Oracle® Communications

FTP SDH Logical Resource Discovery and Assimilation Cartridge Guide

7.4.0
F91541-02
August 2024

Oracle Communications FTP SDH Logical Resource Discovery and Assimilation Cartridge Guide, 7.4.0

F91541-02

Contents

Preface

This guide describes the functionality and design of the Oracle Communications SDH Discovery cartridge and SDH UIM Integration cartridge.

Audience

This guide is intended for Network Integrity administrators who want to understand the design and evaluate the functionality of these cartridges, and for Network Integrity developers who want either to build or to extend similar cartridges.

Developers should have a good working knowledge of FTP operations, specifications, Network Integrity, UIM, and the use of Oracle Communications Design Studio for Network Integrity.

You should be familiar with the following documents included with this release:

- Oracle Communications Network Integrity Concepts
- Oracle Communications Network Integrity Developer's Guide
- Oracle Communications Network Integrity File Transfer and Parsing Guide
- Oracle Communications Network Integrity UIM Integration Cartridge Guide

You should be familiar with the following concepts and technologies:

- Synchronous Digital Hierarchy (SDH) standards and terminology
- Development and extensibility of Network Integrity cartridge

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.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve.

Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

1

SDH Discovery Cartridge

This chapter provides information on the Oracle Communications Network Integrity SDH Discovery cartridge.

Overview

This chapter provides an overview of the Oracle Communications Network Integrity SDH Discovery cartridge.

This chapter contains the following sections:

About the SDH Discovery Cartridge

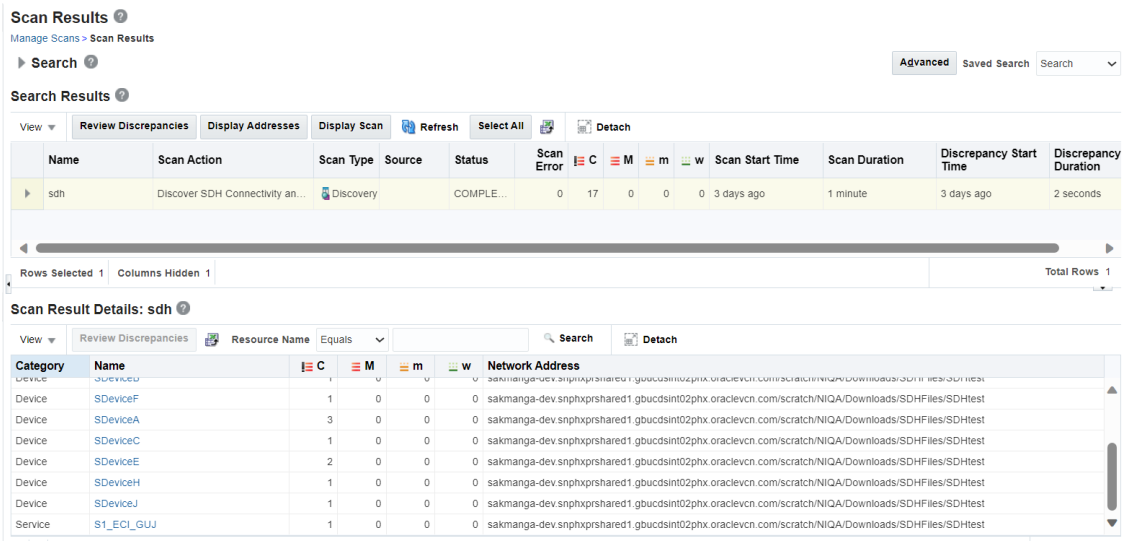
The SDH Discovery cartridge supports modeling of SDH connectivities such as Topological Link, Trail, and Tunnel. It supports modeling the service by managing these connectivities.

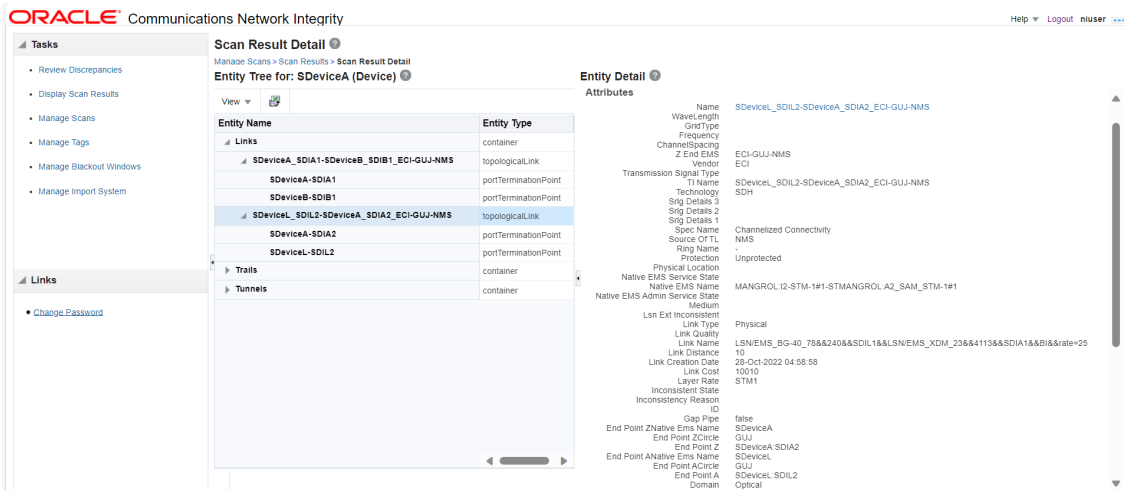
The SDH Discovery cartridge provides the following:

- Read and collect the data from files using FTP.
- Modeling and stitching of connectivities detected from the file.
- Perform Incremental Discovery on the notifications in database.

This cartridge supports logical discovery of the aforementioned entities. The logical hierarchy has a container (Links, Trails and Tunnels) for the device (connected on one of the ends) under which the connectivities are modeled along with the termination points of the connectivity. For trails and tunnels, the Primary and Secondary (if present) paths are modeled under the Trail/ Tunnel. This path upon expansion will display all the links/trails present.

For the SDH Service entity, the modeling begins at the Service level and follows with the Service Configuration ("Main Config Item"). This Configuration is further divided into multiple configurations such as EndPoint_List, Properties_List, and so on.





About Cartridge Dependencies

This section provides information on dependencies that the SDH Discovery cartridge has on other entities.

Run-Time Dependencies

This cartridge requires the **Address_Handlers** cartridge be deployed to Network Integrity.

Design-Time Dependencies

The SDH Discovery cartridge has the following dependencies:

- Address_Handlers
- NetworkIntegritySDK
- ora_uim_model
- Optical_Model
- SDH_Service_Model

- ora_ni_uim_sdh_optical
- ora_uim_network_device
- ora_ni_uim_device

Opening the Cartridge Files in Design Studio

To review and extend the SDH Discovery cartridge, you must first download the Oracle Communications SDH Discovery cartridge software from the Oracle software delivery web site:https://edelivery.oracle.com.

The software contains the SDH Discovery cartridge ZIP file, which has the following structure:

- \UIM_Cartridge_Projects\
- \Network_Integrity_Cartridge_Projects\
- SDH-Discovery-Cartridge-R7_4_0.b0.iar

See *Design Studio Online Help* and *Oracle Communications Network Integrity Developer's Guide* for information about opening files in Design Studio.

Building and Deploying the Cartridge

See *Design Studio Help* for information about building and deploying cartridges.

About the Cartridge Components

This section provides information about the components of the Oracle Communications Network Integrity SDH Discovery cartridge.

The SDH Discovery cartridge contains the following actions:

- Discover SDH Connectivity and Service

Discover SDH Connectivity and Service

The Discover SDH Connectivity and Service action collects the required input files over FTP or SFTP and processes them to model the connectivities and service.. This action creates paths that are covered by the connectivities.

