removes one or more child business interactions from a parent business interaction. This method is called when you try to disassociate one or more child business interactions within the Business Interaction Tree hierarchy. You can extend this method to include additional validations or business logic.
BusinessInteractionManager_sendRequest	Provides an extension point for developing logic for construction for constructing an interaction exchange or request with a third party system.

Table 6-2 (Cont.) Ruleset Extension Points in the Base Extensionpoints Cartridge

Extension Point	Functionality
BusinessInteractionManager_switchContext	Changes the current business interaction context to another business interaction context or reverts it to Current context. If a versioned entity is passed, it is refreshed from the datastore. The context of the version object is changed to the target business interaction context or set to null if it is the Current context. The method returns a null value if the refreshed entity does not exist in the target business interaction or the Current context. You should call this method within the transaction scope.
BusinessInteractionManager_transferItems	Transfers business interaction items to the target business interaction. This method is called when you transfer items from one business interaction to another. You can extend this method to include additional validations or business logic.
BusinessInteractionManager_updateBusinessInteraction	Updates one or more business interactions.
BusinessInteractionUtils_associateWithBusinessInteraction	Associates the versioned object to a business interaction. It creates a new business interaction item if there is none available; otherwise, it returns the existing business interaction item. This method is called when you modify a business-interaction-enabled object in the context of a business interaction. You can extend this method to include additional validations or business logic.
ConditionManager_createConditions	Allows you to modify the way conditions are created in UIM. The API call associated with this extension point creates conditions for the resources in the set of Condition entities sent to the API call. The API call also creates conditions for the resources that are preconfigured with the resources sent as part of the call. Conditions are not currently supported for the configuration items Service and Geographic Place. The UIM base implementation requires that the reason, type, and resource be provided in the condition; otherwise, it will throw an error. This extension point may be used to add more entities that can have conditions attached or to notify that a condition has been created.
ConditionManager_deleteConditions	Enables you to alter the way conditions are deleted. The API call associated with this extension point scans the collection of resources supplied and deletes the conditions associated with the resources. The API call also removes the conditions from the resources that are preconfigured with the resources specified in the collection.
ConditionManager_updateConditions	Allows you to modify the way conditions are inventoried in UIM. The API call associated with this extension point updates the conditions inventoried in UIM with the resources in the set of Condition entities sent to the API call. The API call also updates conditions for the resources that are preconfigured with the resources sent as part of the call. Conditions are not currently supported for the configuration items Service and Geographic Place. The UIM base implementation requires that the reason, type, and resource be provided in the condition; otherwise, it will throw an error.
ConsumerManager_checkAvailability	Determines whether a resource has already been consumed or is available for consumption.
CreatePipes	Creates pipes.

Table 6-2 (Cont.) Ruleset Extension Points in the Base Extensionpoints Cartridge

Extension Point	Functionality
CustomNetworkAddressManager_createCustomNetworkAddress	Creates custom network address entities. It allows you to create custom network addresses with or without specifications. If you create a custom network address with a valid specification, the API associated with the extension point assigns default values to all data elements and characteristics. The API only performs mandatory checks if you create a custom network address without specifications. The operation to create custom network addresses is atomic. Either all of the input custom network addresses are created or none. The API also runs the federation manager to fetch the federated data and validates that the custom network addresses are unique. This extension point may be used to perform preliminary checks to ensure that the data supplied has mandatory data elements. The extension point also enables you to add custom network addresses to inventory groups.
DeleteCustomObjects	Deletes the custom objects.
DeleteEquipment	Deletes the equipment.
DeletePhysicalDevices	Deletes the physical devices.
DeletePhysicalPorts	Deletes the physical ports.
EntityIdGenerator_getId_1	Returns an automatically-generated ID for an entity. The automatically-generated ID is a numeric sequential value, which may optionally contain a prefix and suffix. The prefix and suffix may be defined in the Entity Identification specification. For the extension point to run correctly, the entity specification should not be configured for a manual ID, and the ID field should be empty. You may override this extension point with your extension point to generate a custom ID for the entity.
EntityIdGenerator_getId_2	Returns an automatically-generated ID for a child entity. The automatically-generated ID is a numeric sequential value, which may optionally contain a prefix and suffix. The prefix and suffix may be defined in the Entity Identification specification. For the extension point to run correctly, the entity specification should not be configured for a manual ID, and the ID field should be empty. You may override this extension point with your extension point to generate a custom ID for the child entity.
EquipmentManager_copyPhysicalDevices	Copies physical devices, including other physical devices in their hierarchies. Ports and connectors provided by the devices are also copied. Associations, mappings, and preconfigurations are not copied.
EquipmentManager_createEquipment	Creates equipment entities and the designated ports, connectors, and equipment holders based on specifications.
EquipmentManager_createPhysicalDevices	Creates physical devices along with provided ports and connectors defined in the specification.
EquipmentManager_createPhysicalPorts	Creates physical ports provided by other entities.
EquipmentManager_generateEquipmentLabel	Generates a customized label for equipments, ports, and connectors on the Equipment Tree View page.
EquipmentManager_generateEquipmentHolderLabel	Generates a customized label for equipment holders on the Equipment Tree View page.
EquipmentManager_updateEquipment	Updates equipment entities and the designated ports, connectors and equipment holders in UIM.

