Oracle® Communications Network Integrity

MSS Integration Cartridge Guide

Release 7.3.6

E99034-01

August 2018

ORACLE®

Oracle Communications Network Integrity MSS Integration Cartridge Guide, Release 7.3.6

E99034-01

Contents

Preface vii

Audience vii
Documentation Accessibility vii
Document Revision History viii

1 Overview

About the MSS Integration Cartridge 1-1
Limitations 1-1
About Cartridge Dependencies 1-3
Run-Time Dependencies 1-3
Design-Time Dependencies 1-3
Configuration Dependencies 1-3
Configuring the JDBC Data Source Driver 1-4
Configuring MSS as the Import System 1-5
Adding JacORB JAR Files to the Cartridge Project 1-5
Adding MSS appserver Jar Files to the Cartridge Project 1-6
Configuring Custom Ports for Reference Integration Between Network Integrity and MSS . 1-6
Configuring Connection Between Network Integrity and MSS 1-7
Setting Up Cartridge MBeans 1-8
Downloading and Opening the Cartridge Files in Design Studio 1-9
Building and Deploying the Cartridge 1-10

2 About the Cartridge Components

Import from MSS Action 2-1
Equipment DAOs Initializer 2-2
Page Initializer 2-2
Page Creator 2-2
Node Collector 2-2
Device Modeler 2-3
Equipment Hierarchy Collector 2-3
Equipment Hierarchy Modeler 2-3
Hierarchy Persister 2-3
STM Link Discoverer 2-3
VC4 Circuit Discoverer 2-3

VC3 VC12 LOP Discoverer 2-4
Detect Equipment Discrepancies Action 2-4
Equipment Filters Initializer 2-5
Discrepancy Filter 2-5
MSS Auto Resolve Selected Discrepancies 2-5
MSS Circuit Discrepancy Detection Action 2-6
Partial Circuit Discrepancy Filter 2-6
Resolve in MSS Action 2-7
MSS CORBA Property Initializer 2-7
Resolution Framework Initializer 2-8
MSS Resolution Initializer 2-8
Resolution Framework Dispatcher 2-8
About Discrepancy Detection 2-8
About Discrepancy Resolution 2-8
Extra Entity (Entity+) Discrepancy Resolution 2-9
Network Node Creation 2-9
Equipment Creation 2-10
Circuit Creation 2-10
Channel Assignment Creation on a Trail Pipe 2-11
TrailPath Assignment to a Circuit 2-11
PipeTerminationPoint Assignment to a Circuit 2-11
Missing Entity (Entity-) Discrepancy Resolution 2-12
Network Node Deletion 2-12
Equipment Deletion 2-12
Circuit Deletion 2-12
Channel Assignment Deletion on a Trail Pipe 2-12
TrailPath Unassignment from a Circuit 2-13
PipeTerminationPoint Unassignment from a Circuit 2-13
Attribute Value Mismatch (Attribute) Discrepancy Resolution 2-13
Equipment Mismatch 2-13
Circuit Channel Assignment Mismatch 2-13

3 Using the Cartridge

Creating an MSS Import Scan 3-1
Working with Discrepancies 3-2
Detecting Discrepancies in MSS 3-2
Resolving Discrepancies in MSS 3-2

4 About Collected Data

About Collected Data 4-1
About the MSS Extract Process 4-1
Advantages 4-2
Limitations 4-2
Setting Up the MSS Extract Process 4-3
Refreshing Materialized Views 4-4
MSS Equipment Extract Process 4-5
MSS Equipment Extract Process Materialized Views 4-5

MSS Equipment Extract Process Normal Views 4-13
MSS Circuit Extract Process 4-15
MSS Circuit Extract Process Materialized Views 4-15
MSS Circuit Extract Process Normal Views 4-21
Extending the MSS Extract Process 4-22

5 About Cartridge Modeling

About Cartridge Modeling 5-1
About Import Data Modeling 5-1
API Mapping 5-1
Field Mapping 5-2
Data Import Algorithm 5-9
Import Equipment Hierarchy Algorithm 5-9
Build Equipment Hierarchy Algorithm 5-10
Import Circuit Hierarchy Algorithm 5-10
About Discrepancy Resolution Modeling 5-11
Discrepancy Resolution Field Mapping for Equipment 5-12
Discrepancy Resolution Field Mapping for Circuits 5-12

6 About Design Studio Construction

Model Collections 6-1
Actions 6-1

7 About Design Studio Extension

Importing Additional Information from MSS 7-1

Preface

This guide describes the functionality and design of the Oracle Communications Network Integrity MSS Integration cartridge.

Audience

This guide is intended for network administrators who want to understand the design and functionality of this cartridge. Also, for Network Integrity integrators and developers who want either to build or to extend similar cartridges.

You should be familiar with the following documents:

- *Network Integrity Concepts*: for an overview of Network Integrity.
- *Network Integrity Developer's Guide*: for detailed information about Network Integrity cartridge components and extensibility.
- *Network Integrity Installation Guide*: for information about the cartridge deployer to deploy and undeploy cartridges to the run-time application.

This guide assumes that you are familiar with the following Oracle products and components:

- Oracle Communications Design Studio for Network Integrity
- Oracle Communications MetaSolv Solution (MSS)
- Network Integrity Optical TMF814 CORBA Cartridge
- Network Integrity Circuit Assimilation Cartridge

This guide assumes that you are familiar with the following concepts and technologies:

- TMF814 and Multi Technology Network Management (MTNM) standards and terminology
- Common object request broker architecture (CORBA) standards and terminology
- Development and extensibility of Network Integrity cartridges

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.

Document Revision History

The following table lists the revision history for this guide:

Version	Date	Description
E99034-01	August 2018	Initial release.

1

Overview

This chapter provides an overview of the Oracle Communications Network Integrity MSS Integration cartridge.

About the MSS Integration Cartridge

The MSS Integration cartridge is used to integrate Network Integrity with Oracle Communications MetaSolv Solution (MSS), to retrieve inventory data from MSS, and to compare the imported data with discovered network data.

The MSS Integration cartridge includes the following types of actions:

- Import: retrieves specified equipment and circuit information from MSS and model it in the Oracle Communications Information Model.
- Discrepancy Detection: compares imported MSS data with either discovered equipment data or assimilated circuit data and reports any differences.
- Resolution: resolves discrepancies on equipment and circuits by correcting entities, associations, and attributes in MSS.

This cartridge also includes a reference implementation of automatic discrepancy resolution. Automatic discrepancy resolution enables Network Integrity to automatically correct specific discrepancies without the user having to interact with the UI. Complete the reference implementation to specify the types of discrepancies that you want automatically resolved. In the MSS Integration cartridge, the automatic discrepancy resolution reference implementation is built with a properties file and with Java. See *Network Integrity Developer's Guide* for more information.

Limitations

The MSS Integration cartridge has the following limitations:

- Network Node Resolution: The MSS CORBA createNetworkElement API method is run by the Resolve in MSS action to create a network nodes in MSS. You must use MSS to search for the created node, manually updating it with the type and associating it with the network system, which allows later resolution actions to create equipment and hierarchies under the new node. Network Integrity cannot upload entire equipment hierarchies under a new network node with a single Resolve in MSS action. Subsequent discrepancy detection actions are likely to detect new entity+ discrepancies on the child entities of the new network node.
- Port Resolution: There is no API support to create or delete ports. The ports associated to card equipment are obtained from the equipment specification. There is no API support to update MSS equipment. Therefore, Network Integrity cannot

resolve entity+ or entity- discrepancies on ports. You must manually resolve such discrepancies from MSS.

- Equipment Resolution: When there is more than one root equipment for a logical device, the cartridge considers and resolves the first root equipment only.
- Partial Circuits: The MSS Integration cartridge cannot resolve discrepancies on partial circuits from Network Integrity. Network Integrity assigns the **Ignored** state to discrepancies on partial circuits.
- Network Integrity cannot detect discrepancies on all Information Model fields. See "Field Mapping" for a list of tables listing the fields used for discrepancy detection.
- The MSS Integration cartridge suppresses discrepancies on empty slots and sub-slots.
- Rack, shelf, and card hierarchy: MSS does not follow a consistent standard for identifying equipment types. Therefore, the MSS Integration cartridge uses a logical algorithm for modeling equipment. See "Data Import Algorithm" for more information.
- For discrepancy resolution on customer circuits to work properly, the MSS instance service type configuration must be aligned with customer circuit bandwidth. Possible customer circuit bandwidths in SDH networks are E1, E3, and E4. MSS service type configuration defines the circuit auto-build source higher bandwidth to target lower bandwidth. Network Integrity cannot define multiple service type definitions with same source higher bandwidth to different target lower bandwidths.

  Table 1–1 lists the service type configurations for each customer circuit type.

Table 1–1 Service Type Configurations for Customer Circuit Types

Circuit Type	First Level Service Type	Second Level Service Type	Third Level Service Type	Fourth Level Service Type
E1	STMX-VC4 (X=1, 4, 16)	VC4-TUG3 (Pos=3)	TUG3-VC12 (Pos=28)	VC12-E1 (Pos=1)
E3	STMX-VC4	VC4-TUG3 (Pos=3)	TUG3-VC3 (Pos=3)	VC3-E3 (Pos=1)
E4	STMX-VC4	VC4-E4 (Pos=1)	N/A	N/A

- When uploading customer circuits to MSS to resolve discrepancies, Network Integrity sets the Customer Account ID and Product Catalog ID a configurable, static value. You must use MSS to manually assign uploaded circuits with the correct Customer Account ID and Product Catalog ID. Configure the MSS Customer Account ID and MSS Product Catalog ID MBean attributes to set the static value that Network Integrity assigns to uploaded customer circuits. See "Setting Up Cartridge MBeans" for more information. The Customer Account ID and MSS Product Catalog ID values must be valid values taken from the MSS database.
- In MSS, it is possible to model a fully-protected HOT circuit two different ways:
  - As a single HOT between two devices, connected by two paths
  - As two separate unprotected HOTs between two devices

  By default, the MSS Integration cartridge matches against two separate unprotected HOTs between two devices.

  To match against fully-protected HOTs modeled as a single HOTs between two devices, connected by two paths, you can do one of the following:

  - Extend the Import from MSS action to separate protected HOT circuits into two unprotected HOT circuits. The protected HOT circuits must not have the same originating or terminating port.
  - Extend the Assimilate Optical Circuits action on the Network Integrity Optical Circuit Assimilation Cartridge, adding a processor to find separate HOTs that should be merged, modeling them as single HOTs with multiple paths.

  If you extend the assimilation or the import action, you must also extend your discrepancy resolution actions to understand the extended circuit model.

About Cartridge Dependencies

The MSS Integration cartridge has the following dependencies.

Run-Time Dependencies

For the MSS Integration cartridge to work at run time, the following dependencies must be met:

- MSS must be installed.
  - MSS must be configured with the MSS Extract Schema. The MSS database must be populated using the MSS extract process. See "Setting Up the MSS Extract Process" for more information.
- Network Integrity must be configured with a database connection to the MSS Extract Schema.
  - The data source for the MSS Extract Schema must be created in the Network Integrity WebLogic server domain.
- Network Integrity must be configured with the common object request broker architecture (CORBA) Name Service details.
- Network Integrity must be configured with Enterprise Java Bean (EJB) connection details.

Design-Time Dependencies

The MSS Integration cartridge has the following dependencies:

- Abstract_CORBA_Cartridge
- NetworkIntegritySDK
- Optical_Model
- OpticalAssimilation_Cartridge
- ora_uim_model
- TMF814_Model
- TMF814Discovery_Cartridge

Configuration Dependencies

This section describes the necessary configurations you must perform before you can use the MSS Integration cartridge.

Configuring the JDBC Data Source Driver

1. Log in to the Oracle WebLogic Server Administration Console for Network Integrity using administrator credentials.
2. Under JDBC, select **Data Sources**.

   The Summary of JDBC Data Sources screen appears.
3. Click the **New** button.

   The Create New Data Source screen appears.
4. Do the following:
   a. In the **Name** field, enter a name.
   b. In the **JNDI Name** field, enter a unique JNDI name to be used by Network Integrity. For example, **jdbc/NIMSSDatasource**.
   c. In the **Database Type** field, enter **Oracle**.
   d. In the **Database Driver** field, select **Oracle's Driver (Thin) for service connections; Versions:9.0.1,9.2.0,10,11**.
5. Click **Next**.

   The Transaction Options screen appears.
6. Do the following:
   a. Select the **Support Global Transaction** check box.
   b. Select the **Emulate Two-Phase Commit** option.
7. Click **Next**.

   The Connection Properties screen appears.
8. Do the following:
   a. In the **Database Name** field, enter the SID or service name of the database.
   b. In the **Host Name** field, enter the IP address or host name of the system on which the database running.
   c. In the **Port** field, enter the port number used to communicate with the database.
   d. In the **Database User Name** field, enter the database user name.
   e. In the **Database User Password** field, enter the database user password.
9. Click **Next**.

   The Test Database Connection screen appears.
10. Click the **Test Configuration** button.

    The console displays a success or failure message.
11. Click **Next**.
12. Select the check box corresponding to the target server.
13. Click **Finish**.

    The data source is created.

Configuring MSS as the Import System

To enable Network Integrity to import data from MSS, MSS must be configured as the import system in Network Integrity.

To set MSS as your import system:

1. In Network Integrity, in the Tasks pane, click **Manage Import Systems**.

   The Import System screen appears.