Modeling is performed on various files as follows:

- Topological Link - XML file **<vendor>-<circle>-NMS-TL.xml** comprising of TL information
- Trail - Multiple CSV files and **<vendor>-<circle>-NMS-TL.xml** file
  - <vendor>-<circle>-Trail.csv : Contains the device endpoints information, layer rate, protection, and other details of the trail
  - <vendor>-<circle>-Trail_LegHasTrail_Relation.csv : Contains mapping between legs and trails
  - <vendor>-<circle>-Trail-leg.csv : Contains the leg IDs present in NMS
  - <vendor>-<circle>-Trail-ProtectionSegment.csv : Contains the protection segment ID, nature of protection, and endpoint devices information
  - <vendor>-<circle>-Trail-ProtectionSegmentHasLegsRelation.csv : Contains the mapping between legs and protection segments

  - <vendor>-<circle>-Trail-TrailMultiplexed.csv : Contains the relation between trail and other links and trails
  - <vendor>-<circle>-Trail-TrailProtectedByProtectionSegment.csv : Contains the mapping between trail and protection segment
  - <vendor>-<circle>-Trail-VC.csv : Contains the VCAT size and channel information for each TL in the trail
- Tunnel - Multiple CSV files, **<vendor>-<circle>-NMS-TL.xml** and **<vendor>-<circle>-Trail.csv** files
  - <vendor>-<circle>-Tunnel.csv : Contains the device endpoints information, layer rate, protection, and other details of the tunnel
  - <vendor>-<circle>-Tunnel-Attributes.csv : Contains the characteristics details of the tunnel
  - <vendor>-<circle>-Tunnel_LegHasTrail_Relation.csv : Contains mapping between legs and tunnels
  - <vendor>-<circle>-Tunnel-legs.csv : Contains the leg IDs present in NMS
  - <vendor>-<circle>-Tunnel-ProtectionSegment.csv : Contains the protection segment ID, nature of protection, and the endpoint devices information
  - <vendor>-<circle>-Tunnel-ProtectionSegmentHasLegsRelation.csv : Contains the mapping between legs and protection segments
  - <vendor>-<circle>-Tunnel-TrailMultiplexed.csv : Contains the relation between tunnel and other trails
  - <vendor>-<circle>-Tunnel-TrailProtectedByProtectionSegment.csv : Contains the mapping between tunnel and protection segment
- Service - Multiple CSV files
  - <vendor>-<circle>-Service.csv : Contains the service related information
  - <vendor>-<circle>-ServiceEndpoints.csv : Contains the endpoint related information of the service
  - <vendor>-<circle>-ServiceProperties.csv : Contains the properties information of the service
  - <vendor>-<circle>-ServiceSupportByTrailRelation.csv : Contains the trails and tunnels information managed by the service

Scan Parameter Group

The Discover SDH Connectivity and Service action contains the following scan parameter groups:

- SDH File Parameters
- SDH Device Parameters
- Incremental Discovery Parameters

Table 1-1 SDH File Parameters

Parameter	Default Value	Mandatory	Description
Password	N/A	No	The password to connect to the remote location.

Table 1-1 (Cont.) SDH File Parameters

Parameter	Default Value	Mandatory	Description
Port	N/A	No	The port used to connect to the remote server. The default value is **21** for FTP and **22** for SFTP.
Rename Suffix	processed	No	The suffix to add to the source file if the source file management characteristic has the value **Rename**.
Session Timeout (sec)	60	No	The time in seconds before an idle connection is timed out. The valid range is from **1** to **3600**.
Source File Management	Rename	No	Select the action to take on source files when the file transfer is complete. The available values are: **Delete**, **Rename**, and **Nothing**.
Transfer Type	SFTP	No	Select how files should be transferred: FTP, SFTP, Local.
User Name	N/A	No	The user name to connect to the remote location.

Table 1-2 SDH Device Parameters

Parameter	Default Value	Mandatory	Description
Circle Name	N/A	Yes	The circle name of the OSS.
Vendor Name	N/A	Yes	The vendor name
Topological Link Name	N/A	No	To model and persist only the Topological Links with the names given in this input field.
Topological Link	N/A	No	Select the check box to model and persist Topological Links in this scan.
Trail ID	N/A	No	To model and persist only the Trails with the IDs given in this input field.
Trail	N/A	No	Select this check box to model and persist Trails in this scan.
Tunnel ID	N/A	No	To model and persist only the Tunnels with the IDs given in this input field.
Tunnel	N/A	No	Select this check box to model and persist Tunnels in this scan.
Service ID	N/A	No	To model and persist only the Services with the IDs given in this input field.

Table 1-2 (Cont.) SDH Device Parameters

Parameter	Default Value	Mandatory	Description
Service	N/A	No	Select this check box to model and persist Services in this scan.
Device Name	N/A	No	To model and persist only those connectivities whose both A and Z termination point devices (comma separated) are provided.

Table 1-3 Incremental Discovery Parameters

Parameter	Default Value	Mandatory	Description
Incremental Discovery	N/A	No	Select this check box to perform Incremental Discovery in this scan.
Topological Link	N/A	No	Select this check box to model and persist Topological Links in this scan.
Trail	N/A	No	Select this check box to model and persist Trails in this scan.
Tunnel	N/A	No	Select this check box to model and persist Tunnels in this scan.
Service	N/A	No	Select this check box to model and persist Services in this scan.
NMS Notification Count	N/A	No	The maximum number of NMS notifications to be processed.

Conditions and Processors

The Discover SDH Connectivity and Service action contains the following conditions:

- IsTrailDiscoveryEnabled Condition
- IsTunnelDiscoveryEnabled Condition
- IsServiceDiscoveryEnabled Condition

The Discover SDH Connectivity and Service action contains the following processors run in the following order:

- Incremental Discovery Initializer Processor
- Notification Name Collector Processor
- Topological Link XML File Collector Processor
- Topological Link XML Parser Processor
- CSV Collector Processor
- Trail CSV Parser Processor

- Tunnel CSV Parser Processor
- Service CSV Parser Processor
- Update Notifications for Connectivities and Service Processor

IsTrailDiscoveryEnabled Condition

The condition checks if Trail discovery or incremental discovery is selected in the scan parameters.

IsTunnelDiscoveryEnabled Condition

The condition checks if Tunnel discovery or incremental discovery is selected in the scan parameters.

IsServiceDiscoveryEnabled Condition

The condition checks if Service discovery or incremental discovery is selected in the scan parameters.

Incremental Discovery Initializer Processor

The Incremental Discovery Initializer processor creates the required data structures for incremental discovery: **tlNames**, **trailNames**, **tunnelNames** and **serviceNames**

Notification Name Collector Processor

The Notification Name Collector processor populates the data structures prepared from the Incremental Discovery Initializer processor. If the incremental discovery option is enabled, the NMS Notification Manager fetches the required notification and populates the requisite data structure.

Topological Link XML File Collector Processor

The Topological Link XML File Collector processor collects the XML files required for Topological Link discovery and places them in the **topologicalLinkXMLFileCollectorFileCollection** collection.

Topological Link XML Parser Processor

The Topological Link XML Parser processor iterates **topologicalLinkXMLFileCollectorFileCollection** to read the XML file and model the topological link based on the data. This data is filtered according to the device names, Link names from the scan parameter group, and **tlNames** (if incremental discovery is selected). The links are grouped according to container **Links**. This continues only if the **Topological Link** option is selected in the SDH Device Parameters or Incremental Discovery scan parameter group. The TL information is captured in the **nameToTLBeanMap** collection.

CSV Collector Processor

The CSV Collector processor collects the CSV files required for Trails, Tunnels, or Services discovery and places it in the **csvCollectorFileCollection** collection.

Trail CSV Parser Processor

The Trail CSV Parser processor iterates the **csvCollectorFileCollection** collection to read the required CSV files for modeling trails. This data is filtered according to the device names, Trail IDs from the scan parameter group, and **trailNames** (if incremental discovery is selected). The

trails are grouped according to the container **Trails**. This continues only if the **Trail** option is selected in the SDH Device Parameters/Incremental Discovery scan parameter group.

Tunnel CSV Parser Processor

The Tunnel CSV Parser processor iterates the **csvCollectorFileCollection** collection to read the required CSV files for modeling tunnels. This data is filtered according to the device names, Tunnel IDs from the scan parameter group, and **tunnelNames** (if incremental discovery is selected). The tunnels are grouped according to the container **Tunnels**. This continues only if the **Tunnel** option is selected in the SDH Device Parameters or Incremental Discovery scan parameter group.

Service CSV Parser Processor

The Service CSV Parser processor iterates the **csvCollectorFileCollection** collection to read the required CSV files for modeling services. This data is filtered according to the Service IDs from the scan parameter group and **serviceNames** (if incremental discovery is selected). This continues only if the **Service** option is selected in the SDH Device Parameters or Incremental Discovery scan parameter group.

Update Notifications for Connectivities and Service Processor

The Update Notifications for Connectivities and Service processor updates the NMS notification status to **DISCOVERY_SUCCESSFUL** for entries in **tlNames**, **trailNames**, **tunnelNames**, and **serviceNames**. It also creates empty containers for devices that are not attached to any connectivties

Using the Cartridge

This section provides instructions for using the Oracle Communications Network Integrity SDH Discovery cartridge in Network Integrity.

Creating a Discover SDH Connectivity and Service Scan

The Discover SDH Connectivity and Service scan discovers and models connectivities such as Topological Links, Trails, Tunnels, and Services.

To create a Discover SDH Connectivity and Service discovery scan:

1. Create a new scan.

   See *Network Integrity Online Help* for more information.