Table 6-2 (Cont.) Ruleset Extension Points in the Base Extensionpoints Cartridge

Extension Point	Functionality
EquipmentManager_updatePhysicalDevices	Updates data elements and characteristics in a physical device.
FederationManager_assignFederationData	Assigns federated data.
FederationManager_associateFederationData	Associates federated data.
FederationManager_createFederationData	Creates federated data.
FederationManager_deleteFederationData	Deletes federated data.
FederationManager_findFederationData	Finds federated data.
FederationManager_referFederationData	References federated data.
FederationManager_reserveFederationData	Reserves federated data.
FederationManager_updateFederationData	Updates federated data.
FederationManager_validatePreUpdateFederationData	Validates federated data before update.
InventoryEntityLifeCycleEventListener_postLoad	Listens to the postLoad life cycle event. Occurs when a persistent instance is loaded from the data store.
InventoryEntityLifeCycleEventListener_postPersist	Listens to the postPersist event. Occurs after persisting an instance into the database.
InventoryEntityLifeCycleEventListener_postRemove	Listens to the postRemove event. Occurs after removing a persistent instance from the database.
InventoryEntityLifeCycleEventListener_postUpdate	Listens to the postUpdate event. Occurs after updating a persistent instance in the database.
InventoryEntityLifeCycleEventListener_prePersist	Listens to the prePersist event. Occurs before persisting an instance in the database.
InventoryEntityLifeCycleEventListener_preRemove	Listens to the preRemove event. Occurs before removing a persistent instance in the database.
InventoryEntityLifeCycleEventListener_preUpdate	Listens to the preUpdate event. Occurs before updating a persistent instance in the database.
InventoryGroupManager_associateInventoryGroupsToPersistent	Allows you to customize UIM behavior when associating an inventory group to a persistent entity, such as party.
InventoryGroupManager_associatePersistentToInventoryGroup	Allows you to customize UIM behavior when associating multiple persistent entities to an inventory group.
InventoryGroupManager_createInventoryGroup	Allows you to customize UIM behavior when you create an inventory group.
InventoryGroupManager_deleteInventoryGroup	Allows you to customize UIM behavior when you delete an inventory group.
InventoryGroupManager_deleteInvGroupRef	Allows you to customize UIM behavior when you delete an inventory group reference row.
InventoryGroupManager_updateInventoryGroup	Allows you to customize UIM behavior when you update an inventory group.
IPAddressManager_createIPAddress_Global	Extension point on IP address creation.
IPAddressManager_deleteIPAddress_Global	Extension point on IP address deletion.
IPAddressManager_updateIPAddress_Global	Extension point IP address update.

Table 6-2 (Cont.) Ruleset Extension Points in the Base Extensionpoints Cartridge

Extension Point	Functionality
IPNetworkManager_createIPNetworks_Global	Extension point on IP network creation.
IPNetworkManager_deleteIPNetworks_Global	Extension point on IP network deletion.
IPNetworkManager_updateIPNetworks_Global	Extension point on IPv4 address creation.
IPv4AddressManager_createIPAddresses_Global	Extension point on IPv4 address creation.
IPv4AddressManager_deleteIPAddresses_Global	Extension point on IPv4 address deletion.
IPv6AddressManager_createIPAddresses_Global	Extension point on IPv6 address creation.
IPv6AddressManager_deleteIPAddresses_Global	Extension point on IPv6 address deletion.
IPv6AddressManager_updateIPAddresses_Global	Extension point on IPv6 address update.
LogicalDeviceAccountManager_createLogicalDeviceAccounts	Creates logical device accounts in UIM.
LogicalDeviceAccountManager_deleteLogicalDeviceAccounts	Deletes logical device accounts in UIM.
LogicalDeviceAccountManager_updateLogicalDeviceAccounts	Modifies logical device accounts in UIM.
LogicalDeviceManager_copyLogicalDevices	Copies logical devices, including logical devices in the hierarchies. Provided logical devices are also copied. Associations, relationships, and mappings are not copied.
LogicalDeviceManager_createLogicalDevice	Creates logical device entities and their designated device interfaces and sub device interfaces based on the specification. Device interfaces can also provide other device interfaces. The number of device interfaces to be created is determined by the minimum value defined in the specification relationships.
LogicalDeviceManager_deleteLogicalDevice	Delete logical devices, designated device interfaces, and sub device interfaces. Also deletes associations between designated device interfaces and physical ports/connectors. The extension point does not delete entities with active assignments or reservations.
LogicalDeviceManager_updateLogicalDevice	Modifies the data elements and characteristics of a logical device in UIM. If logical device is deactivated, all designated device interfaces and sub device interfaces are also deactivated. A logical device cannot be deactivated if the designated device interfaces have an active assignment or reservation. If a logical device is activated, all designated device interfaces and sub device interfaces are also activated.
MediaResourceManager_createMediaStreams	Creates a media stream entity based on the input specification and data.
MediaResource_updateMediaStreams	Updates characteristics and data elements in a media stream entity.
NetworkManager_createNetwork	Creates a network entity.
NetworkManager_createNetworkEdge	Creates a network edge.
NetworkManager_createNetworkNode	Creates a network node.
NetworkManager_deleteNetworkEdges	Deletes network edges.