2. Click the **Create** or **Edit** icon.

   > **Note:** The **Create** icon is available only if no import system is configured. The **Edit** icon is available only if an import system is already configured.

   The Edit Import System dialog box appears.

3. Do the following:

   a. In the **Name** field, enter a name for your import system.

      For example, **MSS**.

   b. In the **Address** field, enter the unique JNDI name for the JDBC data source.

      For example, **jdbc/NIMSSDatasource**.

      See "Configuring the JDBC Data Source Driver" for more information.

   c. Click **Save and Close**.

Adding JacORB JAR Files to the Cartridge Project

The Discrepancy Resolution action uses a third-party object request broker (ORB) called JacORB to establish CORBA connectivity with MSS. The JacORB JAR files must be manually added to the **/lib** directory of the cartridge project.

To add the JacORB JAR files to the cartridge project:

1. Download version 3.9 of JacORB from the JacORB web site:

   `http://www.jacorb.org`

2. Open the JacORB ZIP file and extract the following JAR files from the **/lib** directory:

   - slf4j-api-1.7.14.jar
   - slf4j-jdk14-1.7.14.jar
   - jacorb-3.9.jar
   - jacorb-omgapi-3.9.jar

3. In Design Studio, switch to the Package Explorer view.

4. Copy the extracted JAR files to the **MSS_Cartridge/lib** cartridge project directory.

5. Add the JacORB JAR files to the cartridge project classpath:

   a. Right-click **MSS_Cartridge** and select **Properties**.

      The Properties for MSS_Cartridge dialog box appears.

   b. In the Navigation pane, click **Java Build Path**.

   c. On the **Libraries** tab, click the **Add JARs** button.

The Add JARs dialog box appears.

d. Select the new JacORB JAR files and click **Add**.

The new JacORB JAR files are added to the **JARs and class folders on the build path** list.

e. Click **OK**.

The Properties for MSS_Cartridge dialog box closes.

f. Save the project.

Adding MSS appserver Jar Files to the Cartridge Project

To add the MSS appserver Jar files to the cartridge project:

1. Extract **appserver.jar** from *MSS_Installation_Home*/**deploy**/**nur.ear!**/**APP.INF**/**lib**/**appserver.jar**, where *MSS_Installation_Home* is the directory where MSS is installed.
2. In Design Studio, switch to the Package Explorer view.
3. Copy the extracted JAR file to the **MSS_Cartridge/lib** cartridge project directory.

Configuring Custom Ports for Reference Integration Between Network Integrity and MSS

In situations where the reference integration between Network Integrity and MSS is implemented in a way that both the products communicate through a firewall that restricts specific ports, you must configure the **jacorb.properties** file in MSS and Network Integrity to open specific ports or a range of ports, which will allow network data to pass through a firewall.

The reference integration between Network Integrity and MSS uses the CORBA IIOP (Internet Inter-ORB Protocol) Specification and JacORB to establish CORBA connectivity over the network.

JacORB provides a number of socket factories to allow control over the way sockets are created on both the client side and the server side. On the server side, JacORB uses jacorb.net.server_socket_factory and jacorb.ssl.server_socket_factory to control the creation of sockets. On the client side, JacORB uses jacorb.net.socket_factory and jacorb.ssl.socket_factory to control the creation of sockets. A factory design pattern is used for the creation of sockets and server sockets.

You use the jacorb.net.socket_factory property to configure a socket factory that implements the operations defined in the interface org.jacorb.orb.factory.SocketFactory. You use the jacorb.net.server_socket_factory property to configure a server socket factory that implements the operations defined in the interface org.jacorb.orb.factory.ServerSocketFactory, as follows:

```
jacorb.net.socket_factory=org.jacorb.orb.factory.DefaultSocketFactory
jacorb.net.server_socket_factory=org.jacorb.orb.factory.DefaultServerSocketFactory
jacorb.net.socket_factory=org.jacorb.orb.factory.PortRangeSocketFactory
jacorb.net.server_socket_
factory=org.jacorb.orb.factory.PortRangeServerSocketFactory
```

You can use additional socket factories to specify the maximum and minimum port numbers in a fixed port range to enable network data to pass through a firewall, as follows:

```
jacorb.net.socket_factory.port.min
jacorb.net.socket_factory.port.max
```

```
jacorb.net.server_socket_factory.port.min
jacorb.net.server_socket_factory.port.max
```

Configuring JacORB in MSS for Communicating Through a Firewall

To enable MSS to communicate with Network Integrity through a firewall, you must configure custom ports in MSS by configuring the jacorb.properties file located in the *MSS_Home/server_name***/jacORB/etc** folder, where *MSS_Home* is the directory on the server under which the MSS software is installed and *server_name* is the name of the WebLogic Administration server.

To configure JacORB in MSS to communicate through a firewall:

1. Navigate to the *MSS_Home/server_name***/jacORB/etc** folder and open the **jacorb.properties** file.
2. Configure the following factory properties to specify the maximum and minimum port numbers in a fixed port range:

   ```
   jacorb.net.socket_factory.port.min
   jacorb.net.socket_factory.port.max
   jacorb.net.server_socket_factory.port.min
   jacorb.net.server_socket_factory.port.max
   ```

3. Save and close the **jacorb.properties** file.

Configuring JacORB in Network Integrity for Communicating Through a Firewall

To configure JacORB in Network Integrity to communicate through a firewall:

1. Copy the **jacorb.properties** file from the *MSS_Home/server_name***/jacORB/etc** folder to the **MSS_Cartridge/src** folder in Network Integrity.
2. Build and deploy the MSS Integration cartridge.

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

Configuring Connection Between Network Integrity and MSS

In order for CORBA to successfully connect to MSS, you must configure the MSS **gateway.ini** file.

To configure the MSS **gateway.ini** file:

1. Navigate to the *MSS_Home*/**server_name**/**appserver**/**gateway** folder and open the **gateway.ini** file.
2. Uncomment the following line by removing **;** at the beginning of the line:

   ```
   ;INFRASTRUCTURESERVER=MetaSolv.CORBA.WDIInfrastructure.WDIRoot,MetaSolv.WDIInfr
   astructure.WDIRootImpl"
   ```

3. Restart the MSS server.

> **Note:** If the MSS gateway.ini file is not configured, when a connection is attempted from Network Integrity to MSS, Network Integrity returns the following error message:
>
> ```
> Can¿t establish Corba connection with MSS IDL:
> omg.org/CosNaming/NamingContext/NotFound1.0"
> ```

Setting Up Cartridge MBeans

The MSS Integration cartridge uses generic Network Integrity MBeans to communicate discrepancy resolution commands with MSS. These MBeans contain property groups and properties configured with model variables. The default values are set when the cartridge is deployed. You must use Enterprise Manager to define the MBeans.

The configured MBean values are set in the MSS CORBA Properties Initializer processor during run time.

See *Network Integrity System Administrator's Guide* for information about setting MBeans using Enterprise Manager.

Table 1–2 lists the generic Network Integrity MBeans used to communicate with MSS. Set each MBean with the value required to connect the cartridge to your MSS system.

Table 1–2 Cartridge MBeans Required for Discrepancy Resolution

Attribute Name	Property Group	MBean Property Name
MSS CORBA Password	Resolve in MSS:MSS CORBA Property Initializer:MSSCORBAConnectionDetails	MSSCORBAPassword Required to establish the MSS CORBA connection for MSS equipment upload. Use the **runPropertyEncryptor.sh** script to encrypt this property. See *Network Integrity System Administrator's Guide* for more information.
MSS CORBA IOR	Resolve in MSS:MSS CORBA Property Initializer:MSSCORBAConnectionDetails	MSSCORBAIOR Required to establish the MSS CORBA connection for MSS equipment upload. When integrating Network Integrity cluster with an MSS cluster, ensure that you set up the Network Integrity CORBA MBean properties in such a way that each Network Integrity managed server in the Network Integrity cluster refers to the same IOR value of a specific MSS managed server in the MSS cluster.
MSS CORBA UserId	Resolve in MSS:MSS CORBA Property Initializer:MSSCORBAConnectionDetails	MSSCORBAUserId Required to establish the MSS CORBA connection for MSS equipment upload.
MSS EJB JNDI Name	Resolve in MSS:MSS CORBA Property Initializer:MSSEJBConnectionDetails	MSSEJBJNDIName Required to establish the MSS EJB connection for MSS circuit upload.
MSS EJB URL	Resolve in MSS:MSS CORBA Property Initializer:MSSEJBConnectionDetails	MSSEJBURL Required to establish the MSS EJB connection for MSS circuit upload.
MSS EJB UserId	Resolve in MSS:MSS CORBA Property Initializer:MSSEJBConnectionDetails	MSSEJBUserId Required to establish the MSS EJB connection for MSS circuit upload.

Table 1–2 (Cont.) Cartridge MBeans Required for Discrepancy Resolution

Attribute Name	Property Group	MBean Property Name
MSS EJB Password	Resolve in MSS:MSS CORBA Property Initializer:MSSEJBConnectionDetails	MSSEJBPassword Required to establish the MSS EJB connection for MSS circuit upload. Use the **runPropertyEncryptor.sh** script to encrypt this property. See *Network Integrity System Administrator's Guide* for more information.
MSS Customer Account ID	Resolve in MSS:MSS CORBA Property Initializer:MSSEJBConnectionDetails	MSS Customer Account Id Used to assign customer circuits created by Network Integrity to a customer account.
MSS Product Catalog ID	Resolve in MSS:MSS CORBA Property Initializer:MSSEJBConnectionDetails	MSS Product Catalog Id Used to specify the catalog reference for customer circuits created by Network Integrity.
MSS TUG3-VC12 Channel Positions Count	Resolve in MSS:MSS CORBA Property Initializer:MSSEJBConnectionDetails	mssTUG3VC12ChannelPositionsCount Used to specify the number of positions in MSS service type defined for TUG3 to VC12.
Enable Container Resolution	MSS Circuit Discrepancy Detection:Partial Circuit Discrepancy Filter:ResolutionProperties	enableContainerResolution Set to **false** and is used to not display discrepancies on container entities in Network Integrity. Network Integrity cannot resolve discrepancies on containers.
Enable STM Resolution	MSS Circuit Discrepancy Detection:Partial Circuit Discrepancy Filter:ResolutionProperties	enableSTMResolution Set to **false** and is used to not display discrepancies on STMs in Network Integrity. Network Integrity cannot resolve discrepancies on STMs.
Enable HOT Resolution	MSS Circuit Discrepancy Detection:Partial Circuit Discrepancy Filter:ResolutionProperties	enableHOTResolution Set to **false** and is used to not display discrepancies on HOTs in Network Integrity. Network Integrity cannot resolve discrepancies on HOTs.

Password properties should be encrypted using the **runPropertyEncryptor.sh** script. See *Network Integrity System Administrator's Guide* for more information about encrypting properties.

When encrypting MBean properties, you must enter the property name as it appears in the **MBean Property Name** column of Table 1–2. For example, enter **MSSCORBAPassword** when encrypting the MSS CORBA Password attribute.

Downloading and Opening the Cartridge Files in Design Studio

To open, view, and extend the MSS Integration cartridge, you must first download the cartridge ZIP file from the Oracle software delivery web site:

`https://edelivery.oracle.com`

The MSS Integration cartridge ZIP file has the following structure:

- MSS_Cartridge

- Optical_Model

The MSS_Cartridge project contains the extendable Design Studio files.

See *Network Integrity Concepts* for guidelines and best practices for extending cartridges. See *Network Integrity Developer's Guide* for information about opening files in Design Studio.

Building and Deploying the Cartridge

To build and deploy the MSS Integration cartridge, you must first add JacORB JAR files and MSS appserver.jar to the cartridge project. See "Adding JacORB JAR Files to the Cartridge Project" and "Adding MSS appserver Jar Files to the Cartridge Project" for more information.

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

2

About the Cartridge Components

This chapter provides information about the components of the Oracle Communications Network Integrity MSS Integration cartridge.

The MSS Integration cartridge contains the following actions:

- Import from MSS Action
- Detect Equipment Discrepancies Action
- MSS Circuit Discrepancy Detection Action
- Resolve in MSS Action

Import from MSS Action

The Import from MSS action connects to Oracle Communications MetaSolv Solution (MSS) to import specified inventory data. This import action writes the inventory information to materialized views and models it according to the Oracle Communications Information Model.

The Import from MSS action contains the following processors run in the following order:

1. Equipment DAOs Initializer
2. Page Initializer
3. Page Creator
4. Node Collector
5. Device Modeler
6. Equipment Hierarchy Collector
7. Equipment Hierarchy Modeler
8. Hierarchy Persister
9. STM Link Discoverer
10. VC4 Circuit Discoverer
11. VC3 VC12 LOP Discoverer

Figure 2–1 illustrates the processor workflow of the Import from MSS action.

Figure 2–1 Import from MSS Action Processor Workflow

Equipment DAOs Initializer

If **Run MSS Extract** is set to **True** in the Network Integrity UI, this processor refreshes the MSS materialized views.

Also, this processor reads the data source information from the Import System values and initializes the DAOLocator instance. The DAOLocator instance is used by other processors and actions to retrieve equipment and circuit data.

Page Initializer

This processor counts all the unique network nodes from the MSS extract views, according to the scope defined in the Network Integrity UI, and determines the number of pages needed to list all the nodes. By default, a page can contain 50 nodes. This processor produces a pageCountList iterable object.