2. On the **General** tab, do the following:

   a. From the **Scan Action** list, select **Discover SDH Connectivity and Service**. The **Scan Type** field displays **Discovery**.

   b. In the **Scan Action Parameters** section, **SDH File Parameters** is selected by default. Enter the FTP related parameters.

   c. Click on the **Scan Action Parameters** drop down and select **SDH Device Parameters**. Select the type of entities to model and the required filters such as Name, ID, Device Name, and so on.

   d. Enter the Circle Name and Vendor name.

   e. If Incremental Discovery is to be performed, select **Incremental Discovery** from the **Scan Action Parameters** drop down.

f. Select **Incremental Discovery** and the type of entities to be modeled and the number of notifications to be processed in this scan.

3. On the **Scope** tab, specify path to the file directory.
4. Save and run the scan.

About Cartridge Modeling

The Oracle Communications Network Integrity SDH Discovery cartridge models collected data according to the Oracle Communications Information Model. The collected data is modeled into the following entities:

- Pipe
- PipePipeTPRel
- PipeTerminationPoint
- TrailPath
- TrailPipeRel
- TrailPipeRelTrailPathRel
- Service
- ServiceConfigurationItem

See *Oracle Communications Information Model Reference* for more information about the Information Model.

Specifications

This section lists the specifications included in the **Optical_Model** and **SDH_Service** models.

Pipe

Table 1-4 Pipe Specifications

Specification	Cartridge
EthTunnel	Optical_Model
Optical Trail Pipe	Optical_Model
Topological Link	Optical_Model
VC3Circuit	Optical_Model
VC4Circuit	Optical_Model
VC12Circuit	Optical_Model

Pipe Termination Point

Table 1-5 Pipe Termination Point Specifications

Specification	Cartridge
Port Termination Point	Optical_Model

Service

Table 1-6 Service Specifications

Specification	Cartridge
SDH_Service	SDH_Service_Model

Service Configuration

Table 1-7 Service Configuration Specifications

Specification	Cartridge
SDH_Service_Configuration	SDH_Service_Model

About Design Studio Construction

This section provides information on the composition of the Oracle Communications Network Integrity Discovery SDH cartridge from the Oracle Communications Design Studio perspective.

Model Collections

Table 1-8 Discovery SDH Cartridge Model Collection

Specification	Information Model Entity Type	Description
Topological Link	Pipe	Represents Topological Link connectivity
Optical Trail Pipe	Pipe	Represents Optical Trail Pipe which forms the segments of an optical path
VC12Circuit	Pipe	Represents VC12 type of trail connectivity
VC3Circuit	Pipe	Represents VC3 type of trail connectivity
VC4Circuit	Pipe	Represents VC4 type of trail connectivity
EthTunnel	Pipe	Represents EthTunnel type of tunnel connectivity
Port Termination Point	Pipe Termination Point	Represents Port Termination Point which acts as a device end
SDH_Service	Service	Represents SDH_Service that manages the network
SDH_Service_Configuration	Service Configuration	Represents SDH_Service_Configuration

Discovery Action

The SDH Discovery cartridge supports the Discover SDH Connectivity and Service action.

The following table shows the discovery action in the SDH Discovery cartridge.

Table 1-9 Discover SDH Connectivity and Service

Result Category	Address Handler	Scan Parameters	Model	Processors
• Device • Link • Trail • Tunnel • Service	FileTransferAddressHandler	• SDH File Parameters • SDH Device Parameters • Incremental Discovery Parameters **Note**: You must add these scan parameters to the Create Scan web service request even if the values are left empty.	SDH_Discovery_Cartridge	• Incremental Discovery Initializer • Notification Name Collector • Topological Link XML File Collector • Topological Link XML Parser • CSV Collector • Trail CSV Parser • Tunnel CSV Parser • Service CSV Parser • Update Notifications for Connectivities and Service

Discovery Processors

The following table shows the discovery processors in the SDH Discovery cartridge.

Table 1-10 Discover SDH Connectivity and Service Action Processors

Processor Name	Variable
Incremental Discovery Initializer	Input: N/A Output: • tlNames • trailNames • tunnelNames • serviceNames Collections of IDs and names on which the incremental discovery is performed. • discoveryDeviceList Collection of device names over which discovery is performed.
Notification Name Collector	Input: • tlNames • trailNames • tunnelNames • serviceNames Collections of IDs and names on which the incremental discovery is performed. Output: N/A

Table 1-10 (Cont.) Discover SDH Connectivity and Service Action Processors

Processor Name	Variable
Topological Link XML File Collector	Input: N/A Output: • topologicalLinkXMLFileCollectorFileCollection Collection of files read from the scope provided.
Topological Link XML Parser	Input: • topologicalLinkXMLFileCollectorFileCollection Collection of files read from the scope provided. • tlNames Collection of TL names on which the incremental discovery is performed. • discoveryDeviceList Collection of device names that are obtained from the input files. Output: • nameToTLBeanMap Map with TL Name and TL Bean object to be used for processing trails and tunnels. • tlDiscoveredDevices Collection of device names which are attached to the discovered topological Links.
CSV Collector	Input: N/A Output: • csvCollectorFileCollection Collection of files read from the scope provided.
Trail CSV Parser	Input: • csvCollectorFileCollection Collection of files read from the scope provided. • nameToTLBeanMap Map with TL Name and TL Bean object to be used for processing trails and tunnels. • trailNames Collection of trail IDs on which the incremental discovery is performed. • discoveryDeviceList Collection of device names that are obtained from the input files. Output: • trailDiscoveredDevices Collection of device names which are attached to the discovered trails.

Table 1-10 (Cont.) Discover SDH Connectivity and Service Action Processors

Processor Name	Variable
Tunnel CSV Parser	Input: • csvCollectorFileCollection Collection of files read from the scope provided. • nameToTLBeanMap Map with TL Name and TL Bean object to be used for processing trails and tunnels. • tunnelNames Collection of tunnel IDs on which the incremental discovery is performed. Output: • tunnelDiscoveredDevices Collection of device names which are attached to the discovered trails.
Service CSV Parser	Input: • csvCollectorFileCollection Collection of files read from the scope provided. • serviceNames Collection of service IDs on which the incremental discovery is performed. Output: N/A
Update Notifications for Connectivities and Service	Input: • tlNames • trailNames • tunnelNames • serviceNames Collections of IDs and names on which the incremental discovery is performed. • discoveryDeviceList Collection of device names that are obtained from the input files. • tlDiscoveredDevices Collection of device names which are attached to the discovered topological Links. • trailDiscoveredDevices Collection of device names which are attached to the discovered Trails. • tunnelDiscoveredDevices Collection of device names which are attached to the discovered tunnels. Output: N/A

2

SDH UIM Integration Cartridge

This chapter provides information on the Oracle Communications Network Integrity SDH UIM Integration cartridge.

Overview

This section provides an overview of the Oracle Communications Network Integrity SDH UIM Integration cartridge.

This section contains the following sections:

- About the SDH UIM Integration cartridge
- About Cartridge Dependencies
- Opening the Cartridge Files in Design Studio
- Building and Deploying the Cartridge

About the SDH UIM Integration Cartridge

The SDH UIM Integration cartridge supports modeling of SDH connectivities such as Topological Link, Trail, and Tunnel. It also supports modeling of service managing these connectivities.

The SDH UIM Integration cartridge provides the following:

- **Import**: Retrieves the SDH connectivities, service, and circuit information from UIM and model it in the Oracle Communications Information Model.
- Perform Incremental Import along with the notifications in the database.
- **Discrepancy Detection**: Compares the imported UIM data with discovered SDH connectivities and service and reports any differences.
- **Resolution**: Resolves discrepancies on connectivities, service, and circuits by correcting entities, associations, and attributes in UIM.

About Cartridge Dependencies

This section provides information on dependencies that the SDH UIM Integration cartridge has on other entities.

Run-Time Dependencies

This cartridge requires that the Address_Handlers and SDH_Discovery cartridges to be deployed to Network Integrity. UIM must be installed and be accessible to Network Integrity.

The following components must be deployed in UIM:

- ora_ni_uim_sdh_optical
- SDH_Service_Model

- UIM Integration Web Service

Design-Time Dependencies

The SDH UIM Integration cartridge has the following dependencies:

- Optical_Model
- NetworkIntegritySDK
- SDH_Service_Model
- ora_uim_model
- UIM_Integration_Cartridge
- SDH_Discovery_Cartridge

See *Network Integrity UIM Integration Cartridge Guide* for information about the operation of the web service and its dependencies.