Table 6-2 (Cont.) Ruleset Extension Points in the Base Extensionpoints Cartridge

Extension Point	Functionality
NetworkManager_deleteNetworkNodes	Deletes network nodes.
NetworkManager_updateNetwork	Updates characteristics and data elements in a network entity.
NetworkManager_updateNetworkEdges	Updates characteristics and data elements in a network edge.
NetworkManager_updateNetworkNodes	Updates characteristics and data elements in a network node.
NetworkManager.setEdgeColorAndSize	Enables customization of pipe and connectivity colors and sizes (edge widths) within the network visualization.
PartyManager_createParty	Allows you to customize UIM behavior when creating parties.
PartyManager_deleteParty	Allows you to customize UIM behavior when deleting parties.
PartyManager_updateParty	Allows you to customize UIM behavior when modifying parties.
PathAnalysisManager_findPaths	Initiates a path analysis using enabled pipe and custom filtering criteria or analysis mode as parameters.
PathAnalysisManager_findSecondaryPaths	Finds secondary paths for a given primary path. Process the list of primary pipes and identifies available secondary path based associated network properties.
PathAnalysisManager_postProcessFindSecondaryPaths	Provides an extension point for customizing the secondary path logic. Occurs after secondary paths are determined but before results are returned.
PathAnalysisManager_preProcessFindSecondaryPaths	Provides an extension point for customizing the secondary path logics. Occurs before secondary paths are determined.
PipeConfigurationManager_addTrailPipeSegments	Creates pipe configuration items for the pipe segments return by path analysis.
PipeConfigurationManager_addTransportItems	Creates configuration item for each pipe in a given path in the transport group.
PipeConfigurationManager_deleteEnablementConfigItems	Deletes the configuration items from the first transport group for pipes that no longer enable the service trail.
PipeConfigurationManager_reassignEnablementConfigItems	Reassigns enablement configuration items from a previous configuration version.
PipeConfigurationManager_unassignEnablementConfigItems	Unassigns enablement configuration items from a previous configuration version.
PipeConfigurationManager_updatePipeConfiguration	Updates a pipe configuration for the first path. Creates gap pipes and makes pipe assignments.
PipeManager_autoConfigure	Provides an extension point for automating enablement of a trail pipe with no pipe configuration.
PipeManager_createDefaultPipes	Creates pipe entities based on the input specification.
PipeManager_createTrailPipe	Creates a trail pipe and its two provided termination points.
PipeManager_deletePipeEnablement	Deletes the enablements for the trail for the first path.
PipeManager_deletePipes	Deletes pipe entities. If the pipe is part of a trail, it cannot be deleted until the trail is updated to remove this pipe from the enablement The two pipe termination points are also deleted if no other pipes are sharing them.
PipeManager_pipeExportPath	Provides an extension point for the export button in a manual configuration for pipe so that you can extend it to choose the format of excel and the content. A sample ruleset "PIPE_EXPORT_PATH" is included in the baserulesets cartridge. PIPE_EXPORT_PATH writes custom data to the output stream, in pipe manual configure page, when the export button is used.

Table 6-2 (Cont.) Ruleset Extension Points in the Base Extensionpoints Cartridge

Extension Point	Functionality
PipeManager_pipeExportAllPaths	Provides an extension point for the **exportAllPaths** button in a manual configuration for pipe so that you can extend it to choose the format of excel and the content. A sample ruleset "PIPE_EXPORT_ALL_PATH" is included in the baserulesets cartridge. PIPE_EXPORT_ALL_PATHS writes custom data to the output stream, in pipe manual configure page, when the **exportAllPaths** button is used.
PipeManager_updatePipes	Updates the characteristics and data elements in a pipe entity.
PipeManager_updateTrailPipe	Updates a trail pipe.
PipeManager_updateTrailPipeSegments	Updates a pipe configuration for a given path (transport group). Creates gap pipes and makes pipe assignments.
PlaceConfigurationManager_autoAssignResource	Allows you to customize automatic assignment logic for inventory resources.
PlaceConfigurationManager_validatePlaceConfiguration	Validates the list of place configurations and returns a list of valid place configurations.
PlaceManager_createGeographicPlace	Creates one or more entities based on GeographicPlace specifications.
PlaceManager_deleteGeographicPlace	Deletes one or more Geographic Place entities from the database.
PlaceManager_formatGeocodingAddress	Converts a UIM address into a string of address lines in a specific format, similar to a postal address, required by the Oracle Spatial GeoCoder.
PlaceManager_relatePlaces	Creates a relationship between two GeographicPlace entities, which creates a PlaceRel entity. The PlaceRel entity represents a relationship between two Geographic Place instances.
PlaceManager_relatePlaces2	Additional extension point on place relationship.
PlaceManager_updateGeographicPlace	Updates the data elements of one or more Geographic Place entities.
ReservationManager_checkRedeemer	Allows you to check whether a resource is redeemable. If the associated API returns a true value to indicate that a particular reserved resource is redeemable, you must provide values for Reserved For and Reserved For Type fields to redeem the resource. This extension point is used in Redeem Reservation work flows.
ReservationManager_deleteReservation	Allows you to delete resource reservations created in UIM.
ReservationManager_expireReservation	Allows you to expire reservations that have an expiry date prior to the current date.
ReservationManager_extendReservation	Allows you to extend the expiry dates for reservations based on the expiration rules.
ReservationManager_reserveResource	Allows you to create reservations for a group of resources. The API associated with the extension point is used in work flows where reservations are created for resources. The API is used in both UI and work flows for Web services.
ReservationManager_setReservationExpiry	Allows you to configure the expiry dates for reservations based on the expiration rules. You can configure the reservation rules using the RESERVATION_EXPIRATION ruleset.
ReservationManager_unreserveResource	Allows you to delete the reservations for resources you previously reserved.
ReservationManager_updateReservation	Allows you to update the input collection of reservations.
ReservationManager_updateReservation2	Updates the collection of reservations with modified values.
ServiceConfigurationManager_createServiceConfiguration	Create a new configuration version for a service.