Page Creator

This processor creates pages listing unique network node names imported from MSS matching the filtering criteria set in the Network Integrity UI. This processor outputs the node names list in a response object.

Node Collector

This processor collects all root equipment from MSS for each node on the node name list produced by the Page Creator processor. The collected root equipment are placed in the nodesMapByNodeName map, which indexes each node name and its value.

The output iterable object loops over nodeNamesSet, getting one node name per loop.

Device Modeler

This processor models each imported network node as PhysicalDevice and LogicalDevice entities and outputs a root equipment list for each modeled node.

Equipment Hierarchy Collector

This processor retrieves port information for the equipment hierarchy from EquipmentPositionHierDAO and EquipmentPortAddressDAO. This processor outputs a map listing port-to-card IDs.

Equipment Hierarchy Modeler

This processor models root equipment and its floating termination points (FTPs) from the map as physical port and media interface entities. Its associated ports are derived from the output map from the Equipment Hierarchy Collector processor and are modeled as physical port and media interface entities.

This processor builds the equipment hierarchy by parsing the equipment hierarchy string. Slots and subslots are modeled as EquipmentHolders entities, and cards as Equipment entities. This processor saves processed card IDs to an index object to avoid processing duplicate card IDs in a different hierarchy for the same parent.

Note: The equipment hierarchy string in MSS must define the equipment type for equipment for this processor to successfully build the hierarchy.

Hierarchy Persister

This processor saves the logical and physical device trees and saves the modeled hierarchy for each network node.

STM Link Discoverer

This processor discovers synchronous transport module (STM) links from the list of ports produced by the Equipment Hierarchy Modeler processor.

This processor retrieves the STM Circuit information from CircuitExportDAO and CircuitPositionDAO and models each as DisPipe entities. The STM links are modeled with a valid VC4 channel index value.

This processor outputs a list of STM links in an stmSet object.

VC4 Circuit Discoverer

This processor retrieves the VC4 circuit information from the STM Link Discoverer processor. It verifies whether the circuit is a customer circuit. Customer circuits are modeled as E4 circuits with a VC4 display string. Non-customer circuits are modeled as transport pipes with a VC4 higher order transport display string.

This processor produces a list of CircuitExport DAOs for each transport pipe.

VC3 VC12 LOP Discoverer

This processor queries the lower order pipes (LOPs) from the vc4sForLops list and models them as E3 circuits with a VC3 display string or as E1 circuits with a VC12 display string, depending on the layer rate codes.

Detect Equipment Discrepancies Action

The Detect Equipment Discrepancies action compares discovered TMF814 data with the imported MSS data and returns a list of discrepancies.

This discrepancy detection action extends Discrepancy Detector action (from the NetworkIntegritySDK cartridge) and inherits all its processors. For information about the inherited processors, see *Network Integrity Developer's Guide*.

This action also extends the Auto Resolve Discrepancies action (from the NetworkIntegritySDK cartridge) to provide automatic discrepancy resolution. For information about the processors inherited from the Auto Resolve Discrepancies action, see *Network Integrity Developer's Guide*.

The Detect Equipment Discrepancies action contains the following processors run in the following order:

1. Equipment Filters Initializer
2. Discrepancy Detector (inherited)
3. Discrepancy Filter
4. Check Auto Resolve Selected (inherited)
5. MSS Auto Resolve Selected Discrepancies
6. Identify Auto Resolving Discrepancies (inherited)
7. Prepare Resolving Discrepancies (inherited)

Figure 2–2 illustrates the processor workflow of the Detect Equipment Discrepancies action.

Figure 2–2 Detect Equipment Discrepancies Action Processor Workflow

Equipment Filters Initializer

This processor applies the equipment filters set in the Network Integrity UI on the Discrepancy Detection process. This processor also automatically filters out discrepancies that are not relevant to MSS equipment.

Discrepancy Filter

This processor collects the discrepancies generated by the Discrepancy Detector processor and sets the priority and status for discrepancies on physical ports.

Discrepancies on physical ports are labeled **Manually correct in MSS**, and Network Integrity sets the status to **Ignored** because Network Integrity cannot resolve this type of discrepancy.

MSS Auto Resolve Selected Discrepancies

This processor contains the Java class for the automatic discrepancy resolution manager. The Java implementation class determines the types of discrepancies to be automatically resolved and the resolution logic.

The MSS Integration cartridge also implements automatic discrepancy resolution with a properties file.

See *Network Integrity Developer's Guide* for more information about automatic discrepancy resolution.

MSS Circuit Discrepancy Detection Action

The MSS Circuit Discrepancy Detection action compares the results of an Assimilate Optical Circuits scan with the imported MSS data and returns a list of discrepancies. For more information about the Assimilate Optical Circuits scan action type, see *Network Integrity Optical Circuit Assimilation Cartridge Guide*.

This discrepancy detection action inherits all the processors from the Abstract Optical Circuit Discrepancy Detection action (from the Optical Circuit Assimilation cartridge). For information about the inherited processors, see *Network Integrity Optical Circuit Assimilation Cartridge Guide*.

The MSS Circuit Discrepancy Detection action contains the following processors run in the following order:

1. Circuit Discrepancy Name Filter Initializer (inherited)
2. Missing Entity Filter Initializer (inherited)
3. Partial Circuit Discrepancy Filter
4. Discrepancy Detector (inherited)

Figure 2–3 illustrates the processor workflow of the MSS Circuit Discrepancy Detection action.

Figure 2–3 MSS Circuit Discrepancy Detection Action Processor Workflow

Partial Circuit Discrepancy Filter

This processor collects the discrepancies generated by the Missing Entity Filter initializer processor.

Discrepancies on partial pipe entities with a name that begins with GENERATED_ are labeled **Manually correct in MSS**, and Network Integrity sets the status to **Ignored** because Network Integrity cannot resolve this type of discrepancy.

Resolve in MSS Action

The Resolve in MSS action resolves discrepancies between your network data and the imported data by updating equipment and circuit hierarchy in MSS.

This discrepancy resolution action inherits all the processors from the Resolve Abstract CORBA action (from the CORBA cartridge) and inherits all its processors. For information about the inherited processors, see *Network Integrity CORBA Cartridge Guide*.

The Resolve in MSS action contains the following processors run in the following order:

1. CORBA Property Initializer (inherited)
2. MSS CORBA Property Initializer
3. CORBA Connection Manager (inherited)
4. Resolution Framework Initializer
5. MSS Resolution Initializer
6. Resolution Framework Dispatcher

Figure 2–4 illustrates the processor workflow of the Resolve in MSS action.

Figure 2–4 Resolve in MSS Action Processor Workflow

MSS CORBA Property Initializer

This processor sets the common object request broker architecture (CORBA) object request broker (ORB) properties in the JacORB to establish CORBA connectivity with MSS.

Resolution Framework Initializer

This processor initializes the BaseResolutionElement resolution framework class used to register the handlers required to resolve discrepancies in MSS.

MSS Resolution Initializer

This processor registers the following entity handlers to the BaseResolutionElement class:

- DeviceHandler
- EquipmentHandler
- PhysicalPortHandler
- DeviceInterfaceHandler
- CircuitHandler
- PipeTerminationPointHandler
- TrailPathHandler

Resolution Framework Dispatcher

This processor runs the BaseResolutionElement class to evaluate and treat discrepancies using the appropriate registered entity handlers.

About Discrepancy Detection

The MSS Integration cartridge extends the NetworkIntegritySDK cartridge project to detect discrepancies.

Table 2–1 lists the possible discrepancies that can be reported and the types of entities that each discrepancy can be found on.

Table 2–1 Discrepancy Types

Discrepancy Type	Entity Types
Extra Entity (Entity+)	Physical device, equipment, equipment holder, physical port, pipe (STM/HOT/LOP), pipe termination point, trail path, trail pipe
Missing Entity (Entity-)	Physical device, equipment, equipment holder, physical port, pipe (STM/HOT/LOP), pipe termination point, trail path, trail pipe
Attribute Value Mismatch (Attribute)	Physical device, logical device, equipment, equipment holder, pipe

For more information about discrepancy detection and automatic discrepancy resolution, see *Network Integrity Developer's Guide*.

About Discrepancy Resolution

The MSS Integration cartridge has two distinct discrepancy resolution actions: one for circuits and another for equipment. The cartridge communicates equipment resolution using a CORBA connection and communicates circuit resolution using an Enterprise JavaBean (EJB) connection.

This section lists the discrepancy types that the MSS Integration cartridge can resolve from Network Integrity. All other discrepancy types must be resolved manually in MSS.

This action automatically uses the correct handler depending on the type of discrepancy being resolved. Table 2–2 lists the discrepancy types and the handler used to resolve the discrepancy.

Table 2–2 Discrepancy Resolution Handlers

Handler	Handled Entity Types	Discrepancy Type
DeviceHandler	Physical device	Entity+
EquipmentHandler	Equipment, equipment holder	Entity+, Attribute value mismatch
PipeTerminationPointHandler	Pipe termination point	Entity+, Entity-
TrailPathHandler	Trail path	Entity+, Entity-
CircuitHandler	Pipe (customer circuit)	Entity+ (LOP, TrailPipe upload), Entity- (LOP, TrailPipe delete), Attribute value mismatch (for timeslot)
PhysicalPortHandler	Physical port	Entity+, Entity-

Each handler runs creation and removal operations to fully resolve discrepancies. For discrepancies on MSS equipment, the handlers run CORBA API methods and populate Java classes with the resolution information.

Network Integrity updates the status of discrepancies as they are being resolved:

- Processed: Network Integrity successfully processed the discrepancy.
- Failed: Network Integrity could not successfully process the discrepancy. The **reasonForFailure** field explains the cause of the failure. Network Integrity logs exceptions and failure reasons.
- Ignored: Network Integrity does not support making this resolution in MSS. You must manually resolve this discrepancy in MSS.
- Not Implemented: Network Integrity could not upload the resolution to MSS. You can manually resolve this discrepancy in MSS, or extend or develop a handler to resolve the discrepancy from Network Integrity.

For more information about discrepancy resolution, see *Network Integrity Developer's Guide*.

Extra Entity (Entity+) Discrepancy Resolution

Entity+ discrepancies occur when an entity exists in your network but is missing from the imported data. Network Integrity resolves this type of discrepancy by creating the missing entity in MSS.

Network Node Creation

Entity+ discrepancies occur on physical or logical devices when the corresponding network node does not exist in MSS. The MSS Integration cartridge resolves this discrepancy by doing the following:

- Queries location_id using the MSS CORBA getLocationI API method. This method belongs to NetworkLocationSubSession of the InfrastructureSession interface.

- Creates a network node in MSS in the location returned by the getLocation API by running the MSS CORBA createNetworkElement API method. This method belongs to NetworkElementSubSession of the EquipmentSession interface.
- You must open the MSS UI and search for the created node, updating it with the type and manually associating it with the network system, which allows later resolution actions to create equipment and hierarchies under the new node.

> **Note:** Creating a network node can cause additional discrepancies on the next Discrepancy Detection action, such as new Entity+ discrepancies on MSS equipment or equipment hierarchy belonging to the new network node. This is normal.

Equipment Creation

Entity+ discrepancies occur on equipment when the corresponding rack, shelf, sub-shelf, or card does not exist in MSS. The MSS Integration cartridge resolves this discrepancy by doing the following:

- Traverses the Information Model equipment hierarchy. For each equipment found, creates equipment in MSS using the MSS CORBA installEquipment API method. This method belongs to InstallationSubSession of the EquipmentSession interface.
- For each root equipment, queries for its parent and obtains network_node_id using the CORBA getNetworkElement API method. This method belongs to NetworkElementSubSession of the EquipmentSession interface.
- For each equipment, queries for EquipmentSpecification using the MSS CORBA queryEquipSpec_v2 API method and obtains equip_spec_id. This method belongs to SpecificationSubsession of the EquipmentSession interface. NativeEMSName, discoveredPartNumber, and modelName on modeled Information Model equipment are used to identify the equipment specification in MSS.

Ensure that the equipment specification for the equipment being created exists in MSS before uploading the equipment from Network Integrity. Port creation is determined by the card equipment specification.

Circuit Creation

Entity+ discrepancies occur on circuits when a circuit does not exist in MSS. You must resolve the discrepancies on equipment, HOTs, and STMs and reconcile the data again before you can upload the resolution for customer circuits. It is recommended that you limit your first discrepancy detection scan to equipment, HOTs, and STMs. Expand subsequent scans to detect all discrepancies.

The MSS Integration cartridge resolves this discrepancy by doing the following:

- Uploads customer circuits to MSS by creating new end-to-end customer connections using the EJB createNewCustomerConnection API method. This method also assigns ports and channels to the uploaded circuit if the information is available.
- If the channel information is not available, the MSS Integration cartridge builds the channel hierarchy at the given circuit position using the EJB autoBuild API method and updates the provisioning information. The method derives the circuit position based on the synchronous digital hierarchy (SDH).
- Prepares the port and channel assignment containers for the circuit based on the traced circuit information.

- Calls the EJB updateCircuit and updateProvisioningInfo API methods to pass updated circuit and provisioning information to MSS.

You can use a wrapper API to call multiple MSS methods in a single transaction.

You can also extend the circuit resolution handler to use custom logic.

STM links and higher-order transport (HOT) circuits cannot be uploaded to or corrected in MSS with API methods from Network Integrity. Discrepancies on STMs and HOTs must be corrected manually in MSS.