Opening the Cartridge Files in Design Studio

To review and extend the SDH UIM Integration cartridge, you must download the Oracle Communications SDH UIM Integration Cartridge software from the Oracle software delivery web site: https://edelivery.oracle.com

The software contains the SDH UIM Integration cartridge ZIP file, which has the following structure:

- \UIM_Cartridge_Projects\
- \Network_Integrity_Cartridge_Projects\

See *Design Studio Online Help* and *Oracle Communications Network Integrity Developer's Guide* for information about opening files in Design Studio.

Building and Deploying the Cartridge

See *Design Studio Online Help* for information about building and deploying cartridges.

About the Cartridge Components

This section provides information about the components of Oracle Communications Network Integrity SDH UIM Integration cartridge.

The SDH UIM Integration Cartridge contains the following actions:

- Abstract Optical Import from UIM
- Import SDH Connectivity and Service from UIM
- Incremental Import SDH Connectivity and Service from UIM
- Detect Discrepancies for SDH Connectivity and Service
- Resolve SDH Connectivity and Service into UIM

Abstract SDH Service Actions

The Abstract SDH Service Actions is an abstract action that interacts with UIM to fetch the devices associated with SDH connectivities and model the Service.

Scan Parameter Group

The Abstract SDH Service Actions action contains the following scan parameter groups:

- SDH Import Parameters

SDH Import Parameters

Table 2-1 SDH Import Parameters

Parameter	Default Value	Mandatory	Description
Circle Name	N/A	No	The Circle Name of the OSS.
Device Name	N/A	No	Import Devices with the names provided in this input field.
Device Name File	N/A	No	Import Devices with the names in the file path provided in this input field.
Logical Device Specification	N/A	No	Import Devices with the specification provided in this input field.
Service	N/A	No	Select this checkbox to model and persist Services in this scan.
Service ID	N/A	No	Model & Persist only the Services with the IDs given in this input field.
Vendor Name	N/A	No	The Vendor Name

Conditions and Processors of Abstract SDH Service Actions

The Abstract SDH Service Actions contains the following processors run in the following order:

1. Service Import Initializer Processor
2. Service Modeler Processor

Service Import Initializer Processor

The Service Import Initializer processor provides variable **serviceIds** of type Set and **serviceUIMImportContextList** of type List. This is performed only if the **Service** option is selected in the SDH Import Parameters or SDH Incremental Import Parameters scan parameter group.

Service Modeler Processor

The Service Modeler Processor uses **serviceIds** and **serviceUIMImportContextList** as input parameters. The services with IDs in **serviceIds** set are imported with parallel processing. This is performed only if the **Service** option is selected in the SDH Import Parameters or SDH Incremental Import Parameters scan parameter group.

Import SDH Connectivity and Service From UIM

The Import SDH Connectivity and Service From UIM action extends the Abstract Optical Import From UIM to fetch the logical device details which are used to fetch the associated connectivities and service from UIM.

Scan Parameter Group

The Import SDH Connectivity and Service From UIM action extends the Abstract SDH Service Actions. This action fetches the logical device details from the inputs which are used to fetch the associated connectivities and service from UIM. It contains the following scan parameter groups:

- SDH Import Parameters

SDH Import Parameters

Table 2-2 SDH Import Parameters

Parameter	Default Value	Mandatory	Description
Circle Name	N/A	No	The Circle Name of the OSS.
Vendor Name	N/A	No	The Vendor Name
Logical Device Specification	N/A	No	Import Devices with the specification provided in this input field.
Service ID	N/A	No	Model & Persist only the Services with the IDs given in this input field.
Service	N/A	No	Select this checkbox to model and persist Services in this scan.
Device Name	N/A	No	Import Devices with the names provided in this input field.
Device Name File	N/A	No	Import Devices with the names in the file path provided in this input field.

Conditions and Processors

The Import SDH Connectivity and Service From UIM action contains the following conditions:

- IsServiceImportSelected Condition

The Import SDH Connectivity and Service From UIM action contains the following processors run in the following order:

- SDH Import UIM Initializer Processor
- Optical Logical Device Finder Processor
- SDH Multithread TL Importer Processor
- SDH Multithread Trail Importer Processor
- SDH Multithread Tunnel Importer Processor

- Service Import Initializer Processor (inherited)
- Service Finder Processor
- Service Modeler Processor (inherited)

IsServiceImportSelected Condition

This condition checks if the Service import is selected in the scan parameters.

SDH Import UIM Initializer Processor

The SDH Import UIM Initializer Processor outputs the variable **filters** of type DeviceFilter with default values already populated and the variable **uimImportContext** of type UIMImportContext, which contains a context used by multiple processors.

Optical Logical Device Finder Processor

The Optical Logical Device Finder Processor uses **filters** and **uimImportContext** as input parameters to perform one or more Logical Device find operations through the UIM web service API. This processor provides the output through the variable **logicalDeviceIdNameMap** of type Map with the device ID as key and the device name as value, the variable **uimImportContext** of type UIMImportContext containing context used by multiple processors and also the variable **uimLogicalDeviceIDs** of type Set.

SDH Multithread TL Importer Processor

The SDH Multithread TL Importer processor uses the **logicalDeviceIdNameMap**, **uimLogicalDeviceIDs**, and **uimImportContext** as input parameters; This processor imports the Topological Links associated with the input logical devices using parallel processing.

SDH Multithread Trail Importer Processor

The SDH Multithread Trail Importer processor uses the **logicalDeviceIdNameMap**, **uimLogicalDeviceIDs**, and **uimImportContext** as input parameters; This processor imports the Trails associated with the input logical devices using parallel processing.

SDH Multithread Tunnel Importer Processor

The SDH Multithread Tunnel Importer processor uses the **logicalDeviceIdNameMap**, **uimLogicalDeviceIDs**, and **uimImportContext** as input parameters; This processor imports the Tunnels associated with the input logical devices using parallel processing.

Service Finder Processor

The Service Finder processor uses the **serviceIds**, **serviceUIMImportContextList** and **uimImportContext** as input parameters. This processor creates the request to find Services based on the input service IDs.

Incremental Import SDH Connectivity and Service From UIM

The Incremental Import SDH Connectivity and Service From UIM action extends the Abstract Optical Import From UIM and performs Incremental Import for connectivities and service from UIM.

Scan Parameter Group

The Incremental Import SDH Connectivity and Service From UIM action contains the following scan parameter groups:

- SDH Incremental Import Parameters

SDH Incremental Import Parameters

Table 2-3 SDH Incremental Import Parameters

Parameter	Default Value	Mandatory	Description
Circle Name	N/A	No	The Circle Name of the OSS.
Vendor Name	N/A	No	The Vendor Name.
NMS Notification Count	N/A	No	The maximum number of NMS notifications to be processed.
Topological Link	N/A	No	Select this checkbox to import and model Topological Links in this scan.
Trail	N/A	No	Select this checkbox to import and model Trails in this scan.
Tunnel	N/A	No	Select this checkbox to import and model Tunnel in this scan.
Service	N/A	No	Select this checkbox to import and model Services in this scan.

Conditions and Processors

The Incremental Import SDH Connectivity and Service From UIM action contains the following conditions:

- IsTLImportEnabled Condition
- IsTrailImportEnabled Condition
- IsTunnelImportEnabled Condition
- IsServiceImportEnabled Condition

The Incremental Import SDH Connectivity and Service From UIM action contains the following processors run in the following order:

- Incremental Import Initializer Processor
- Incremental TL Names Collector Processor
- Incremental Trail Names Collector Processor
- Incremental Tunnel Names Collector Processor
- Incremental Service Names Collector Processor
- Topological Link Multi Thread Incremental Importer Processor
- Trail Multi Thread Incremental Importer Processor
- Tunnel Multi Thread Incremental Importer Processor
- Service Import Initializer Processor (inherited)
- Incremental Service Finder Processor
- Service Modeler Processor (inherited)

- Update SDH Notifications to Import Status Processor

IsTLImportEnabled Condition

The condition checks if the Topological Link import is selected in the scan parameters.

IsTrailImportEnabled Condition

The condition checks if the Trail import is selected in the scan parameters.

IsTunnelImportEnabled Condition

The condition checks if the Tunnel import is selected in the scan parameters.

IsServiceImportEnabled Condition

The condition checks if the Service import is selected in the scan parameters.

Incremental Import Initializer Processor

The Incremental Import Initializer processor creates the required data structures for incremental import: **tlNames**, **trailIds**, **tunnelIds**, and **serviceNames**.

Incremental TL Names Collector Processor

The Incremental TL Names Collector processor displays the set **tlNames** . The NMS Notification manager fetches the required TL names from the database.

Incremental Trail Names Collector Processor

The Incremental Trail Names Collector processor displays the set **trailIds** . The NMS Notification manager fetches the required Trail IDs from the database.

Incremental Tunnel Names Collector Processor

The Incremental Tunnel Names Collector processor displays the set **tunnelIds** . The NMS Notification manager fetches the required Tunnel IDs from the database.