Table 6-2 (Cont.) Ruleset Extension Points in the Base Extensionpoints Cartridge

Extension Point	Functionality
ServiceConfigurationManager_refreshTopology	Refreshes the topology for the specified version based on rules.
ServiceConfigurationVersionWorker_validateCompleteConfigurationForCustomState	Validates the Service configuration on completion for a custom state.
ServiceConfigurationVersionWorker_validateCreateConfigurationForCustomState	Validates the Service configuration on creation for a custom state.
ServiceManager_cancelService	Starts the cancel operation for a service. The status to which the service is transitioned depends on the starting status.
ServiceManager_completeService	Starts the complete operation for a service. The status to which the service is transitioned depends on the starting status.
ServiceManager_createService	Create one or more service entities.
ServiceManager_disconnectService	Transitions the service to Pending Disconnect status. Creates a new configuration version and unassigns or dereferences current assignments and references. The configuration version that is created must be completed to transition the service to Disconnected status.
ServiceManager_resumeService	Resumes a service by transitioning it from Suspended status to In Service status. Does not change the status of assignments or references on the active configuration.
ServiceManager_suspendService	Transitions the service to Suspended status. Does not change assignments or references in the active configuration version.
ServiceManager_updateService	Changes service characteristics and data elements.
SpecManager_getEditMask	Enables you to customize code to retrieve the edit mask specified in the Telephone Number specification. The default edit mask for a telephone number is '##########'.
TelephoneNumberManager_createTelephoneNumbers	Creates telephone numbers in UIM.
TelephoneNumberManager_deleteTelephoneNumbers	Deletes telephone numbers from UIM.
TelephoneNumberManager_recallDisconnectedTelephoneNumber	Manages the transition of the telephone number assignment status from Disconnected to Transitional, and subsequently to Unassigned. This extension point is called by the TN_AGING ruleset.
TelePhoneNumberManager_transitionOnSnapBack	Transitions the telephone number status from Ported to Unassigned for telephone numbers that were previously ported out.
TelephoneNumberManager_updateTelephoneNumbers	Updates telephone numbers in UIM.
TimeoutEventListener_timerExpired	Notifies timer expiry events in UIM.
TopologyMapper_createEnabledByPipesTopologyEdge	Enables you to track pipe topology in non-optimal cases. The default topology behavior requires that each individual pipe be terminated on a resource to be stored in topology. This mechanism is used so Service Trails are not stored in topology and path analysis will not find those paths. There are two situations where this default behavior is not optimal. A T1 is riding a T3 and the T1 is not terminated, because it is terminated on the T3, and a Local Loop scenario where the provisioner wishes to assign to the local loop and not the individual cable pairs. For these cases, extend this method using either an 'INSTEAD' ruleset extension point for the specification you wish to track in topology. If this method returns 'TRUE' topology will use the termination points of the parent connection as the termination points of the child.