Because VC4 HOTs span across multiple links in a network, you should follow these guidelines while creating HOTs in MSS:

1. Create the facility connection for the VC4 rate code.
2. Using CLR/DLR design, assign the channel from the STM links to the HOT.
3. In the network system, create the connection spanning the entire VC4 HOT.
4. Associate the VC4 HOT to the network system.

Channel Assignment Creation on a Trail Pipe

Entity+ discrepancies occur on trail pipes when a circuit in MSS is missing its channel assignment. This discrepancy occurs when channel assignments were not assigned to the circuit in MSS or when a circuit is rerouted.

When dealing with unassigned channel assignments, the MSS Integration cartridge resolves this discrepancy by assigning the circuit to the channel in MSS.

You can identify a rerouted circuit when Network Integrity reports multiple Entity+ and Entity- discrepancies. By filtering on the circuit name, you can see that the discrepancies are all related.

The MSS Integration cartridge resolves rerouted circuits by creating or correcting the channel assignments on trail pipes in MSS.

TrailPath Assignment to a Circuit

Entity+ discrepancies on trail paths are resolved by assigning the trail path to a circuit and updating the provisioning information. The MSS Integration cartridge resolves this discrepancy by doing the following:

- Calls the updateCircuit method to assign the entire trail path to the customer circuit.
- Calls the updateProvisioningInfo method to pass and update the provisioning information on the customer circuit.

PipeTerminationPoint Assignment to a Circuit

Entity+ discrepancies on pipe termination points are resolved by assigning the pipe termination point to a circuit and updating the circuit with the provisioning information. The MSS Integration cartridge resolves this discrepancy by doing the following:

- Assigns the pipe termination point to the circuit using the updateCircuit method.
- Calls the updateProvisioningInfo MSS method with the PipeTerminationPoint information to update the circuit.

Missing Entity (Entity-) Discrepancy Resolution

Entity- discrepancies occur when an entity exists in the imported data and not in your network data. Network Integrity resolves this type of discrepancy by deleting the entity from MSS.

Take special care when resolving Entity- discrepancies, because there can be many underlying causes. Review the cause carefully, choosing to resolve the root cause (either from Network Integrity or manually in MSS).

Network Node Deletion

Entity- discrepancies occur on physical or logical devices when the corresponding network node does not exist in your network. The MSS Integration cartridge resolves this discrepancy by doing the following:

- Queries the network node using the MSS CORBA getNetworkElement API method. This method belongs to NetworkElementSubSession of the EquipmentSession interface. This method may not return the network node if it was deleted while resolving another discrepancy.
- Searches for root equipment. For each root equipment, traverses the hierarchy and deletes the lowest-level equipment. See "Equipment Deletion" for more information.
- Deletes the parent network node after all child equipment are deleted. Network nodes are deleted using the MSS CORBA deleteNetworkElement API method.

Equipment Deletion

Entity- discrepancies occur on equipment when the corresponding rack, shelf, sub-shelf, or card does not exist in your network. The MSS Integration cartridge resolves this discrepancy by doing the following:

- Queries the equipment using the MSS CORBA searchEquipmentInstall_v2 API method. This method belongs to InstallationSubSession of the EquipmentSession interface.
- Uninstalls the equipment from MSS using the MSS CORBA uninstallEquipment API method at the location obtained from equipment_id.

Circuit Deletion

Entity- discrepancies occur on circuits when an additional circuit exists in MSS. The MSS Integration cartridge resolves this discrepancy by doing the following:

- Locates the additional circuit in MSS and calls the MSS EJB deleteCircuit API method with a specific circuit ID or name.

Channel Assignment Deletion on a Trail Pipe

Entity- discrepancies occur on trail pipes when a trail pipe in the network is missing its channel assignment. This discrepancy occurs when a cross-connect is missing in the network or a customer circuit is down.

To resolve this discrepancy, you must repair the circuit in the network or release the circuit from MSS.

TrailPath Unassignment from a Circuit

Entity- discrepancies on trail paths are resolved by unassigning the trail path from a circuit and updating the provisioning information. The MSS Integration cartridge resolves this discrepancy by doing the following:

- Calls the updateCircuit method to unassign the entire trail path from the customer circuit.
- Calls the updateProvisioningInfo MSS method to update the circuit.

> **Note:** You must resolve Entity+ discrepancies on trail paths before resolving Entity- discrepancies on trail paths if the discrepancies are reported on the same circuit. Or, you can also resolve the Entity+ and the Entity- discrepancies at the same time and Network Integrity will fix them in the correct order.

PipeTerminationPoint Unassignment from a Circuit

Entity- discrepancies on pipe termination points are resolved by unassigning the pipe termination point from a circuit:

- Unassigns the pipe termination point from the circuit using the updateCircuit method.
- Updates the circuit using the updateProvisioningInfo MSS method.

Attribute Value Mismatch (Attribute) Discrepancy Resolution

Attribute discrepancies occur when an entity exists in the imported data and in your network data, but the attribute values in the two sets of information do not match. Network Integrity resolves this type of discrepancy by correcting the attribute values in MSS.

Equipment Mismatch

The MSS Integration cartridge resolves attribute discrepancies on equipment by running the MSS CORBA updateEquipment API method. This method belongs to NetworkElementSubSession of the EquipmentSession interface.

Circuit Channel Assignment Mismatch

This attribute discrepancy appears on trail pipes when a customer circuit is rerouted.

The MSS Integration cartridge resolves attribute discrepancies on circuits by identifying the timeslot on the circuit and using the following APIs:

- To update a circuit entity attribute, calls the MSS EJB updateCircuit() API method.
- To update a channel attribute, calls the MSS EJB updateProvisioningInfo() API method.

The MSS Integration cartridge also calls the necessary MSS APIs to unassign the circuit from its timeslot before setting the new attribute value.

3

Using the Cartridge

This chapter provides information on how to use Oracle Communications Network Integrity when the Oracle Communications Network Integrity MSS Integration cartridge is deployed to the run-time application.

Creating an MSS Import Scan

The MSS Import Scan action imports inventory data from Oracle Communications MetaSolv Solution (MSS).

You must already have created a data source in the Network Integrity WebLogic Server domain that points to the MSS extract database, and MSS must be configured as the import system. See "Configuration Dependencies" for more information.

To create an MSS Import scan:

1. Create a scan.

   See the Network Integrity Help for more information.

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

   a. From the **Scan Action** list, select **Import from MSS**.

      The **Scan Type** field displays **Import**.

   b. (Optional) To refine the scope of the imported data, do any of the following:

      – In the **Network Location** field, specify the network location. Enter either the CLLI code or the coded location format.

      – In the **Status** field, specify the status of the data to import.

      – Filter the imported nodes by name by entering one or more names (separated by commas) in the **Node Name** field and choose a value from the **Node Name Qualifier** list.

      – In the **Node ID** field, enter one or more node IDs separated by commas.

      – In the **Scope** field, specify the scope of data to be imported.

      – In the **Run MSS Extract** field, specify whether you want to run the MSS extract procedure before running the MSS Import scan.

      See Table 6–2, " Cartridge Scan Parameter Groups Design Studio Construction" for more information.

3. Make any other required configurations.

4. Save the scan.

Note: The **Scope** tab is automatically set to the MSS Extract Schema configured on the Import System screen of Network Integrity.

Working with Discrepancies

This cartridge allows you to detect and resolve discrepancies between your discovered data and your imported MSS data. When you resolve a discrepancy, the resolution is submitted to MSS by Network Integrity.

See the Network Integrity Help for information about using Network Integrity to resolve discrepancies.

See "About Discrepancy Detection" and "About Discrepancy Resolution" for information about how the MSS Integration cartridge detects and resolves discrepancies.

When the MSS Integration cartridge is deployed to your run-time application, you can use Network Integrity for:

- Detecting Discrepancies in MSS
- Resolving Discrepancies in MSS

Detecting Discrepancies in MSS

To detect discrepancies between discovered data and imported data from MSS:

1. Create a discovery scan.
2. Create an Import from MSS scan.
3. For your Import from MSS scan, select the **Detect Discrepancies** option.
4. Run the scans: first the discovery scan, then the import scan.

   The scan with **Detect Discrepancies** enabled must be run last. Discrepancy detection runs automatically after the import scan completes.

Resolving Discrepancies in MSS

To resolve discrepancies in MSS:

1. Review the scan results for a scan with **Detect Discrepancies** enabled.
2. On the Scan Details page, click **Review Discrepancies**.
3. For every discrepancy you want to resolve, right-click on the discrepancy and select **Correct in MSS**.
4. Click **Submit**.

   The MSS Integration cartridge calls the appropriate APIs to resolve the discrepancy in MSS.

4

About Collected Data

This chapter provides information about how the Oracle Communications Network Integrity MSS Integration cartridge treats collected data.

About Collected Data

The reference integration between Oracle Communications Network Integrity and Oracle Communications MetaSolv Solution (MSS) uses the MSS extract process that uses fast-refreshable, read-only materialized views to store the MSS inventory data, which is imported by the MSS Integration cartridge for discrepancy detection/resolution.

About the MSS Extract Process

The MSS extract process includes the following:

- **MSS Equipment Extract Process:** The MSS equipment extract process extracts the relevant equipment information from MSS and stores it into read-only materialized views in the MSS database, which is imported by the MSS Integration cartridge for discrepancy resolution.

  See "MSS Equipment Extract Process" for more information.

- **MSS Circuit Extract Process:** The MSS circuit extract process extracts the relevant circuit information from MSS and stores it into read-only materialized views in the MSS database, which is imported by the MSS Integration cartridge for discrepancy resolution.

  See "MSS Circuit Extract Process" for more information.

The MSS extract process enables you to do the following:

- Retrieve information about:
  - Equipment
  - Equipment custom attributes
  - Port address custom attributes
  - Circuits
  - Template-based connections
  - Service Trails for circuits and connections
  - Connection custom attributes, including allocation parameters, such as VLAN ID and VPI/VCI

- Store the retrieved equipment and circuit information into read-only materialized views within the EXTRACT schema in the MSS database, which is imported by the MSS Integration cartridge to do the following:
  - Compare the imported MSS data with either the discovered equipment data or assimilated circuit data and report any differences
  - Resolve discrepancies on equipment and circuits by using MSS APIs to correct entities, associations, and attributes in MSS

Note: A materialized view is a complete or partial copy (replica) of one or more target (master) tables.

Advantages

The MSS extract process has the following advantages:

- Improves the end-to-end performance and reliability of the integrated solution.
- Includes custom attributes (connections, equipment, port addresses, and allocation parameters) and service trails of virtual connections. The system integrator can use these attributes to extend the SDH reference integration to support other technologies and meet specific business requirements.
- Enables the system integrator to extend the MSS extract process without:
  - Defining new tables and/or columns within the MSS EXTRACT schema
  - Writing any Procedural Language (PL)/Structured Query Language (SQL) logic to update the tables
- Enables the system integrator to:
  - Update the definition of an existing materialized view to retrieve the required data
  - Define a new materialized view to store the retrieved data
- Leverages the capability of the Oracle database and its materialized view logs to keep the retrieved data in sync with the ASAP schema, instead of relying on complex user-written PL/SQL logic to update the retrieved data.

Limitations

The MSS extract process has the following limitations:

- **Equipment and Port Address Custom Attributes Resolution:** The MSS extract process supports the extraction of equipment and port address custom attributes; however, there is no MSS API support to upload this data to MSS. Therefore, you must manually resolve such discrepancies in MSS.
- **End-to-end Reconciliation of SONET/SDH circuits modeled within the traditional SONET/SDH Network Design Module:** The MSS extract process supports the extraction of the channelized connectivity that constitutes a synchronous optical networking/synchronous digital hierarchy (SONET/SDH) network built within the traditional SONET/SDH Network Design module; however, there is no API support to create or update the existing SONET/SDH network assignments and their related SONET blocks on the circuit's design layout report (DLR) in MSS. Therefore, Network Integrity must use custom logic to resolve such circuit discrepancies in MSS.

Setting Up the MSS Extract Process

Before you run the equipment/circuit extract process, you must set up the MSS extract process.

Setting up the MSS extract process involves the following steps:

- Creating a new EXTRACT schema in the MSS database.
- Creating new materialized views within the EXTRACT schema, which stores information about equipment and circuits.
- Granting appropriate privileges to the ASAP and EXTRACT user to define and use the new materialized views.

To set up the MSS extract process:

1. Download the MSS Integration cartridge ZIP file from the Oracle software delivery Web site:

   https://edelivery.oracle.com

   The MSS Integration cartridge ZIP file has the following structure:

   - MSS_Cartridge
   - Optical_Model

2. Connect to the MSS database as the ASAP user through sqlplus at the command prompt.

3. Navigate to the MSS_Cartridge/**scripts** folder and run the **mss_ni_ext_using_mviews_mstr.sql** file with database administrator privileges.

   The **mss_ni_ext_using_mviews_mstr.sql** file is the master file for the refactored MSS extract process using materialized views.

   When you run the **mss_ni_ext_using_mviews_mstr.sql** master file, the following scripts are run:

   - **extr_schema.sql**: Creates a new EXTRACT schema in the MSS database if the EXTRACT schema does not already exist.
   - **extr_log.sql**: Creates materialized view logs within the ASAP schema for the appropriate master tables that are used by the materialized views within the EXTRACT schema. The materialized view logs keep track of the changes to the data in the master tables and can be used to perform a fast refresh (incremental) for all materialized views without requiring a complete refresh every time the data in the master tables is modified.