Incremental Service Names Collector Processor

The Incremental Service Names Collector processor displays the set **serviceNames** . The NMS Notification manager fetches the required Service names from the database.

Topological Link Multi Thread Incremental Importer

The Topological Link Multi Thread Incremental Importer processor uses the **tlNames** and **uimImportContext** as input parameters; This processor performs Incremental import over the TL names using parallel processing.

Trail Multi Thread Incremental Importer

The Trail Multi Thread Incremental Importer processor uses the **trailIds** and **uimImportContext** as input parameters. This processor performs Incremental import over the Trail Ids using parallel processing.

Tunnel Multi Thread Incremental Importer

The Tunnel Multi Thread Incremental Importer processor uses the **tunnelIds** and **uimImportContext** as input parameters. This processor performs Incremental import over the Tunnel Ids using parallel processing.

Incremental Service Finder

The Incremental Service Finder processor uses the **serviceIds**, **serviceNames**, **serviceUIMImportContextList** and **uimImportContext** as input parameters. This processor creates the request to find services based on the input service IDs during Incremental Import.

Update SDH Notifications to Import Status

The Update SDH Notifications to Import Status processor updates the NMS notification status to **IMPORTED** for entries in **tlNames**, **trailIds**, **tunnelIds**, and **serviceNames.**

Detect Discrepancies for SDH Connectivity and Service

The Detect Discrepancies for SDH Connectivity and Service action detects discrepancies between discovery scan results of Discover SDH Connectivity and Service action and data imported from UIM.

This discrepancy detection action extends the Discrepancy Detection action from the NetworkIntegritySDK cartridge and inherits all its processors.

The Detect Discrepancies for SDH Connectivity and Service action contains the following processors run in the following order:

1. DD Filters Initializer
2. Discrepancy Detector (inherited)

Detect Discrepancy Filters Initializer Processor

This processor implements the following filters:

- Ignore **ID** on all entities.
- Use **Name** as the comparator across all entities to determine whether objects from discovery and import are the same.
- Ignore case sensitivity for entity relationship for port termination point.
- On a Pipe, ignore the ID, physical location, description, rerouted, partial, sequence, AEnd, ZEnd, and nativeEMSName.
- On Service and Service Configuration, ignore the ipAddress.
- Ignore partial circuits, containers, and channels

Resolve SDH Connectivity and Service in UIM

The Resolve SDH Connectivity and Service in UIM action resolves discrepancies between discovery scan results of Discover SDH Connectivity and Service action and data imported from UIM.

This discrepancy resolution action extends the Abstract Resolve in UIM action from the UIM Integration cartridge and inherits all its processors. For information about the inherited processors, see *Network Integrity UIM Integration Cartridge Guide*.

The Resolve SDHConnectivity and service in UIM action contains the following processors run in the following order:

1. UIM Resolution Framework Initializer (inherited)

2. UIM Resolution Initializer (inherited)
3. Optical Resolution Initializer
4. UIM Resolution Framework Dispatcher (inherited)

Optical Resolution Initializer Processor

This processor registers the following handlers required for connectivities and service resolution:

- PipeTerminationPointHandler
- CircuitHandler
- TrailPathHandler
- ServiceHandler
- ServiceConfigurationItemHandler

Supported Creation Scenarios in UIM

This section describes the following creation scenarios, which are supported in UIM:

- Creation of a Topological Link (Channelized Connectivity & INNI Connectivity specifications)
- Creation of a Trail
- Creation of a Tunnel
- Creation of a TrailPipe
- Creation of a PipeTerminationPoint
- Creation of a TrailPath
- Creation of a Service
- Creation of a ServiceConfigurationItem

> **Note:**
>
> If the devices are not created in UIM, the **Entity+** discrepancies would be **Ignored** state as they are created for a container. Once the devices are created in the UIM and imported in Network Integrity, the discovery scan should be performed again to enable the resolving of **Entity+** discrepancies. The order of reconciliation for the **Create** operations from NI must be as follows:
>
> 1. Topological Link
> 2. Trail
> 3. Tunnel
> 4. Service

Creation of a Topological Link (Channelized Connectivity & INNI Connectivity specifications)

When a Topological Link is discovered and when it does not exist in UIM, Network Integrity displays **Entity+** for the Topological Link.

The resolution creates the Topological Link. The Topological Link is associated with the device.

Creation of a Trail

When a Trail is discovered and when it does not exist in UIM, Network Integrity displays **Entity+** for the Trail.

The resolution creates the Trail. The Trail is associated with the device.

Creation of a Tunnel

When a Tunnel is discovered and when it does not exist in UIM, Network Integrity displays **Entity+** for the Tunnel.

The resolution creates the Tunnel. The Tunnel is associated with the device.

Creation of a TrailPipe

When a Trail or Tunnel is discovered and when it does not exist in UIM, Network Integrity displays **Entity+**.

The resolution creates the TrailPipe segment. The TrailPipe segment is associated with the Trail or Tunnel.

Creation of a PipeTerminationPoint

When a Topological Link, Trail ,or Tunnel is discovered and when it does not exist in UIM, Network Integrity displays **Entity+**.

The resolution creates the PipeTerminationPoint which also creates a DeviceInterface. ThePipeTerminationPoint is associated with the Topological Link, Trail or Tunnel and the DeviceInterface is associated with the Device.

Creation of a TrailPath

When a Trail or Tunnel is discovered and when it does not exist in UIM, Network Integrity displays **Entity+**.

The resolution creates the TrailPath. The TrailPath is associated with the Trail or Tunnel.

Creation of a Service

When a Service is discovered and when it does not exist in UIM, Network Integrity displays **Entity+** for the Service.

The resolution creates the Service and its child objects. As each object is created, it is associated to the Service.

Creation of a ServiceConfigurationItem

When a ServiceConfigurationItem is discovered and when it does not exist in UIM, Network Integrity displays **Entity+**.

The resolution creates **ServiceConfigurationItem** and its child objects. **ServiceConfigurationItem** is associated to the service.

Teardown, Deletion, and Removal Scenarios in UIM

This section describes the following teardown, deletion, and removal scenarios, which are supported in UIM:

- Deletion of a Topological Link
- Deletion of a Trail
- Deletion of a Tunnel
- Removal of a Connectivity Design Path from Connectivity
- Deletion of a PipeSegment
- Deletion of a TrailPath
- Deletion of a Service
- Deletion of a ServiceConfigurationItem
- Removal of a ServiceConfigurationItem from Service

> **Note:**
>
> The order of reconciliation for the **Delete** operations from NI must be as follows:
>
> 1. Service
> 2. Tunnel
> 3. Trail
> 4. Topological Link

Deletion of a Topological Link

When a Topological Link exists in UIM but is not discovered, Network Integrity displays **Entity-**.

If the Topological Link has only one design version and is in **INPROGRESS** state, the Topological Link is deleted upon resolution.

Deletion of a Trail

When a Trail exists in UIM but is not discovered, Network Integrity displays **Entity-**.

If the Trail has only one design version and is in **INPROGRESS** state, the Trail is deleted upon resolution.

Deletion of a Tunnel

When a Tunnel exists in UIM but is not discovered, Network Integrity displays **Entity-**.

If the Tunnel has only one design version and is in **INPROGRESS** state, the Tunnel is deleted upon resolution.

Removal of a Connectivity Design Path from Connectivity

When a Topological Link, Trail, or Tunnel exists in UIM but is not discovered, Network Integrity displays **Entity-**.

The resolution of **Entity-** removes the connectivity design path from the connectivity.

Deletion of a PipeSegment

When a Topological Link, Trail, or Tunnel exists in UIM but is not discovered, Network Integrity displays **Entity-**.

The resolution of **Entity-** deletes the **PipeSegment** from the connectivity.

Deletion of a TrailPath

When a Trail or Tunnel exists in UIM but is not discovered, Network Integrity displays **Entity-**.

The resolution of **Entity-** deletes the **TrailPath** from the Trail or Tunnel.

Deletion of a Service

When a Service exists in UIM but is not discovered, Network Integrity displays **Entity-**.

The resolution of **Entity-** deletes the Service.

Deletion of a ServiceConfigurationItem

When a ServiceConfigurationItem exists in UIM but is not discovered, Network Integrity displays **Entity-**.

The resolution of **Entity-** deletes the ServiceConfigurationItem.

Removal of a ServiceConfigurationItem from Service

When a ServiceConfigurationItem exists in UIM but is not discovered, Network Integrity displays **Entity-**.

The resolution of **Entity-** removes the ServiceConfigurationItem from the Service.