Table 6-2 (Cont.) Ruleset Extension Points in the Base Extensionpoints Cartridge

Extension Point	Functionality
TopologyProfileMapper_createTopologyProfile	Enables you to create nodes for logical devices, physical devices, device interfaces, equipment, networks, network nodes, geographic sites, locations, edges for pipes and networks. You must assign the nodes and edges to a service configuration for them to appear in the service topology. This extension point is called when you select **View Topology** on the Service Topology work area for the first time. This extension point also calls the extension point TopologyProfileMapper_processTopologyProfileExtension after it runs.
TopologyProfileMapper_processTopologyProfileExtension	Enables you to access the TopologyProfile TopologyEdgeProfile and TopologyProfile TopologyNodeProfile lists to add, modify, and delete edges and nodes respectively. You should associate this extension point to the ServiceConfigurationVersion specification that needs to be extended. You should run this extension point after the refreshServiceTopology and createTopologyProfile extension points have run.
TopologyProfileMapper_refreshServiceTopology	Refreshes the topology for a service topology profile. By default, this extension point deletes the current service topology profile and then recreates it. This extension point calls the extension point 'TopologyProfileMapper_processTopologyProfileExtension to run custom implementations.
TransitionManager_validateBusinessStateTransitions	Lets you overwrite existing validation rules for business state transitions or perform additional tasks before or after a validation. This extension point is called before an entity transitions to a new business state.
TransitionManager_validateObjectStateTransition	Lets you overwrite existing validation rules for object state transitions or perform additional tasks before or after a validation. This extension point is called before an entity transitions to a new object state.
UpdateCustomObjects	Updates the custom objects.
UpdatePhysicalPorts	Updates the physical ports.
WorkflowManager.assignActivity	Used to assign activities to users or resources.
WorkflowManager.canActivityStartNow	Checks whether an activity can be started.
WorkflowManager.canActivityCompleteNow	Checks whether an activity can be completed now.
WorkflowManager.completeActivity	Triggered when an activity is completed.
WorkflowManager.createActivities	Triggered when multiple activities are created.
WorkflowManager.insertActivity	Triggered when an activity is inserted into an existing workflow.
WorkflowManager.isAllowedToWorkOnActivity	Checks whether an activity can be assigned to a particular user.
WorkflowManager.startActivity	Triggered when an activity starts.
EquipmentManager_addEquipmentToEquipmentHolders	For an existing shelf, run this extension point when an equipment is added to the shelf holder. You can create sub holders using the CREATE_SUBHOLDERS_BASED_ON_SHELF_TYPE ruleset.
EquipmentManager.addEquipmentToEquipmentHolders_createShelfCase	Use this extension point when a shelf is created. Creating shelf auto creates the holder and adds cards as designed. This extension point is used in auto add equipment to the holder case. You can create sub holders using the CREATE_SUBHOLDERS_BASED_ON_SHELF_TYPE ruleset.

7

NFV Orchestration Base Cartridges

This chapter provides information about the Oracle Communications Unified Inventory Management (UIM) NFV Orchestration base cartridges. These cartridges are required only for an NFV Orchestration implementation.

- NFV Orchestration Base Specifications Cartridge
- NFV Orchestration Base Tags Cartridge

See *UIM NFV Orchestration Implementation Guide* for more information.

NFV Orchestration Base Specifications Cartridge

NFV Orchestration functionality uses entities defined by specifications in the **OracleComms_NSO_BaseCartridge** cartridge. Although these entities are defined in Design Studio as standard UIM entities, they have different labels in UIM. For example, the Network Service Request and VNF Request specifications are defined in Design Studio as Business Interaction specifications, but entities based on them are labeled as Orchestration Requests in UIM.

Business Interaction Specification

Table 7-1 lists and describes the Business Interaction specifications included in the NFV Orchestration Base Specifications cartridge. Entities based on these specifications are labeled as Orchestration Requests in UIM.

Table 7-1 Business Interaction Specifications

Specification	Description
Network Service Request	Defines a network service orchestration request. All network service lifecycle operations are controlled by the network service request. Includes the **Service ID** characteristic, which stores the ID of the service created as a result of the request.
VNF Request	Defines a VNF orchestration request. All VNF life cycle operations are controlled by the VNF request.
PNF Request	Defines a PNF orchestration request. All PNF life cycle operations are controlled by the VNF request.

Custom Object Specification

Table 7-2 lists and describes the Custom Object specifications included in the NFV Orchestration Base Specifications cartridge.

Table 7-2 Custom Object Specifications

Specification	Description
Availability Zone	Represents a grouping of resources based on availability characteristics. This information is captured during the VIM discovery process. In OpenStack, availability zones enable you to arrange OpenStack compute hosts into logical groups and provide a form of physical isolation and redundancy from other availability zones, such as by using a separate power supply or network equipment. Includes the following characteristics: • **Disk Total**: Total disk size of the availability zone. • **Memory Total**: Total memory size of the availability zone. • **VCPU Total**: Total number of vCPUs in the availability zone. • **Disk Used**: The amount of disk space currently used by the availability zone. • **Memory Used**: The amount of memory currently used by the availability zone. • **VCPU Used**: The number of vCPUs currently used in the availability zone. • **VIM ID**: The ID of the VIM associated with this availability zone.
Flavor	Defines the compute, memory, and storage capacity of computing instances. A flavor is an available hardware configuration for a server. It defines the size of a virtual server that can be launched. Includes the following characteristics: • **External ID**: ID of the flavor in the NFVI. • **Shared**: Indicates whether the flavor is shared. • **RAM**: RAM value of flavor. • **Root Disk**: Disk value of the flavor. • **vCPUs**: CPU value of the flavor. • **VIM ID**: The VIM ID.
Host	Represents a compute host, a physical host dedicated to running compute nodes. These hosts are captured during the VIM discovery process. Includes the following characteristics: • **Disk Total**: Total disk size of the host. • **Disk Used**: Amount of disk space used on the host. • **Egress Node**: Indicates whether this is an egress host. • **External ID**: ID of the host. • **Host IP**: IP address of the host. This field is not used by default. • **Ingress Node**: Indicates whether this host is the ingress host. • **Memory Total**: Total memory size of the host. • **Memory Used**: Memory used on the host. • **OVS ID**: OVS ID of the host. • **Tunnel IP**: Tunnel IP of the host. • **vCPUs Total**: Total number of vCPUs on the host. • **vCPUs Used**: Number of vCPUs on the host. • **VIM ID**: ID of the VIM to which this host belongs.
NSSubscriber	Represents a network service endpoint.Includes the following characteristics: • **IP Address**: IP address of the endpoint. • **Service Location**: Service location of the endpoint. • **VLAN ID**: VLAN ID of the endpoint.
OpenDayLight	This specification is deprecated.
OpenVirtualSwitch	This specification is deprecated.