     Note: A materialized view log is a table associated with the master table of a materialized view.

   - **extr_grants.sql**: Grants the following privileges to the ASAP user to define and use the new materialized views:
     - GRANT CREATE ANY MATERIALIZED VIEW TO ASAP;
     - GRANT CREATE TABLE TO EXTRACT;
     - GRANT GLOBAL QUERY REWRITE TO EXTRACT;
     - GRANT SELECT ON ASAP.*TABLE_NAME* TO EXTRACT;
     - GRANT SELECT ON MLOG$_*TABLE_NAME* TO EXTRACT;

where:

TABLE_NAME is the name of the master ASAP table from which the data is extracted. For example, ASAP.EQUIPMENT, ASAP.EQUIPMENT_SPEC, ASAP.CIRCUIT, ASAP.CIRCUIT_XREF, and so on.

- **extr_jklm.sql**: Adds the JKLM function to the EXTRACT schema. The JKLM function calculates JKLM values.
- **extr_PKG_VIEW_PARAMETERS.sql**: Creates the EXTRACT.PKG_VIEW_PARAMETERS package that you can use to GET/SET equipment ID and circuit design ID to retrieve data from the V_MP_HIER and V_PA_HIER hierarchy views:
  - pkg_view_parameters.set_equip_id(*e_id in number*). For example, pkg_view_parameters.set_equip_id(45332).
  - pkg_view_parameters.set_ckt_id(*c_id in number*). For example, pkg_view_parameters.set_ckt_id(1015332).

  You must set the equipment ID and circuit design ID in the EXTRACT.PKG_VIEW_PARAMETERS package before using V_MP_HIER and V_PA_HIER hierarchy views on the same transaction.
- **extr_mviews.sql**: Creates materialized views under the EXTRACT schema.

  See the following for more information:
  - MSS Equipment Extract Process Materialized Views
  - MSS Circuit Extract Process Materialized Views
- **extr_views.sql**: Creates normal views under the EXTRACT schema.

  See the following for more information:
  - MSS Equipment Extract Process Normal Views
  - MSS Circuit Extract Process Normal Views
- **extr_index.sql**: Creates indexes on materialized views.

Refreshing Materialized Views

Because the MSS data is updated constantly, you must refresh the materialized views at regular intervals to ensure that the materialized views always contain the latest data.

You can refresh the materialized views in the following ways:

- Through the Network Integrity GUI, do the following:
  - When running the MSS Import scan, select the **Run MSS Extract** check box to refresh the materialized views.

Note: The scope of the materialized views to be refreshed is governed by the option you select from the **Scope** list of the Import Scan. For example, if you select **Equipment Only** from the **Scope** list, only those materialized views that store information about MSS equipment are refreshed. If you select **Equipment and STM Links Only** from the **Scope** list, only those materialized views that store information about MSS equipment/synchronous transport module (STM) links are refreshed. If you select **Equipment, STM Links, and Circuits** from the **Scope** list, all the materialized views that store information about MSS equipment/STM links/circuits are refreshed.

- Manually call the following procedure:

  ```
  DBMS_MVIEW.REFRESH('MV_NAME','argument');
  ```

  where:

  – *MV_NAME* is the name of the materialized view

  – *argument* is one of the following:

  * **?**: Performs a fast refresh, and if fast refresh is not successful, performs a complete refresh.
  * **F**: Performs a fast refresh, and if fast refresh is not successful, the materialized view is not refreshed.
  * **C**: Performs a complete refresh.

  Note: Oracle recommends that you use the **?** argument to refresh the materialized views.

- Using Oracle Scheduler (DBMS_SCHEDULER), you can schedule jobs to run at a specified time or interval.

MSS Equipment Extract Process

The MSS equipment extract process extracts the relevant equipment information from MSS and stores it into fast-refreshable, read-only materialized views within the EXTRACT schema in the MSS database, which is imported by the MSS Integration cartridge to compare the imported MSS data with discovered network data and resolve discrepancies on equipment in MSS.

MSS Equipment Extract Process Materialized Views

The MSS equipment extract process retrieves equipment information and stores it in the following MSS materialized views:

- **EXTRACT.MV_EQUIPMENT:** Stores the attributes and defining information of an equipment instance. See Table 4–1 for more information.
- **EXTRACT.MV_MOUNTING_POSITION:** Stores the slot hierarchy and installed equipment within an equipment instance. See Table 4–2 for more information.
- **EXTRACT.MV_EQUIPMENT_SPEC:** Stores the attributes and defining information of an equipment specification. See Table 4–3 for more information.

- **EXTRACT.MV_EQUIPMENT_SPEC_MPOS:** Stores information about the number of mounting positions each equipment specification contains. See Table 4–4 for more information.
- **EXTRACT.MV_PORT_ADDRESS:** Stores the port address hierarchy and assigned circuits for an equipment instance. See Table 4–5 for more information.
- **EXTRACT.MV_NETWORK_NODE:** Stores the attributes and defining information of a network element. See Table 4–6 for more information.
- **EXTRACT.MV_NS_COMPONENT:** Stores the attributes and defining information of a network component. See Table 4–7 for more information.
- **EXTRACT.MV_NETWORK_LOCATION:** Stores the attributes and defining information of a network location. See Table 4–8 for more information.
- **EXTRACT.MV_EQUIPMENT_CA:** Stores the configurable parameters tied to an equipment instance which is stored within Custom Attributes. See Table 4–9 for more information.
- **EXTRACT.MV_PORT_ADDRESS_CA:** Stores the configurable parameters tied to a port address which is stored within Custom Attributes. See Table 4–10 for more information.
- **EXTRACT.MV_NS_COMP_EQUIP:** Stores the attributes and defining information of a network component tied to an equipment instance. See Table 4–11 for more information.

The following tables describe the contents of the MSS materialized views in which the MSS equipment extract process stores the inventory data.

Table 4–1 describes the contents of the EXTRACT.MV_EQUIPMENT materialized view.

Table 4–1 EXTRACT.MV_EQUIPMENT Materialized View

Column Name	Data Type	Description
EQUIPMENT_ID	NUMBER(9)	The unique table key.
EQUIPMENT_NAME	VARCHAR2(15)	The name of the equipment.
AVAILABILITY_STATUS	CHAR(1)	Indicates the current state of this item. Valid values are: ■ **I** = Installed ■ **S** = Spare ■ **U** = Under Construction
LOCATION_ID	NUMBER(9)	A unique identifier visible only to the system. Used to store and retrieve information about the location.
LOCATION_ID_2	NUMBER(9)	The location ID that represents the 11-byte CLLI location.
NETWORK_NODE_ID	NUMBER(9)	Used to uniquely identify a network node.
TIMING_SOURCE	VARCHAR2(15)	Identifies the origination of the timing signal for this equipment. Valid values are: ■ **External** ■ **Loop/Line** ■ **Internal**

Table 4–1 (Cont.) EXTRACT.MV_EQUIPMENT Materialized View

Column Name	Data Type	Description
VERSION_OF_HARDWARE_INSTALLED	VARCHAR2(20)	The version of the hardware equipment to be installed.
SERIAL_NBR	VARCHAR2(35)	The unique identification for a piece of equipment. Entered/modified as an attribute residing on a circuit.
EQUIPMENT_SPEC_ID	NUMBER(9)	An identifier visible only to the system. Used for storing and retrieving information about an equipment specification.
SOFTWARE_RELEASE_IDENTIFIER	VARCHAR2(10)	The current software release for an operating system. For example, a Northern Telecom DNX-100 DACS may be at NSR-5 software release.

Table 4–2 describes the contents of the EXTRACT.MV_MOUNTING_POSITION materialized view.

Table 4–2 EXTRACT.MV_MOUNTING_POSITION Materialized View

Column Name	Data Type	Description
EQUIPMENT_ID	NUMBER(9)	A unique identifier visible only to the system. Used to store and retrieve information about the equipment.
EQUIPMENT_ID_2	NUMBER(9)	Describes the current piece of equipment that is installed in equipment_id.
MOUNTING_POSITION_NUMBER	VARCHAR2(8)	Identifies the exact location of an assignable item (equipment or termination) within relay rack or multi-position equipment.
GROUP_IDENTIFIER	VARCHAR2(12)	Allows you to associate mounting positions and port addresses with complement information (for example, DIGROUP A) for a piece of equipment (for example, D4 channel bank).
MOUNTPOS_SEQ	NUMBER(5)	System generated number to uniquely identify and sequence mounting positions for an equipment specification or a piece of installed equipment.
SLOT_NODE_ADDR	VARCHAR2(30)	Used to build the node address for a given port address when the software address depends on mounting information.

Table 4–3 describes the contents of the EXTRACT.MV_EQUIPMENT_SPEC materialized view.

Table 4–3 EXTRACT.MV_EQUIPMENT_SPEC Materialized View

Column Name	Data Type	Description
EQUIPMENT_SPEC_ID	NUMBER(9)	An identifier visible only to the system. Used for storing and retrieving information about an equipment specification.
EQUIPMENT_ACRONYM	VARCHAR2(10)	Used on the connection layout record (CLR) or design layout report (DLR). It is an acronym for a material item. For example, FXS is the acronym for a Foreign Exchange Channel Unit on the subscriber's end.
VENDOR_PART_NUMBER	VARCHAR2(25)	The part number for this unit of equipment as assigned by the manufacturer. For example, 263DB2, 1011, 4420D, and so on.

Table 4–3 (Cont.) EXTRACT.MV_EQUIPMENT_SPEC Materialized View

Column Name	Data Type	Description
VENDOR_NAME	VARCHAR2(20)	The manufacturer of this unit of equipment.
EQUIPSPEC_TYPE	VARCHAR2(50)	Identifies the equipment type within an equipment category. For a category of SHELF, the type can be CHANNEL BANK or MUX. You can define the types within a category.
OCCUPIES_MOUNTING_POSITIONS	NUMBER(4)	The number of spaces or slots required in a parent piece of equipment (bay/rack/shelf) to mount this hardware.

Table 4–4 describes the contents of the EXTRACT.MV_EQUIPMENT_SPEC_MPOS materialized view.

Table 4–4 EXTRACT.MV_EQUIPMENT_SPEC_MPOS Materialized View

Column Name	Data Type	Description
EQUIPMENT_SPEC_ID	NUMBER(9)	An identifier visible only to the system. Used for storing and retrieving information about an equipment specification.
NBR_OF_MOUNT_POS	NUMBER	Stores the number of mounting positions an equipment specification contains, on which other equipment can be installed.

Table 4–5 describes the contents of the EXTRACT.MV_PORT_ADDRESS materialized view.

Table 4–5 EXTRACT.MV_PORT_ADDRESS Materialized View

Column Name	Data Type	Description
EQUIPMENT_ID	NUMBER(9)	A unique identifier used to store and retrieve information about equipment.
PORTADDR_SEQ	NUMBER(9)	System-generated number to uniquely identify and sequence port addresses for an equipment specification or a piece of installed equipment.
NODE_ADDRESS	VARCHAR2(30)	Identifies the specific port/channel addressing designation for a port address. The physical or logical address (software address) associated with this port. It may be derived from the node address of the equipment specification. The node addresses are of the following types: ■ The node addresses that remain constant irrespective of the location of the installed device. ■ The node addresses that are dependent on the hierarchy of the equipment on which they are installed. For example, a DCM card in a DCM shelf has an address that is dependent on the bay-shelf combination on which it is installed. ■ The node addresses that are entirely dependent on the slot in which they are installed. For example, a low-speed card installed on a DDM2000 shelf inherits the node address of its ancestor, A-1-1.
RATE_CODE	VARCHAR2(10)	Identifies the bit rate associated with a circuit, facility, or equipment. For example, DS0, DS1, DS3, N/A, and so on.
CIRCUIT_DESIGN_ID	NUMBER(9)	A unique identifier used for storing and retrieving information about a single circuit.