Mismatched Data Scenarios

This section describes the mismatched data scenarios:

- Mismatch of Topological Link data
- Mismatch of Trail data
- Mismatch of Tunnel data
- Mismatch of channel data in a TrailPipe
- Mismatch of Service data
- Mismatch of ServiceConfigurationItem data

Mismatch of Topological Link data

When the Topological Link data in UIM has data that does not match the discovered data, Network Integrity displays **Mismatch**.

The resolution updates the mismatched Topological Link data attribute in UIM and sets the value in UIM to the discovered value.

Mismatch of Trail data

When the Trail data in UIM has data that does not match the discovered data, Network Integrity displays **Mismatch**.

The resolution updates the mismatched Trail data attribute in UIM and sets the value in UIM to the discovered value.

Mismatch of Tunnel data

When the Tunnel data in UIM has data that does not match the discovered data, Network Integrity displays **Mismatch**.

The resolution updates the mismatched Tunnel data attribute in UIM and sets the value in UIM to the discovered value.

Mismatch of channel data in a TrailPipe

When the channel data in a TrailPipe in UIM is different from the one in Network Integrity, Network Integrity displays **Mismatch**.

The resolution updates the channel data in UIM and sets the value in UIM to the discovered value.

Mismatch of Service data

When the Service data in UIM has data that does not match the discovered data, Network Integrity displays **Mismatch**.

The resolution updates the mismatched Service data attribute in UIM and sets the value in UIM to the discovered value.

Mismatch of ServiceConfigurationItem data

When the ServiceConfigurationItem data in UIM has data that does not match the discovered data, Network Integrity displays **Mismatch**.

The resolution updates the mismatched ServiceConfigurationItem data attribute in UIM and sets the value in UIM to the discovered value.

Groom Support for SDH

Grooming a connectivity is changing the design path of a connectivity by either changing segment's channel within a facility or to a channel from different facility. For example, when you groom a SDH Trail connectivity, you change the VC12 channel within same STM16 facility or to a VC12 channel from different STM4/STM16 facility.

The groom feature can also be used for:

- Reallocating bandwidth to ensure optimal usage
- Consolidating traffic to make better use of high-capacity links
- Upgrading equipment with newer, more capable devices

Grooming frequently happens due to network modifications that introduce new routes or render existing ones less effective. For example, the addition of new devices to a network might cause existing routes to have unacceptable number of hops. Planning and management procedures identify such scenarios and recommend more efficient routing adjustments through grooming. For more information, refer to "Maintaining Channelized Connectivity and Network Resources" in *UIM Concepts Guide*.

UIM exposes a Groom API via the REST protocol. Network Integrity identifies design path discrepancies for each connectivity rider, triggering a groom request and calling UIM's Groom endpoint. Upon receiving the groom request, UIM promptly acknowledges it with a 202

response and processes it in the background. Network Integrity then regularly checks UIM for a Groom response and addresses the discrepancies accordingly.

For more information on how to invoke Groom REST APIs, see *REST API for Unified Inventory Management*

A default API is introduced within the DiscrepancyHandler interface of the UIM_Integration_Cartridge, which takes collections of discrepancies (such as attribute mismatches, missing or extra pipe segments) and the reference pipe entity (either trails or tunnels) as inputs, as shown below. Reconciliation handler classes can be used to override this API.

For SDH or Ethernet networks, groom can be used on trail or tunnel connectivities by modifying the necessary segments, or changing channels within existing segments.

```
public default void handleDiscrepancyGroom(Collection<DisDiscrepancy> missingEntity,
Collection<DisDiscrepancy> extrEntity, Collection<DisDiscrepancy> attributeMismatch,
Pipe pipe) {
    }
```

The input to the API is passed from the oracle.communications.integrity.uim.resolutionprocessors.base.BaseResolutionElement class within the UIM_Integration_Cartridge. This class has the logic to group discrepancies respective to the attribute mismatches, missing or extra entities, and reference root entities for each result group.

The oracle.communications.integrity.sdhuim.resolutionprocessors.opticalresolutioninitializer.CircuitHandler OOB reconciliation handler has the logic to Groom SDH connectivities.

Rehome Support for SDH

When you rehome a connectivity, you alter one of its endpoints. This action may be necessary for load balancing purposes or due to the removal or replacement of devices and interfaces. For more information, see "Maintaining Channelized Connectivity and Network Resources" in *UIM Concepts Guide*.

The groom feature can also be used for:

- Moving customer connections to different network nodes to improve service quality
- Physical moving network devices to different location
- Switching network services from one provider to another
- Shifting network resources, such as IP addresses or bandwidth, to different parts of the network.

Rehoming a facility requires changes to the termination of the facility itself and to any channels it provides. Channels are re-terminated on sub-device interfaces provided by the new device interface on which the facility is terminated.

UIM exposes a Rehome API over REST protocol. Network Integrity detects such port change discrepancies for each connectivity facility and generates a Rehome request and invokes Rehome endpoint of UIM. Upon receiving the rehome request, UIM immediately acknowledges with a 202 response and processes it in the background. Network Integrity then regularly checks UIM for a Rehome response and addresses the discrepancies accordingly.

For more information on how to invoke Rehome REST APIs, see *REST API for Unified Inventory Management*.

A default API is introduced within the DiscrepancyHandler interface of the UIM_Integration_Cartridge, which takes a collection of discrepancies (such as missing or extra pipe or PTP entities) as input, as shown below. You can use reconciliation handler classes to override this API.

For SDH or Ethernet networks, rehome can be used on topological link, trail or tunnel pipes by modifying device changes or termination point changes on one side.

```
public default void handleDiscrepancyRehome(DisDiscrepancy missingEntity, DisDiscrepancy
extrEntity){
    }
```

The input to the API is passed from the oracle.communications.integrity.uim.resolutionprocessors.base.BaseResolutionElement class within the UIM_Integration_Cartridge. This class has the logic to group discrepancies respective to the attribute mismatches, missing or extra entities, and reference root entities for each result group.

The following OOB reconciliation handlers have the logic to rehome SDH connectivities:

- oracle.communications.integrity.sdhuim.resolutionprocessors.opticalresolutioninitializer.CircuitHandler – When a pipe entity's identifier changes due to change on one side of the device or termination point, a discrepancy is generated on the pipe entity. This handle is invoked for such discrepancies.
- oracle.communications.integrity.sdhuim.resolutionprocessors.opticalresolutioninitializer.PipeTerminationPointHandler – When a pipe entity's identifier stays unique even when one side of the device or termination point changes, then discrepancies are recorded at the Pipe's termination point. This handler is invoked for such discrepancies.

Using the Cartridge

This section provides instructions for using the Oracle Communications Network Integrity SDH UIM Integration cartridge in Network Integrity.

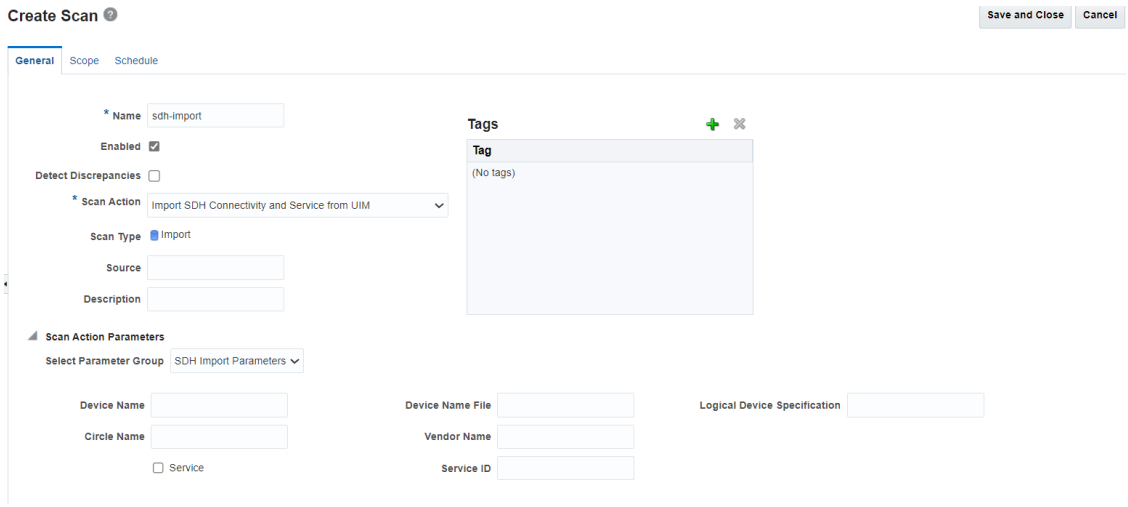
Creating an Import SDH Connectivity and Service From UIM Scan

The Import SDH Connectivity and Service From UIM scan is used to import and model connectivities such as Topological Links, Trails, Tunnels, and Services.