Table 7-2 (Cont.) Custom Object Specifications

Specification	Description
SDN	Represents a software-defined networking (SDN) controller that provides control and management functionality in a network domain. Includes the following characteristics: • **Host**: Host name of the SDN controller. • **Password**: Password details of the SDN controller. • **Port**: Port details of the SDN controller. • **SDN Type**: Type of the SDN. • **User Name**: User name details of the SDN controller.
VIM	Represents a VIM that manages resources in your NFVI. Includes the following characteristics: • **Domain Name**: Domain name of the VIM. • **Host**: Host name or IP address of the VIM. • **Disk Overcommitted Ratio**: Over-commit disk ratio of the VIM. • **Memory Overcommitted Ratio**: Over-commit memory ratio details of the VIM. • **CPU Overcommitted Ratio**: Over-commit vCPU ratio details of the VIM. • **Password**: Password for the VIM. • **Port**: Port value for the VIM. • **SSI Enabled**: Indicates whether the VIM is SSL enabled or not. • **Tenant Name**: Tenant name of the VIM. • **User Name**: User name of the VIM. • **Version**: Version of the VIM. • **VIM Type**: Type of the VIM.
Virtual Data Center	Represents a virtual data center. There is a one-to-one relation ship between Virtual Data Centers and VIMs in NFV Orchestration. Includes the following characteristics: • **Disk Total**: Total disk size allocated to the virtual data center. • **Disk Used**: The amount of disk space currently used by the virtual data center. • **Memory Total**: Total memory size allocated to the virtual data center. • **Memory Used**: The amount of memory currently used by the virtual data center. • **Tenant ID**: ID of the tenant for which the VDC manages resources. • **vCPUs Total**: Total number of vCPUs in the virtual data center. • **vCPUs Used**: The number of vCPUs currently used in the virtual data center.

Device Interface Specification

Table 7-3 lists and describes the Device Interface specifications included in the NFV Orchestration Base Specifications cartridge.

Table 7-3 Device Interface Specifications

Specification	Description
CPD	Represents a port on the VNF. Connection points connect virtual links to VNFs. They represent the virtual interfaces and physical interfaces of the VNFs and their associated properties and other metadata.

Inventory Group

Table 7-4 lists and describes the Inventory Group specifications included in the NFV Orchestration Base Specifications cartridge.

Table 7-4 Inventory Group Specifications

Specification	Description
ServingArea	Represents a storing serving area. Contains service locations.

IP Address Resource Extension

Table 7-5 lists and describes the IP Address Resource Extension specifications included in the NFV Orchestration Base Specifications cartridge. There are separate specifications for IP networks, subnets, and addresses.

IP networks are either created or referenced in the Network Service configuration. During activation, the corresponding network, subnet, and ports are created in the VIM on which the VNF virtual machine is deployed.

Table 7-5 IP Resource Extension

Specification	Description
IPv4 Address	Represents a port in OpenStack or other Cloud infrastructure. Includes the following characteristics: **External ID**: Cloud ID of the IP address. **External Network ID**: Cloud ID of the network hat includes this IP address. **External Subnet ID**: Cloud ID of the subnet that includes this IP address. **Floating IP**: Any associated floating IP associated with this. **Gateway IP**: Gateway IP. **Is Floating IP Created**: Floating IP is created or not.
Pv4Network	Represents the IP network of a VNF.
IPv4 Subnet	Represents an NFVI Cloud subnet. Includes the following characteristics: **External ID**: Cloud ID of the subnet. **External Router Name**: External router associated with the subnet. **External Network ID**: Cloud ID of the network. **Gateway IP**: Gateway IP of the subnet. **IP Version**: Not used by default. **Is DHCP Enabled**: Indicates whether subnet is DHCP enabled. **Is Security Enabled**: Indicates whether subnet is security enabled. **Network ID**: Not used by default. **Provider Network Name**: Not used by default. **Tenant ID**: Tenant ID where this subnet is present. **VIM ID**: VIM ID where this subnet is present.

Network

Table 7-6 lists and describes the Network specification included in the NFV Orchestration Base Specifications cartridge.

> **Note**
>
> This specification is present in the cartridge but not used.

Table 7-6 Network Specifications

Specification	Description
VIM Network	Not used.

Network Address Domain

Table 7-7 lists and describes the Network Address Domain specification included in the NFV Orchestration Base Specifications cartridge.