Table 4–5 (Cont.) EXTRACT.MV_PORT_ADDRESS Materialized View

Column Name	Data Type	Description
PORT_ADDR_STATUS	CHAR(1)	Describes the current status of the circuit position. Valid values are: ■ **1** = Unassigned ■ **2** = Pending installation work order ■ **3** = In service ■ **4** = Pending removal work order ■ **5** = Trouble ■ **6** = Reserved ■ **7** = Reserved capacity
PORTADDR_TYPE	CHAR(1)	Indicates whether the port address (or enabled port address) is physical or virtual. Physical ports are those that have actual wired connections and include their enabled (software) ports. Virtual ports are those that have no actual physical appearance or connection and are entirely in the software of the equipment. Valid values are: ■ **P** = Physical ■ **V** = Virtual The existing rows in the TBS database at the time of implementation default to **P**.
CIRCUIT_POSITION_NUMBER_CP	NUMBER(9)	The subposition within a mounting position. This column applies only to plug-in cards that have multi-position capabilities. For such cards, this field identifies the multiple position number of a transmission facility circuit (TFC) or a channel number within a carrier system. This number may correspond to the mounting position of the equipment used to terminate the TFC. This column on this table is a foreign key describing the circuit position that this port address enables.
CIRCUIT_DESIGN_ID_CP	NUMBER(9)	An identifier visible only to the system; Used for storing and retrieving information about a single circuit. This column is a foreign key describing the circuit position that this port address enables.
NODE_ADDR_LEVELS	VARCHAR2(2)	Determines how many pieces of equipment (levels up from the circuit_attachable piece) are used to determine the node address.
ORIG_ASSIGNMENT_IND	CHAR(1)	Used to designate whether or not an equipment assignment is the original assignment in a cross-connect chain. This is mainly used in the reconcile process of the circuit design to identify where the original assignment was made. Valid values are: ■ **Y** = Yes ■ **N** (default) = No
A_Z_OTHER_CD	CHAR(1)	Identifies the location of a piece of equipment residing on a circuit. Valid values are: ■ A = A location ■ Z = Z location ■ O = On the circuit but not at the A or Z location

Table 4–5 (Cont.) EXTRACT.MV_PORT_ADDRESS Materialized View

Column Name	Data Type	Description
EQUIPMENT_ID_VE	NUMBER(9)	A unique identifier visible only to the system, used to store and retrieve information about equipment. This plus the portaddr_seq_ve column indicate that this port_address is virtual and enabled by the port address referenced.
PORTADDR_SEQ_VE	NUMBER(9)	System generated number to uniquely identify and sequence port addresses for an equipment spec or piece of installed equipment. This plus the equipment_id_ve column indicate that this port_address is virtual and enabled by the port address referenced.
GROUP_IDENTIFIER	VARCHAR2(12)	Allows the user to associate mounting positions and port addresses with complement information (for example, DIGROUP A) for a piece of equipment (for example, D4 channel bank).
ADDITIONAL_ASSIGNMENT_SEQ_NBR	NUMBER(2)	This assignment sequence is used to keep track of equipment assignments for multiple assignments of a circuit to the same network.
NETWORK_NODE_ID	NUMBER(9,0)	Artificial key used to uniquely identify a network node. Allows nodes to be defined outside the network.

Table 4–6 describes the contents of the EXTRACT.MV_NETWORK_NODE materialized view.

Table 4–6 EXTRACT.MV_NETWORK_NODE Materialized View

Column Name	Data Type	Description
NETWORK_NODE_ID	NUMBER(9)	Artificial key used to uniquely identify a network node. Allows nodes to be defined outside of a network.
TFC_NETWORK_ID	NUMBER(9)	A system-generated number used to uniquely identify a network node
LOCATION_ID	NUMBER(9)	A unique identifier for a specific location. This ID is visible only to the system an dit is used to store and retrieve information about the location.
NODE_NAME	VARCHAR2(50)	An identifier for the network element.
NODE STATUS	CHAR(1)	Status of the network node (network location on a SONET ring). Valid values are: ■ 1 = Pending ■ 2 = In Service ■ 3 = Pending Removal
NODE_SEQUENCE	NUMBER(9)	Designates the nodes in sequential order to identify the switching or signaling directions, such as clockwise or counterclockwise.
TARGET_IDENTIFIER	VARCHAR2(25)	An equipment's network element address for the network node for communications between network elements and between operating systems and network elements.
NETWORK_ELEMENT_CD	CHAR(1)	Identifies the scope of the network node or network element. The network element can be a system (for example, a number of shelf assemblies) as with a switch or Digital Cross-connect System (DCS) or it can be a single shelf with a SONET network node. Values include: ■ S = System (for example, DCS) ■ N = SONET network node

Table 4–7 describes the contents of the MV_NS_COMPONENT materialized view.

Table 4–7 MV_NS_COMPONENT Materialized View

Column Name	Data Type	Description
NS_COMP_ID	NUMBER(9)	An Oracle sequence number that uniquely identifies entities of this type.
NST_COMP_TYPE	VARCHAR2(10)	A type of component that can be part of a network system. For example, local digital switch (LDS), central office terminal (COT), remote digital terminal (RDT), digital cross-connect system (DCS), and so on.
NS_COMP_ACRONYM	VARCHAR2(50)	A short name for a network system component. The default value comes from NST Component Type. Examples, of acronyms are host digital terminal (HDT) and remote services terminal (RST). The name of the NST Component Type is generic. This attribute allows the acronym represented by the NST Component Type to be tailored, as it can be different for different types of equipment. For example, one vendor may refer to an RDT as an RST.
NS_COMP_NM	VARCHAR2(50)	The name of the network system component. The default value for entities of this type comes from the NS_CONFIG_COMP_DEFAULT_NM. An example of this name is Remote Services Terminal.
LOCATION_ID	NUMBER(9)	A unique identifier for a specific location. This ID is visible only to the system and it is used to store and retrieve information about the location.
STATUS	CHAR(1)	Describes the current operational state. Valid values are: ■ 1 = (Pending) ■ 2 = (Assigned) ■ 3 = (In Progress) ■ 4 = (CLR Issued) ■ 5 = (DLR Issued) ■ 6 = (In Service) ■ 7 = (Pending Disconnect) ■ 8 = (Disconnected) ■ 9 = (Problem) ■ A = (Cancelled)
NETWORK_NODE_ID	NUMBER(9,0)	Unique ID to identify a network node. Foreign key to NETWORK_NODE. Allows nodes to be defined outside of a network.

Table 4–8 describes the contents of the EXTRACT.MV_NETWORK_LOCATION materialized view.

Table 4–8 EXTRACT.MV_NETWORK_LOCATION Materialized View

Column Name	Data Type	Description
LOCATION_ID	NUMBER(9)	A unique identifier for a specific location. This ID is visible only to the system and it is used to store and retrieve information about the location.
LOCATION_NAME	VARCHAR2(50)	The name for a location.
CLLI_CODE	VARCHAR2(20)	A location identification code that identifies specific locations or terminations. This code may be free-form and user-defined, or the Common Language Location Identification (CLLI) code administered by iconectiv.
NETLOC_TYPE_CD	CHAR(1)	Describes whether this network location represents an end user, terminal location or a CLLI. This column is used only with the new architecture location model where the network location table becomes an entity with sub-types (CLLI location, terminal location, and end user location). Valid values are: ■ E = end user ■ C = CLLI ■ T = terminal location ■ O = Other

Table 4–9 describes the contents of the EXTRACT.MV_EQUIPMENT_CA materialized view.

Table 4–9 EXTRACT.MV_EQUIPMENT_CA Materialized View

Column Name	Data Type	Description
EQUIP_CA_VALUE_ID	NUMBER(9)	An Oracle sequence that uniquely identifies entities of this type.
EQUIPMENT_ID	NUMBER(9)	An Oracle sequence number that uniquely identifies a piece of equipment.
CA_VALUE	VARCHAR2(1500)	The value taken on by an attribute, such as 320 for a Local Cell ID.
CA_VALUE_LABEL	VARCHAR2(50)	The name of the attribute associated to a value, such as Local Cell ID whose value is 320.
CA_VALUE_UOM	VARCHAR2(32)	The unit in a system that is used to determine the dimensions, area, volume, weight, or such of the attribute's value.
CA_USAGE_ID	NUMBER(9)	An Oracle sequence number that uniquely identifies an entity of this type.
CA_USAGE_VV_ID	NUMBER(9)	An Oracle sequence that uniquely identifies the valid value for an attribute associated to a building block.
MS_BB_ID	NUMBER(9)	Foreign Key from MS_BUILDING_BLOCK. Identifies the table or key (building block) to which this CA_Usage applies.
CURRENT_ROW_IND	CHAR(1)	Indicates whether this row of custom attributed is one of the current in-service rows for the network component.
CA_ID	NUMBER(9)	Foreign Key from CA_CUSTOMIZED_ATTRIBUTE. Identifies the CA value.

Table 4–10 describes the contents of the EXTRACT.MV_PORT_ADDRESS_CA materialized view.

Table 4–10 EXTRACT.MV_PORT_ADDRESS_CA Materialized View

Column Name	Data Type	Description
PORT_ADDR_CA_ VALUE_ID	NUMBER(10)	A system assigned unique identifier for port_addr_ca_value table. It is populated by an Oracle generated sequence and is hidden to the user. This information is used internally for tracking purposes
EQUIPMENT_ID	NUMBER(9)	An Oracle sequence number that uniquely identifies a piece of equipment.
MS_BB_ID	NUMBER(9)	Foreign Key from MS_BUILDING_BLOCK. Identifies the table or key (building block) to which this CA_Usage applies.
CURRENT_ROW_IND	CHAR(1)	Indicates whether this row of custom attributes is one of the current in-service rows for the network component.
PORTADDR_SEQ	NUMBER(10)	System generated number to uniquely identify and sequence port addresses for an equipment spec or piece of installed equipment.
CA_VALUE	VARCHAR2(1500)	The value taken on by an attribute, such as 320 for a Local Cell ID.
CA_VALUE_LABEL	VARCHAR2(50)	The name of the attribute associated to a value, such as Local Cell ID whose value is 320.
CA_VALUE_UOM	VARCHAR2(32)	The unit in a system that is used to determine the dimensions, area, volume, weight, or such of the attribute's value.
CA_USAGE_ID	NUMBER(9)	An Oracle sequence number that uniquely identifies an entity of this type.
CA_USAGE_VV_ID	NUMBER(9)	An Oracle sequence that uniquely identifies the valid value for an attribute associated to a building block.
CA_ID	NUMBER(9)	Foreign Key from CA_CUSTOMIZED_ATTRIBUTE. Identifies the CA value.

Table 4–11 describes the contents of the EXTRACT.MV_NS_COMP_EQUIP materialized view.

Table 4–11 EXTRACT.MV_NS_COMP_EQUIP Materialized View

Column Name	Data Type	Description
NS_COMP_ID	NUMBER(9)	An Oracle sequence number that uniquely identifies entities of this type.
NS_COMP_EQUIP_SEQ	NUMBER(3)	A number that together with NS_COMP_ID uniquely identifies entities of this type. This number starts with one for each value of NS_COMP_ ID.
EQUIPMENT_ID	NUMBER(9)	An Oracle sequence number that uniquely identifies entities of this type.

MSS Equipment Extract Process Normal Views

You use normal views to:

- Consolidate the data from multiple fast-refreshable materialized views
- Simplify the presentation of the data

All the MSS extract normal views retrieve data from the EXTRACT schema.

The following normal views enable Network Integrity to consolidate equipment-related data from the normal/materialized views:

- **EXTRACT.V_EQUIPMENT**: Network Integrity uses this normal view to consolidate the required information from the following materialized views:

  - EXTRACT.MV_EQUIPMENT
  - EXTRACT.MV_EQUIPMENT_SPEC
  - EXTRACT.MV_NETWORK_LOCATION
  - EXTRACT.MV_NETWORK_NODE
  - EXTRACT.MV_NS_COMPONENT
  - EXTRACT.MV_NS_COMP_EQUIP
- **EXTRACT.V_EQUIPMENT_SPEC**: Network Integrity uses this normal view to consolidate the required information from the following materialized views:
  - EXTRACT.MV_EQUIPMENT_SPEC
  - EXTRACT.MV_EQUIPMENT_SPEC_MPOS
- **EXTRACT.V_EQUIP_LEAF**: Network Integrity uses this normal view to consolidate the required information from the following materialized views:
  - EXTRACT.MV_EQUIPMENT
  - EXTRACT.MV_PORT_ADDRESS
  - EXTRACT.MV_CIRCUIT
  - EXTRACT.MV_MOUNTING_POSITION
- **EXTRACT.V_NN_FOR_HIER**: Network Integrity uses this normal view to consolidate the required information from the following materialized views:
  - EXTRACT.MV_EQUIPMENT
  - EXTRACT.MV_NETWORK_NODE
  - EXTRACT.MV_NS_COMPONENT
  - EXTRACT.MV_NS_COMP_EQUIP
- **EXTRACT.V_NETWORK_NODE**: Network Integrity uses this normal view to consolidate the required information from the following materialized views:
  - EXTRACT.MV_NETWORK_NODE
  - EXTRACT.MV_NS_COMPONENT
  - EXTRACT.MV_NETWORK_LOCATION
- **EXTRACT.V_PA_HIER**: Network Integrity uses this normal view to consolidate the required information from the following normal/materialized views:
  - EXTRACT.MV_EQUIPMENT
  - EXTRACT.V_EQUIPMENT_SPEC
  - EXTRACT.MV_MOUNTING_POSITION
  - EXTRACT.V_NN_FOR_HIER
  - EXTRACT.MV_NETWORK_LOCATION
  - EXTRACT.MV_PORT_ADDRESS
  - EXTRACT.MV_CIRCUIT
- **EXTRACT.V_MP_HIER**: Network Integrity uses this normal view to consolidate the required information from the following normal/materialized views:
  - EXTRACT.MV_MOUNTING_POSITION

- EXTRACT.V_EQUIPMENT_SPEC
- EXTRACT.MV_EQUIPMENT
- EXTRACT.MV_NETWORK_LOCATION
- EXTRACT.V_NN_FOR_HIER

MSS Circuit Extract Process

The MSS circuit extract process extracts the relevant circuit information from MSS and stores it into fast-refreshable, read-only materialized views within the EXTRACT schema in the MSS database, which is imported by the MSS Integration cartridge to compare the imported MSS data with discovered network data and resolve discrepancies on circuits in MSS.