To create an Import SDH Connectivity and Service From UIM scan:

1. Create a new scan.
   See *Network Integrity Online Help* for more information.
2. On the **General** tab, do the following:
   - From the **Scan Action** list, select **Import SDH Connectivity and Service From UIM**. The Scan Type field displays Import
   - In the **Scan Action Parameters** section, SDH Import Parameters is selected by default.
   - (Optional) Enter the required parameters of the logical device and service parameters.
3. Save and run the scan.

Figure 2-1 Creation of an Import SDH Connectivity and Service from UIM scan



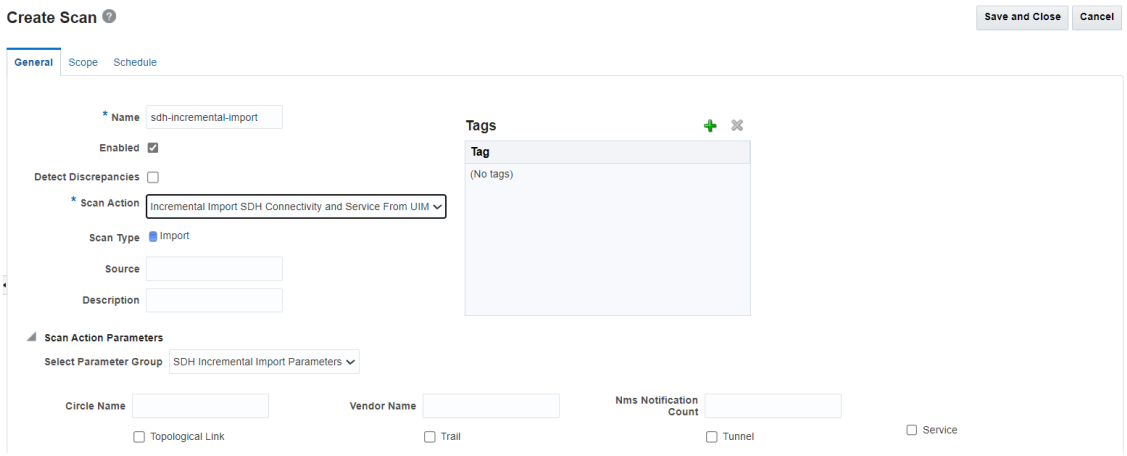
Creating an Incremental Import SDH Connectivity and Service From UIM Scan

The Incremental Import SDH Connectivity and Service From UIM scan is used to import and model connectivities such as Topological Links, Trails, Tunnels, and Services using NMS notifications.

To create an Incremental Import SDH Connectivity and Service From UIM scan:

1. Create a new scan.
   See *Network Integrity Online Help* for more information.
2. On the **General** tab, do the following:
   - From the **Scan Action** list, select **Incremental Import SDH Connectivity and Service From UIM**.
     The Scan Type field displays Import
   - In the **Scan Action Parameters** section, **SDH Incremental Import Parameters** is selected by default.
   - Enter the required parameters.
3. Save and run the scan.

Figure 2-2 Creation of an Incremental Import SDH Connectivity and Service from UIM scan



Populating UIM with Discovered SDH Connectivities and Services

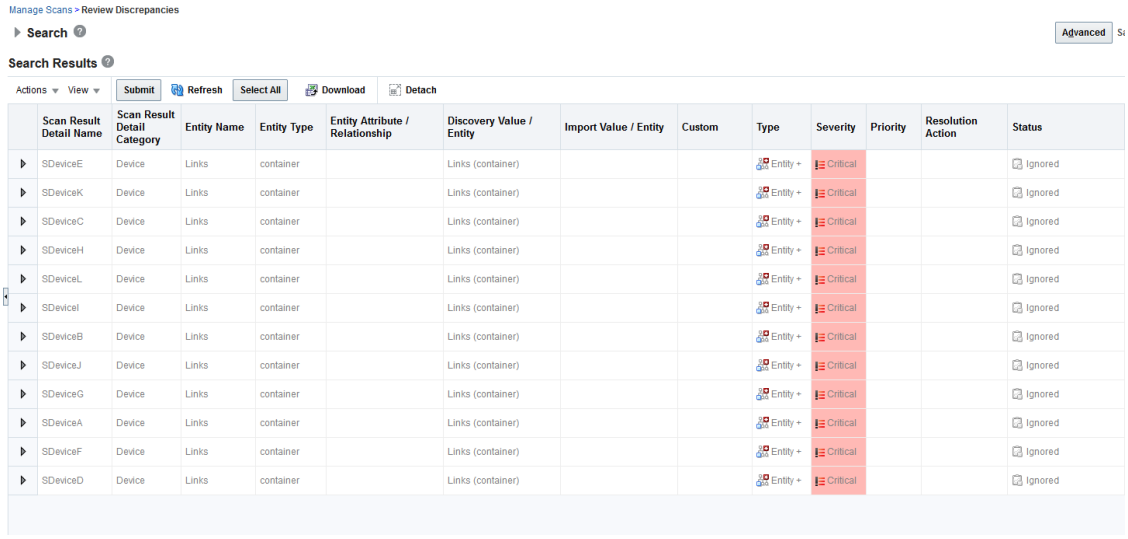
This procedure describes how to process UIM with network data discovered by the Discover SDH Connectivity and Service discovery action.

To process UIM with discovered network data:

1. Create a new scan.
   See *Network Integrity Online Help* for more information.
2. On the **General** tab, do the following:
   - From the **Scan Action** list, select **Discover SDH Connectivity and Service**.
     The Scan Type field displays Discovery.
   - Select **Detect Discrepancies**.
   - In the **Scan Action Parameters** section, **SDH File Parameters** is selected by default.
   - Enter the FTP related parameters.
   - From the **Scan Action Parameters** list, select **SDH Device Parameters**.
   - Select the type of entities to model and the required filters such as Name, ID, Device Name, and so on.
   - Enter the Circle Name and Vendor name.
   - If Incremental Discovery is to be performed, select **Incremental Discovery** from the **Scan Action Parameters** list. Select **Incremental Discovery** option and the type of entities to be modeled.
3. On the **Scope** tab, specify path to the file directory.
4. Save and run the scan.
5. Using UIM, create the required Logical Devices with **Optical Logical Device** specification from **ora_ni_uim_sdh_optical**.
6. Create a new scan.
   See **Network Integrity Online Help** for more information.
7. On the **General** tab, do the following:

- From the **Scan Action** list, select **Import SDH Connectivity and Service From UIM**. The Scan Type field displays Import
- In the **Scan Action Parameters** section, **SDH Import Parameters** is selected by default.
- Enter the required parameters of the logical device created above.

8. Save and run the scan.
   The logical devices should be present as part of the scan result.
9. Run the discovery scan again.
10. The scan generates **Entity+** discrepancies for each discovered device.
11. Right-click on the discrepancies you want to populate in UIM and select **Correct SDH Connectivity and Service in UIM**.
12. Click **Submit**.
13. Verify that UIM displays the discovered data.



Synchronizing UIM and NI with NMS Notifications of SDH Connectivities

This procedure describes steps to synchronize UIM with network data using NMS Notifications and incremental scans.

To populate UIM with discovered network data:

1. Populate the NMSNotification table in the database with the record of SDH Connectivities and set the status is set to **INITIAL**.
2. Create a new scan.

   See *Network Integrity Online Help* for more information.
3. On the General tab, do the following:
   - From the Scan Action list, select **Incremental Import SDH Connectivity and Service From UIM**.

     The Scan Type field displays *Import*.

- In the Scan Action Parameters section, **SDH Incremental Import Parameters** option is selected by default. Enter the required parameters

4. Save and run the scan.

   The connectivities present in UIM will be imported to NI and the status of the records will be changed to **IMPORTED**.

5. On the General tab, do the following:
   - From the Scan Action list, select Discover SDH Connectivity and Service.

     The Scan Type field displays Discovery.
   - Select **Detect Discrepancies**.
   - In the Scan Action Parameters section, **SDH File Parameters** option is selected by default. Enter the FTP related parameters.
   - Click on the Scan Action Parameters drop down and select **SDH Device Parameters**. Enter the Circle Name and Vendor name.
   - From the Scan Action Parameters drop down, select **Incremental Discovery**.
   - Select the **Incremental Discovery** option and the type of entities to be modelled.
6. On the Scope tab, specify path to the file directory.
7. Save and run the scan.

   The SDH connectivities present in the database are discovered as per the input files and the status of the records will be changed to **DISCOVERY_SUCCESSFULL**. The scan generates discrepancies as per the two previous scans.
8. Right-click on the discrepancies you want to populate in UIM and select **Correct SDH Connectivity and Service in UIM**.
9. Click **Submit**.
10. Verify that UIM is populated with the required data.