Table 7-7 Network Address Domain Specification

Specification	Description
NsoDomain	Represents an OpenStack network in NFV Orchestration. IP addresses in each domain and its corresponding network must be unique. Includes the following characteristics: • **External ID**: Cloud ID of the network. • **Shared**: Indicates whether the network is shared. • **Provider Network Type**: Not used by default. • **Provider Physical Network**: Not used by default. • **Provider Segmentation ID**: Not used by default. • **Router Type**: Not used by default. • **VLAN ID**:VLAN ID of the network.

Place

Table 7-8 lists and describes the Place specification included in the NFV Orchestration Base Specifications cartridge.

Table 7-8 Place Specification

Specification	Description
ServiceLocation	Used to create service locations based on the **Service Location** fields in the NSSubscriber entities that represent network service endpoints.

Extension Point

Table 7-9 lists and describes the extension points included in the NFV Orchestration Base Specifications cartridge.

Table 7-9 Extension Points

Specification	Description
ConsumerHelper_getDataCenterLookupIdentifier	Returns the representation of the dynamic property in the JSON request for network service instantiation. Can be used to define a different data center lookup identifier for a Network Service specification. The default implementation considers the **Service Location** field in the network service endpoint as the data center lookup identifier.
ConsumerHelper_postNSTerminate	Called after a network service is terminated.
CustomObjectManager_updateCustomObjects	Called during Custom Object update operation.
NetworkServiceDesignManager_processChange	Implements the design-and-assign logic or cleans up the resources when a network service is updated.
NetworkServiceDesignManager_processCreate	Implements the design-and-assign logic for a network service when the network service is instantiated.
NetworkServiceDesignManager_processDisconnect	Cleans up the network service resources when the network service is terminated.
NetworkServiceManager_designInstantiate_Global	Used to design the Network Service during instantiation.
NetworkServiceManager_designUpdate_Global	Designs a network service for update.
NetworkServiceManager_processTechnicalActions	Activates or removes the resources in a VIM for each network service.
NetworkService_createAndAssignIPNetworks	Creates and assigns entities that correspond to a cloud network in a network service configuration.
NetworkService_createIPSubnet	Creates IP subnets for VLDs that are not referenced. Internally called from createAndAssignIPNetworks.
VNFServiceDesignManager_processChange	Implements the design-and-assign logic for a VNF service when the network service is updated.
VNFServiceDesignManager_processCreate	Implements the design-and-assign logic for the VNF service when a network service is instantiated with a VNF.
VNFServiceDesignManager_processDisconnect	Cleans up the VNF service resources when a network service is terminated.
VNFServiceHelper_createVNF	Creates a VNF device.
VNFServiceManager_processTechnicalActions	Activates or removes the resources in a VIM for each VNF service.

Enabled Extension Point

Table 7-10 lists and describes the enabled extension points included in the NFV Orchestration Base Specifications cartridge.

Table 7-10 Enabled Extension Points

Specification	Description
CustomObjectSpecification_CustomObjectManager_updateCustomObjects	Enabled extension point for the Custom Object specification and the CustomObjectManager_updateCustomObjects extension point.
ServiceConfigurationManager__NetworkServiceDesignManager_processChange	Enabled extension point for the Service Configuration specification and the NetworkServiceDesignManager_processChange extension point.
ServiceConfigurationManager__NetworkServiceDesignManager_processCreate	Enabled extension point for the Service Configuration specification and the NetworkServiceDesignManager_processCreate extension point.
ServiceConfigurationManager__NetworkServiceDesignManager_processDisconnect	Enabled extension point for the Service Configuration specification and the NetworkServiceDesignManager_processDisconnect extension point.
ServiceConfigurationManager__NetworkServiceManager_processTechnicalActions	Enabled extension point for the Service Configuration specification and the NetworkServiceManager_processTechnicalActions extension point.
ServiceConfigurationManager__VNFServiceDesignManager_processChange	Enabled extension point for the Service Configuration specification and the VNFServiceDesignManager_processChange extension point.
ServiceConfigurationManager__VNFServiceDesignManager_processCreate	Enabled extension point for the Service Configuration specification and the VNFServiceDesignManager_processCreate extension point.
ServiceConfigurationManager__VNFServiceDesignManager_processDisconnect	Enabled extension point for the Service Configuration specification and the VNFServiceDesignManager_processDisconnect extension point.
ServiceConfigurationManager__VNFServiceHelper_createVNF	Enabled extension point for the Service Configuration specification and the VNFServiceHelper_createVNF extension point.
ServiceConfigurationManager__VNFServiceManager_processTechnicalActions	Enabled extension point for the Service Configuration specification and the VNFServiceManager_processTechnicalActions extension point.
ServiceSpecification__ConsumerHelper_getConsumerDescriptorName	Enabled extension point for the Service specification and the ConsumerHelper_getConsumerDescriptorName extension point.