MSS Circuit Extract Process Materialized Views

The MSS circuit extract process retrieves circuit information and stores it in the following MSS materialized views:

- **EXTRACT.MV_CIRCUIT**: Stores the attributes of a circuit. See Table 4–12 for more information.
- **EXTRACT.MV_CIRCUIT_POSITION**: Stores the channelization and assignment information for circuits. See Table 4–13 for more information.
- **EXTRACT.MV_CIRCUIT_XREF**: Stores information about the circuit cross-reference. See Table 4–14 for more information.
- **EXTRACT.MV_CIRCUIT_TRAIL**: Stores the hop-by-hop path for all non-channelized connectivity, including the allocation parameters such as VLAN ID, VPI/VCI, and DLCI, stored within custom attributes. See Table 4–15 for more information.
- **EXTRACT.MV_CIRCUIT_CA**: Stores all of the custom attributes of a template-based connection such as the Bit Rate, Broadband Service Category, and Capacity Allocation Thresholds. See Table 4–16 for more information.
- **EXTRACT.MV_TFC**: Stores additional information about CLF-formatted circuits. See Table 4–17 for more information.
- **EXTRACT.MV_NETWORK_LOCATION:** Stores the attributes and defining information of a network location. See Table 4–8 for more information.

The following tables describe the contents of the MSS materialized views in which the MSS circuit extract process stores the inventory data.

Table 4–12 describes the contents of the EXTRACT.MV_CIRCUIT materialized view.

Table 4–12 EXTRACT.MV_CIRCUIT Materialized View

Column Name	Data Type	Description
CIRCUIT_DESIGN_ID	NUMBER(9)	An identifier visible only to the system. Used for storing and retrieving information about a single circuit.
EXCHANGE_CARRIER_CIRCUIT_ID	VARCHAR2(53)	Commonly known as EC ID. This is the circuit number assigned by you or provided on the order by the OEC (Other Exchange Company). Oracle recommends that you use the iconectiv COMMON LANGUAGE CLF, CLS, CLT, and CLM formats for circuits; however, freeform formatted identifications are also stored here for "Other" facility, serial, and telephone-type formatted identifications. The other identifications are identified by "OTF," "OTS," and "OTT" as an ECCKT_Type.
ECCKT_TYPE	VARCHAR2(3)	The ECCKT type: ■ **CLF** (Common Language Facility) ■ **CLM** (Common Language Message) ■ **CLT** (Common Language Telephone) ■ **CLS** (Common Language Serial) ■ **OTF** (Free format of the CLF; unformatted facility) ■ **OTS** (Free format of the CLS; unformatted serial) ■ **OTT** (Free format of the CLT; unformatted telephone number, used for PSR dialtone products: line and trunk) ■ **CLF**, **CLT**, **CLS**, and **CLM** values indicate the iconectiv Common Language Circuit Identification format.
TYPE	CHAR(1)	The type of circuit. Valid values are: ■ **F** = Facility (connects two terminating locations with a rate that is usually higher than DS0 and carries other circuits). ■ **T** = Trunk (connects two serving office switching systems. A serving office can be a C.O. or MTSO). ■ **S** = Special (a dedicated circuit connecting two end-user locations or an end-user location to a coded location). ■ **P** = Product (a circuit ID created at the back end in PSR for dial tone circuits or trunks). ■ **C** = Template-based connections.
STATUS	CHAR(1)	The status for the circuit. Valid values are: ■ **1** = Pending ■ **3** = In Progress ■ **4** = Record Issued ■ **5** = DLR Issued ■ **6** = In Service ■ **7** = Pending Disconnect ■ **8** = Disconnected ■ **9** = Problem ■ **A** = Cancelled
RATE_CODE	VARCHAR2(10)	The rate code associated with the circuit. For example, DS0, DS1, DS3, N/A, and so on.

Table 4–12 (Cont.) EXTRACT.MV_CIRCUIT Materialized View

Column Name	Data Type	Description
SERVICE_TYPE_ CATEGORY	VARCHAR(20)	A description of the service provided, such as special services (for IntraLATA and LATA Access), switched services, and facility services.
SERVICE_TYPE_CODE	VARCHAR2(10)	Identifies the service provided by a circuit: ■ For special services, the characters in positions 3 and 4 of the CLCI-SS format indicate the service that is provided. ■ For message services, the characters in positions 5 and 6 of the CLCI-MSG Trunk Group format indicate the traffic use. ■ For facility service, the characters in positions 6 to 11 of the CLFI format indicate the facility type.
NST_CON_TYPE	NUMBER(6)	A value that along with the category and name logically identifies a type of connection used to join two network system component types. Valid values are: ■ **1** = Physical ■ **2** = Virtual ■ **3** = Group
NST_CON_CATEGORY_ CD	NUMBER(6)	Indicates the category of connector spec type. This value is denormalized from the value within the NST_CON_TYPE table. Similar to circuit type. It further defines the type of link or connector. Valid values are: ■ **1** = Facility (connects two terminating locations, with a rate that is usually higher than DS0, and carries other circuits). ■ **2** = Trunk (connects two serving office switching systems. A serving office can be a C.O. or MTSO). ■ **3** = Special (a dedicated circuit connecting two end user locations or an end user location to a coded location). ■ **4** = Product (a circuit ID created at the back end in the PSR for dial tone circuits or trunks). ■ **5** = Virtual circuits. ■ **6** = Bandwidth circuits. ■ **7** = Virtual connection that is not a PVC.
LOCATION_ID	NUMBER(9)	The A location ID of the circuit.
LOCATION_ID_2	NUMBER(9)	The Z location ID of the circuit.

Table 4–13 describes the contents of the EXTRACT.MV_CIRCUIT_POSITION materialized view.

Table 4–13 EXTRACT.MV_CIRCUIT_POSITION Materialized View

Column Name	Data Type	Description
CIRCUIT_DESIGN_ID	NUMBER(9)	The circuit design ID of the circuit.
CIRCUIT_POSITION_ NUMBER	NUMBER(5)	The channel position of the circuit with respect to the parent circuit.
CIRCUIT_DESIGN_ID_3	NUMBER(9)	An identifier visible only to the system, used for storing and retrieving information about a single circuit. This circuit is assigned to the circuit represented by Circuit Design ID

Table 4–13 (Cont.) EXTRACT.MV_CIRCUIT_POSITION Materialized View

Column Name	Data Type	Description
ADDITIONAL_ ASSIGNMENT_SEQ_NBR	NUMBER(2)	This assignment sequence is used to keep track of equipment assignments for multiple assignments of a circuit to the same network.
CIRCUIT_NODE_STATUS	CHAR(1)	Describes the current status of the circuit position. Valid values are: ■ **1** = Unassigned ■ **2** = Pending installation work order ■ **3** = In Service ■ **4** = Pending removal work order ■ **5** = Trouble
STS_CHAN_NBR	NUMBER(3)	The synchronous transport number that is used to identify the actual designation for the virtual channel assignment. This is used in the concatenation process of network assignment identification. For example, 12-7-4, where 12 equals the STS assignment.
VTG_CHAN_NBR	NUMBER(1)	The virtual tributary group (VTG) number that is used to identify the actual designation for the virtual channel assignment. This is used in the concatenation process of network assignment identification. For example, 12-7-4, where 7 equals the VTG assignment.
VT_CHAN_NBR	NUMBER(1)	The virtual tributary (VT) number that is used to identify the actual designation for the virtual channel assignment. This is used in the concatenation process of network assignment identification. For example, 12-7-4, where 4 equals the VT assignment.
PROTECTED_PATH_TRI	CHAR(1)	Distinguishes the primary path from the protection path. Valid values are: ■ **Y** = Identifies a protected path. ■ **N** = Identifies a primary (working) path. ■ **Null** = Identifies that the assignment is not part of a network path. The attribute is set when creating a new network assignment block using the optical network provisioning assistant. It is used in mass reconcile and reconciliation from the circuit reconciliation window to keep the design lines in the proper order with the primary path displayed on top of the protection path in the Connection Design window.

Table 4–14 describes the contents of the EXTRACT.MV_CIRCUIT_XREF materialized view.

Table 4–14 EXTRACT.MV_CIRCUIT_XREF Materialized View

Column Name	Data Type	Description
CIRCUIT_DESIGN_ID	NUMBER(9)	An identifier visible only to the system. Used for storing and retrieving information about a single circuit.
CIRCUIT_XREF_SEQ	NUMBER(3)	Sequence number that starts over with every new relationship to circuit. (Not an Oracle sequence).

Table 4–14 (Cont.) EXTRACT.MV_CIRCUIT_XREF Materialized View

Column Name	Data Type	Description
CIRCUIT_XREF_ECCKT	VARCHAR2(60)	ECCKT that needs to be cross-referenced to an ECCKT that was provisioned. An ECCKT that another provider provisioned or an alias of one of the circuits.
LOCATION_ID	NUMBER(9)	The A location ID for the circuit.
STATUS	CHAR(1)	The status for the circuit. Valid values are: ■ **1** = Pending ■ **3** = In Progress ■ **4** = Record Issued ■ **5** = DLR Issued ■ **6** = In Service ■ **7** = Pending Disconnect ■ **8** = Disconnected ■ **9** = Problem ■ **A** = Cancelled

Table 4–15 describes the contents of the EXTRACT.MV_CIRCUIT_TRAIL materialized view.

Table 4–15 EXTRACT.MV_CIRCUIT_TRAIL Materialized View

Column Name	Data Type	Description
CIRCUIT_DESIGN_ID_PARENT	NUMBER(9)	This is a foreign key from the CIRCUIT table. Represents the parent when connectors are associated with other connectors. For example, when a connection is allocated to a link, the link is the parent. Or, when multiple connectors are associated to one another to create a group, the parent is the group. For example, this occurs for inverse multiplexing.
CIRCUIT_DESIGN_ID_CHILD	NUMBER(9)	This is a foreign key from the CIRCUIT table. Represents the child when connectors are associated with other connectors. For example, when a connection is allocated to a link, the connection is the child. Or, when multiple connectors are associated to one another to create a group, the individual connectors are the children. For example, this occurs for inverse multiplexing.
CA_ID	NUMBER(9)	Foreign Key from CA_CUSTOMIZED_ATTRIBUTE. Identifies the CA value.
CA_VALUE	VARCHAR2(1500)	The value taken on by an attribute, such as 320 for a Local Cell ID.
CA_USAGE_ID	NUMBER(9)	An Oracle sequence number that uniquely identifies an entity of this type.
CA_VALUE_LABEL	VARCHAR2(50)	The name of the attribute associated to a value, such as Local Cell ID whose value is 320.

Table 4–15 (Cont.) EXTRACT.MV_CIRCUIT_TRAIL Materialized View

Column Name	Data Type	Description
CA_USAGE_VV_ID	NUMBER(9)	An Oracle sequence that uniquely identifies the valid value for an attribute associated to a building block.
MS_BB_ID	NUMBER(9)	Foreign Key from MS_BUILDING_BLOCK. Identifies the table or key (building block) to which this CA_Usage applies.
NS_COMP_ID	NUMBER(9)	An Oracle sequence number that uniquely identifies entities of this type.

Table 4–16 describes the contents of the EXTRACT.MV_CIRCUIT_CA materialized view.

Table 4–16 EXTRACT.MV_CIRCUIT_CA Materialized View

Column Name	Data Type	Description
CONN_CA_VALUE_ID	NUMBER(9)	An Oracle sequence that uniquely identifies an attribute.
CIRCUIT_DESIGN_ID	NUMBER(9)	An identifier visible only to the system. Used for storing and retrieving information about a single circuit.
CA_VALUE_LABEL	VARCHAR2(50)	The name of the attribute associated to a value, such as Local Cell ID whose value is 320.
CA_VALUE	VARCHAR2(1500)	The value taken on by an attribute, such as 320 for a Local Cell ID.
CA_VALUE_UOM	VARCHAR2(32)	The unit in a system that is used to determine the dimensions, area, volume, weight, or such of the attribute's value.
CA_USAGE_ID	NUMBER(9)	An Oracle sequence number that uniquely identifies an entity of this type.
CA_USAGE_VV_ID	NUMBER(9)	An Oracle sequence that uniquely identifies the valid value for an attribute associated to a building block.
MS_BB_ID	NUMBER(9)	Foreign Key from MS_BUILDING_BLOCK. Identifies the table or key (building block) to which this CA_Usage applies.
CA_ID	NUMBER(9)	Foreign Key from CA_CUSTOMIZED_ATTRIBUTE. Identifies the CA value.
CURRENT_ROW_IND	CHAR(1)	Indicates whether this row of custom attributed is one of the current in-service rows for the network component.

Table 4–17 describes the contents of the EXTRACT.MV_TFC materialized view.

Table 4–17 EXTRACT.MV_TFC Materialized View

Column Name	Data Type	Description
CIRCUIT_DESIGN_ID	NUMBER(9)	An identifier visible only to the system. Used for storing and retrieving information about a circuit.
VIRTUAL_IND	CHAR(1)	Indicates whether the circuit is part of a virtual assignment or not. Valid values are: ■ **Y**: Yes ■ **N**: No
TFC_NETWORK_ID	NUMBER(9)	The unique ID which identifies a network. A TFC Network maintains information concerning the various transmission facility circuit network topologies, such as point-to-point, linear Add/Drop, hubbing, and rings. These fiber networks are normally SONET-based; however, specific asynchronous facilities can be included. Other information pertaining to these networks are the Fiber Network Identification, assignment methods, protection schemes, and switching directions.

MSS Circuit Extract Process Normal Views

You use normal views to:

- Consolidate the data from multiple fast-refreshable materialized views
- Simplify the presentation of the data

All the MSS extract normal views retrieve data from the EXTRACT schema.