About Cartridge Modelling

The Oracle Communications Network Integrity SDH UIM Integration cartridge models collected data according to the Oracle Communications Information Model.

For information about the following, see *Network Integrity SDH Discovery Cartridge Guide*:

- Pipe Specifications
- Pipe Termination Point Specifications
- Service Specifications
- Service Configuration Specifications

Shared Specifications

You must first model inventory (UIM) specifications in an inventory cartridge using Design Studio, define the cartridge dependency such that the Network Integrity cartridge is dependent on the inventory cartridge, and then use the inventory cartridge specifications in the Network Integrity cartridge model.

Specifications shared with Oracle Communications Unified Inventory Management (UIM) are defined in the **SDH_Service_Model** and **ora_ni_uim_sdh_optical** cartridges. These cartridges are used to directly deploy specifications to UIM.

The following table shows the list of specifications.

Table 2-4 List of Specifications

Specification	Entity	Cartridge
TRAIL	Connectivity Function	ora_ni_uim_sdh_optical
TUNNEL	Connectivity Function	ora_ni_uim_sdh_optical
Channelized Connectivity	Connectivity	ora_ni_uim_sdh_optical
ENNI Connectivity	Connectivity	ora_ni_uim_sdh_optical
INNI Connectivity	Connectivity	ora_ni_uim_sdh_optical
Service Connectivity	Connectivity	ora_ni_uim_sdh_optical
TDM Connectivity	Connectivity	ora_ni_uim_sdh_optical
Trail	Connectivity	ora_ni_uim_sdh_optical
Tunnel	Connectivity	ora_ni_uim_sdh_optical
40GigE	Device Interface	ora_ni_uim_sdh_optical
VC12_1 to VC12_63	Device Interface	ora_ni_uim_sdh_optical
VC3_1 to VC3_40	Device Interface	ora_ni_uim_sdh_optical
VC4_1 to VC4_64	Device Interface	ora_ni_uim_sdh_optical
Ethernet Terminates Packet Rider Configuration	Flow Interface Configuration	ora_ni_uim_sdh_optical
Terminates Packet Rider Configuration	Flow Interface Configuration	ora_ni_uim_sdh_optical
Ethernet Terminates Packet Rider	Flow Interface	ora_ni_uim_sdh_optical
Terminates Packet Rider	Flow Interface	ora_ni_uim_sdh_optical
Optical Logical Device	Logical Device	ora_ni_uim_sdh_optical
SDH_Service	Service	SDH_Service_Model
SDH_Service_Configuration	Service Configuration	SDH_Service_Model

About Design Studio Construction

This section provides information on the composition of the Oracle Communications Network Integrity SDH UIM cartridge from the Oracle Communications Design Studio perspective.

Model Collections

The following table shows the SDH UIM Cartridge model collection.

Table 2-5 SDH UIM Cartridge Model Collection

Specification	Information Model Entity Type	Intended Usage/Notes
Topological Link	Pipe	Represents Topological Link connectivity
Optical Trail Pipe	Pipe	Represents Optical Trail Pipe which form the segments of an optical path
VC12Circuit	Pipe	Represents VC12 type of trail connectivity

Table 2-5 (Cont.) SDH UIM Cartridge Model Collection

Specification	Information Model Entity Type	Intended Usage/Notes
VC3Circuit	Pipe	Represents VC3 type of trail connectivity
VC4Circuit	Pipe	Represents VC4 type of trail connectivity
EthTunnel	Pipe	Represents EthTunnel type of tunnel connectivity
Port Termination Point	Pipe Termination Point	Represents Port Termination Point which acts as a device end
SDH_Service	Service	Represents SDH_Service that manages the network
SDH_Service_Configuration	Service Configuration	Represents SDH_Service_Configuration

Import Actions

SDH UIM Cartridge supports the following import actions.

Table 2-6 Abstract SDH Service Actions

Result Category	Address Handler	Scan Parameters	Model	Processors
• Device	N/A	• SDH Import Parameters	SDH_UIM_Cartridge	• Service Import Initializer • Service Modeler

Table 2-7 Import SDH Connectivity and Service From UIM

Result Category	Address Handler	Scan Parameters	Model	Processors
• Device • Link • Trail • Tunnel • Service	N/A	• SDH Import Parameters	SDH_UIM_Cartridge	• Processors inherited from the Abstract SDH Service Actions • SDH Import UIM Initializer • Optical Logical Device Finder • SDH Multithread TL Importer • SDH Multithread Trail Importer • SDH Multithread Tunnel Importer • Service Finder

Table 2-8 Incremental Import SDH Connectivity and Service From UIM

Result Category	Address Handler	Scan Parameters	Model	Processors
• Device • Link • Trail • Tunnel • Service	N/A	• SDH Incremental Import Parameters	SDH_UIM_Cartridge	• Processors inherited from the Abstract SDH Service Actions • Incremental Import Initializer • Incremental TL Names Collector • Incremental Trail Names Collector • Incremental Tunnel Names Collector • Incremental Service Names Collector • Topological Link Multi Thread Incremental Importer • Trail Multi Thread Incremental Importer • Tunnel Multi Thread Incremental Importer • Incremental Service Finder • Update SDH Notifications to Import Status

Import Processors

Table 2-9 Abstract SDH Service Actions Action Processors

Processor Name	Variable
Service Import Initializer	Input: N/A Output: • serviceIds • serviceUIMImportContextList
Service Modeler	Input: • serviceIds • serviceUIMImportContextList Output: N/A

Table 2-10 Import SDH Connectivity and Service From UIM Action Processors

Processor Name	Variable
SDH Import UIM Initializer	Input: N/A Output: • filters • uimImportContext
Optical Logical Device Finder	Input: • filters • uimImportContext Output: • uimLogicalDeviceIds • logicalDeviceIdNameMap • uimImportContext
SDH Multithread TL Importer	Input: • uimLogicalDeviceIds • logicalDeviceIdNameMap • uimImportContext Output: N/A
SDH Multithread Trail Importer	Input: • uimLogicalDeviceIds • logicalDeviceIdNameMap • uimImportContext Output: N/A
SDH Multithread Tunnel Importer	Input: • uimLogicalDeviceIds • logicalDeviceIdNameMap • uimImportContext Output: N/A
Service Finder	Input: • serviceIds • uimImportContext • serviceUIMImportContextList Output: N/A

Table 2-11 Incremental Import SDH Connectivity and Service From UIM Action Processors

Processor Name	Variable
Incremental Import Initializer	Input: N/A Output: • tlNames • trailIds • tunnelIds • serviceNames • uimImportContext
Incremental TL Names Collector	Input: • tlNames Output: N/A

Table 2-11 (Cont.) Incremental Import SDH Connectivity and Service From UIM Action Processors

Processor Name	Variable
Incremental Trail Names Collector	Input: • trailIds Output: N/A
Incremental Tunnel Names Collector	Input: • tunnelIds Output: N/A
Incremental Service Names Collector	Input: • serviceNames Output: N/A
Topological Link Multi Thread Importer	Input: • tlNames • uimImportContext Output: N/A
Trail Multi Thread Importer	Input: • trailIds • uimImportContext Output: N/A
Tunnel Multi Thread Importer	Input: • tunnelIds • uimImportContext Output: N/A
Incremental Service Finder	Input: • serviceIds • serviceNames • uimImportContext • serviceUIMImportContextList Output: N/A
Update SDH Notifications to Import Status	Input: • tlNames • trailIds • tunnelIds • serviceNames Collections of IDs and names over which incremental import is performed. Output: N/A

Discrepancy Detection Action

Detect Discrepancies for SDH Connectivity and Service is the action you use to perform Discrepancy detection.

Table 2-12 Detect Discrepancies for SDH Connectivity and Service

Result Category	Result Source	Scan Parameters	Model	Processors
All	• Discover SDH Connectivity and Service	N/A	SDH_UIM_Cartridge	• This action extends the Discrepancy Detection action • DD Filters Initializer

Table 2-13 Detect Discrepancies for SDH Connectivity and Service Action Processors

Processor Name	Variable
DD Filters Initializer	Input: N/A Output: N/A

Discrepancy Resolution Action

Resolve SDH Connectivity and Service in UIM is the action you use to perform Discrepancy Resolution.

Table 2-14 Resolve SDH Connectivity and Service in UIM

Result Category	Result Source	Processors
All	• Discover SDH Connectivity and Service	• This action extends the Abstract Resolve in UIM action included in the Network Integrity UIM Integration cartridge. See *Network Integrity UIM Integration Cartridge Guide* for more information. • Optical Resolution Initializer

Table 2-15 Resolve SDH Connectivity and Service in UIM Action Processors

Processor Name	Variable
Optical Resolution Initializer	Input: • baseResolutionElement • uimResolutionContext Output: N/A