Table 7-10 (Cont.) Enabled Extension Points

Specification	Description
ServiceSpecification__ConsumerHelper_getDataCenterForConsumer	Enabled extension point for the Service specification and the ConsumerHelper_getDataCenterForConsumer extension point.
ServiceSpecification__ConsumerHelper_getDataCenterLookupIdentifier	Enabled extension point for the Service specification and the ConsumerHelper_getDataCenterLookupIdentifier extension point.
ServiceSpecification__ConsumerHelper_postNSTerminate	Enabled extension point for the Service specification and the ConsumerHelper_postNSTerminate extension point.
ServiceSpecification__NetworkService_createAndAssignIPNetworks	Enabled extension point for the Service specification and the NetworkService_createAndAssignIPNetworks extension point.
ServiceSpecification__NetworkService_createIPSubnet	Enabled extension point for the Service specification and the NetworkService_createIPSubnet extension point.

Rulesets

Table 7-11 lists the rulesets and their corresponding extension points included in the NFV Orchestration Base Specifications cartridge.

Table 7-11 Rulesets

Ruleset	Functionality
AutomateNetworkServiceConfig_NSOBaseRuleset AutomateNetworkServiceConfig_NSOBaseRulesetExtPt	Designs and assigns in the service configuration version of a network service during instantiation, updates, and termination.
AutomatePNFServiceConfig_NSOBaseRuleset AutomatePNFServiceConfig_NSOBaseRulesetExtPt	Designs and assigns in the service configuration version of the PNF service during instantiation, updates, and termination.
AutomateVNFCapabilityServiceConfig_NSOBaseRuleset AutomateVNFCapabilityServiceConfig_NSOBaseRulesetExtPt	Designs and assigns in the service configuration version of the VNF capability service during instantiation, updates, and termination.
AutomateVNFServiceConfig_NSOBaseRuleset AutomateVNFServiceConfig_NSOBaseRulesetExtPt	Designs and assigns in VNF service configuration versions during instantiation, updates, and termination.
CancelNetworkServiceConfig_NSOBaseRuleset CancelNetworkServiceConfig_NSOBaseRulesetExtPt	Handles rollback features on network service operations.
Cancel_VNFServiceConfigRuleset Cancel_VNFServiceConfigRulesetExtPt	Handles rollback features during VNF operations.

Table 7-11 (Cont.) Rulesets

Ruleset	Functionality
CompleteNetworkServiceConfig_NSOBaseRuleset CompleteNetworkServiceConfig_NSOBaseRulesetExtPt	Completes a network service configuration version.
CompleteVNFServiceConfig_NSOBaseRuleset CompleteVNFServiceConfig_NSOBaseRulesetExtPt	Completes a VNF service configuration version.
CreateEMS_Ruleset CreateEMS_RulesetExtPt	Validates EMS details during creation.
CreatePNF_Ruleset CreatePNF_RulesetExtPt	Validates PNF details during creation.
CreateVIM_NSOBaseRuleset CreateVIM_NSOBaseRulesetExtPt	Validates VIM details during creation.
IssueNetworkServiceConfig_NSOBaseRuleset IssueNetworkServiceConfig_NSOBaseRulesetExtPt	Issues a network service configuration, which in turn activates or deactivates the corresponding network service. The resources assigned to the configuration are created or deleted.
IssuePNFServiceConfig_NSOBaseRuleset IssuePNFServiceConfig_NSOBaseRulesetExtPt	Issues a PNF service configuration, which in turn activates or deactivates the corresponding PNF service. The resources assigned to the configuration are created or deleted.
IssueVNFCapabilityServiceConfig_NSOBaseRuleset IssueVNFCapabilityServiceConfig_NSOBaseRulesetExtPt	Issues a VNF service configuration, which in turn activates or deactivates the corresponding VNF service. The resources assigned to the configuration are created or deleted.
IssueVNFServiceConfig_NSOBaseRuleset IssueVNFServiceConfig_NSOBaseRulesetExtPt	Calls issue on VNFServiceManager which further (de)activates the VNF service by creating/deleting/configuring cloud resources required for VNF.
UpdatePNF_Ruleset UpdatePNF_RulesetExtPt	Validates PNF details during PNF update.
UpdateEMS_Ruleset UpdateEMS_RulesetExtPt	Validates EMS details during EMS update.
UpdateVIM_NSOBaseRuleset UpdateVIM_NSOBaseRulesetExtPt	Validates VIM details during VIM update.

NFV Orchestration Base Tags Cartridge

The **OracleComms_NSO_BaseTags** cartridge includes specifications for the tags used in NFV Orchestration functionality. These tags are used by Design Studio when it realizes NFV Orchestration-related conceptual model specifications as UIM specifications. They are also used in UIM to limit searches to NFV Orchestration entities of various types.

Tags

Table 7-12 lists and describes the Tag specifications included in the NFV Orchestration Base Tags cartridge.

Table 7-12 Tags

Tag	Description
EMS	Realize this conceptual model specification as an EMS Custom Object specification
EndPoint	Realize this conceptual model specification as an EndPoint Custom Object specification.
NetworkService	Realize this conceptual model specification as a Network Service specification.
OrchestrationRequest	Realize this conceptual model specification as a Business Interaction specification for ab orchestration request.
PNF	Realize this conceptual model specification as a PNF Service or Logical Device specification.
VNF	Realize this conceptual model specification as VNF Service or Logical Device specification.