The following normal views enable Network Integrity to consolidate circuit-related data from the normal/materialized views:

- **EXTRACT.V_CIRCUIT**: Network Integrity uses this normal view to consolidate the required information from the following materialized views:
  - EXTRACT.MV_CIRCUIT
  - EXTRACT.MV_NETWORK_LOCATION
- **EXTRACT.V_PA_HIER**: Network Integrity uses this normal view to consolidate the required information from the following normal/materialized views:
  - EXTRACT.MV_CIRCUIT
  - EXTRACT.MV_MOUNTING_POSITION
  - EXTRACT.MV_EQUIPMENT
  - EXTRACT.V_EQUIPMENT_SPEC
  - EXTRACT.V_NN_FOR_HIER
  - EXTRACT.MV_NETWORK_LOCATION
  - EXTRACT.MV_PORT_ADDRESS
- **EXTRACT.V_CIRCUIT_POSITION**: Network Integrity uses this normal view, which uses the JKLM stored function, to consolidate the information about JKLM values from the following materialized views:
  - EXTRACT.MV_CIRCUIT
  - EXTRACT.MV_CIRCUIT_POSITION

Extending the MSS Extract Process

This section provides information on extending the MSS extract process.

The system integrator can extend the Equipment/Circuit extract process by adding additional columns to the definition of an existing materialized view, create new materialized views, and create new normal views to retrieve data from new or existing materialized views. When retrieving data from a table that does not already have a materialized view log, you must first create the materialized view log in order to incrementally refresh the materialized view. This process does not require writing any PL/SQL logic.

When creating a new materialized view or extending an existing materialized view, Oracle recommends that you structure the materialized views to be incrementally (fast) refreshed.

The reference integration also provides normal views to consolidate and simplify the data from multiple materialized views. See the following sections for more information on the normal views:

- MSS Equipment Extract Process Normal Views
- MSS Circuit Extract Process Normal Views

You can find more information about materialized view concepts and architecture at the following Web site:

http://docs.oracle.com/cd/B28359_01/server.111/b28326/repmview.htm#i34980

5

About Cartridge Modeling

This chapter provides information on how imported data is modeled.

About Cartridge Modeling

To facilitate discrepancy detection and resolution, the Oracle Communications Network Integrity MSS Integration cartridge models the imported data from Oracle Communications MetaSolv Solution (MSS) to the Oracle Communications Information Model.

The MSS Integration cartridge uses different modeling logic depending on the action it is performing. See the following sections for more information:

- About Import Data Modeling
- About Discrepancy Resolution Modeling

About Import Data Modeling

This section explains how Network Integrity models data imported from MSS.

You can configure various parameters in the Network Integrity UI to determine the quantity of data to import from MSS. Network Integrity logically applies the parameters to filter the imported data. When no filtering parameters are configured in the UI, Network Integrity imports all MSS data.

API Mapping

Network Integrity uses APIs to map the imported MSS data to the Information Model, which allows the MSS data to map directly to TMF814 entities. The MSS Integration cartridge uses TMF814 specifications to model the MSS data.

Table 5–1 shows the relationship between the imported MSS data, physical TMF814 entities, and physical Information Model entities.

Table 5–1 MSS Data Modeling to the Physical Information Model Tree

MSS Data Object	TMF814 Entity	Information Model Entity
Network Node	Managed Element (ME)	Physical Device, Logical Device
Equipment	Equipment Holder (Rack)	Equipment
Equipment	Equipment Holder (Shelf)	Equipment
Equipment	Equipment Holder (Sub-Shelf)	Equipment
Mounting Position	Equipment Holder (Slot)	Equipment Holder

Table 5–1 (Cont.) MSS Data Modeling to the Physical Information Model Tree

MSS Data Object	TMF814 Entity	Information Model Entity
Mounting Position	Equipment Holder (Sub-Slot)	Equipment Holder
Equipment	Equipment (Card)	Equipment
Port	Physical Termination Point (PTP)	Physical Port

Table 5–2 shows the relationship between the imported MSS data, logical TMF814 entities, and logical Information Model entities.

Table 5–2 MSS Data Modeling to the Logical Information Model Tree

MSS Data	TMF814 Entity	Information Model Entity
Network Node from EXTRACT.MV_ EQUIPMENT	ME	LogicalDevice
Port from EXTRACT.MV_PORT_ ADDRESS	Point Termination Port (PTP) and Floating Termination Point (FTP)	DeviceInterface
(Port) Circuit and (Rack/Shelf) Circuit from EXTRACT.V_PA_HIER	Connection Termination Point (CTP)	DeviceInterface
N/A	LayeredParameters	DeviceInterfaceConfigurationItem

Field Mapping

The following tables explain the field mappings for each imported MSS object.

- Table 5–3, " Physical Device Field Mapping"
- Table 5–4, " Root Equipment Field Mapping"
- Table 5–5, " Non-Root Equipment Field Mapping"
- Table 5–6, " Equipment Holder Field Mapping"
- Table 5–7, " Physical Port Field Mapping"
- Table 5–8, " Logical Device Field Mapping"
- Table 5–9, " Media Interface Field Mapping"
- Table 5–10, " Pipe Field Mapping"
- Table 5–11, " Transport Pipe Field Mapping"
- Table 5–12, " STM Link Field Mapping"
- Table 5–13, " Trail Path Field Mapping"
- Table 5–14, " Trail Pipe Field Mapping"
- Table 5–15, " Pipe Termination Point Field Mapping"

Table 5–3 Physical Device Field Mapping

Information Model Attribute	Information Model Support	MSS Inventory Field	Used for Discrepancy Detection
Id	Static	EXTRACT.MV_EQUIPMENT.NETWORK_NODE_ID/EXTRACT.MV_NETWORK_NODE.NETWORK_NODE_ID	No
name	Static	EXTRACT.V_EQUIPMENT.NW_NODE_NAME/EXTRACT.MV_NETWORK_NODE.NETWORK_NODE_NAME	No
description	Static	N/A	No
discoveredVendorName	Dynamic	EXTRACT.V_EQUIPMENT_SPEC.VENDOR_NAME	No
serialNumber	Static	EXTRACT.MV_EQUIPMENT.SERIAL_NBR	No
physicalLocation	Static	EXTRACT.V_EQUIPMENT.LOC_ID_CLLI_CODE/V_NETWORK_NODE.CLLI_CODE	No
softwareRev	Dynamic	EXTRACT.MV_EQUIPMENT.SOFTWARE_RELEASE_IDENTIFIER	No
modelName	Dynamic	EXTRACT.MV_EQUIPMENT_SPEC.EQUIPSPEC_TYPE/V_NETWORK_NODE.NST_COMP_TYPE	No
nativeEmsName	Static	EXTRACT.V_EQUIPMENT.NW_NODE_NAME/ EXTRACT.MV_NETWORK_NODE.NETWORK_NODE_NAME	No
userLabel	Dynamic	EXTRACT.V_EQUIPMENT.NW_NODE_NAME/ EXTRACT.MV_NETWORK_NODE.NETWORK_NODE_NAME	No
owner	Dynamic	EXTRACT.V_EQUIPMENT_SPEC.VENDOR_NAME	No

Table 5–4 Root Equipment Field Mapping

Information Model Attribute	Information Model Support	MSS Inventory Field	Used for Discrepancy Detection
Id	Static	EXTRACT.V_EQUIPMENT.EQUIPMENT_ID	No
name	Static	EXTRACT.MV_EQUIPMENT_SPEC.EQUIPMENT_ACRONYM	No
description	Static	N/A	No
discoveredVendorName	Dynamic	EXTRACT.V_EQUIPMENT_SPEC.VENDOR_NAME	Yes
serialNumber	Static	EXTRACT.MV_EQUIPMENT.SERIAL_NBR	Yes
physicalLocation	Static	EXTRACT.V_EQUIPMENT.LOC_ID_CLLI_CODE	No
discoveredPartNumber	Dynamic	EXTRACT.MV_EQUIPMENT_SPEC.VENDOR_PART_NUMBER	Yes
hardwareRev	Dynamic	EXTRACT.MV_EQUIPMENT.VERSION_OF_HARDWARE_INSTALLED	No
modelName	Dynamic	EXTRACT.MV_EQUIPMENT_SPEC.EQUIPSPEC_TYPE	No

Table 5–4 (Cont.) Root Equipment Field Mapping

Information Model Attribute	Information Model Support	MSS Inventory Field	Used for Discrepancy Detection
nativeEmsName	Static	EXTRACT.MV_EQUIPMENT_SPEC.EQUIPMENT_ACRONYM	No
expectedObjectType	Dynamic	N/A	No
serviceState	Dynamic	EXTRACT.MV_EQUIPMENT.AVAILABILITY_STATUS Valid values are IN_SERVICE, OUT_OF_SERVICE, IN_MAINTENANCE, UNKNOWN, TESTING	No
userLabel	Dynamic	EXTRACT.MV_EQUIPMENT.EQUIPMENT_NAME	No
owner	Dynamic	EXTRACT.MV_EQUIPMENT_SPEC.VENDOR_NAME	No

Table 5–5 Non-Root Equipment Field Mapping

Information Model Attribute	Information Model Support	MSS Inventory Field	Used for Discrepancy Detection
Id	Static	Derived From EQUIP_ID_2_HIER with mounting_position_seq	No
name	Static	Derived from EXTRACT.V_MP_HIER.EQUIPMENT_ACRONYM_HIER	No
description	Static	N/A	No
discoveredVendorName	Dynamic	Derived from EXTRACT.V_MP_HIER.VENDOR_NAME_HIER	No
serialNumber	Static	Derived from EXTRACT.V_MP_HIER.SERIAL_NBR_HIER	Yes
physicalLocation	Static	EXTRACT.V_EQUIPMENT.LOC_ID_CLLI_CODE This field value corresponds to the root equipment.	No
discoveredPartNumber	Dynamic	Derived from EXTRACT.V_MP_HIER.VENDOR_PART_NUMBER_HIER	Yes
hardwareRev	Dynamic	N/A	Yes
modelName	Dynamic	Derived from EXTRACT.V_MP_HIER.EQUIPSPEC_TYPE_HIER	No
nativeEmsName	Static	Derived from EXTRACT.V_MP_HIER.EQUIPMENT_ACRONYM_HIER	No
expectedObjectType	Dynamic	N/A	No

Table 5–5 (Cont.) Non-Root Equipment Field Mapping

Information Model Attribute	Information Model Support	MSS Inventory Field	Used for Discrepancy Detection
serviceState	Dynamic	EXTRACT.V_MP_HIER.AVAILABILITY_STATUS_HIER This field is assigned one of the following values: IN_SERVICE, OUT_OF_SERVICE, IN_MAINTENANCE, UNKNOWN, TESTING.	No
userLabel	Dynamic	Derived from EXTRACT.V_MP_HIER.EQUIPMENT_NAME_HIER	No
owner	Dynamic	Derived from EXTRACT.V_MP_HIER.VENDOR_NAME_HIER	No

Table 5–6 Equipment Holder Field Mapping

Information Model Attribute	Information Model Support	MSS Inventory Field	Used for Discrepancy Detection
Id	Static	Derived From EQUIP_ID_2_HIER with mounting_position_seq	No
name	Static	Derived from V_MP_HIER.EQUIPMENT_ACRONYM_HIER The slot number is equivalent to mounting position number.	No
description	Static	N/A	No
serialNumber	Static	N/A	No
physicalLocation	Static	N/A	No
modelName	Dynamic	EXTRACT.MV_EQUIPMENT_SPEC.EQUIPSPEC_TYPE	No
nativeEmsName	Static	Derived from V_MP_HIER.EQUIPMENT_ACRONYM_HIER	No
userLabel	Dynamic	Derived from V_MP_HIER.EQUIPMENT_NAME_HIER	No
owner	Dynamic	Derived from V_MP_HIER.VENDOR_NAME_HIER	No

Table 5–7 Physical Port Field Mapping

Information Model Attribute	Information Model Support	MSS Inventory Field	Used for Discrepancy Detection
Id	Static	Derived with parent-id from EXTRACT.MV_PORT_ADDRESS.PORTADDR_SEQ	No
name	Static	Derived from EXTRACT.MV_PORT_ADDRESS.PORTADDR_SEQ	Yes
description	Static	N/A	No
portNumber	Static	N/A	No
customerPortName	Static	N/A	No
vendorPortName	Static	N/A	No
serialNumber	Static	N/A	No

Table 5–7 (Cont.) Physical Port Field Mapping

Information Model Attribute	Information Model Support	MSS Inventory Field	Used for Discrepancy Detection
physicalLocation	Static	N/A	No
nativeEmsName	Static	N/A	No
direction	Dynamic	Bidirection	No
tpProtectionAssociation	Dynamic	N/A	No
edgePoint	Dynamic	True	No
physicalAddress	Static	EXTRACT.MV_PORT_ADDRESS.NODE_ADDRESS when EXTRACT.MV_PORT_ADDRESS.PORTADDR_TYPE value corresponds to physical	No

Table 5–8 Logical Device Field Mapping

Information Model Attribute	Information Model Support	MSS Inventory Field	Used for Discrepancy Detection
Id	Static	N/A	No
name	Static	EXTRACT.V_EQUIPMENT.NW_NODE_NAME	Yes
description	Static	N/A	No
specification	Static	N/A	No
nativeEmsAdminServiceState	Static	N/A	No
nativeEmsServiceState	Static	N/A	No
physicalLocation	Static	EXTRACT.V_EQUIPMENT.LOC_ID_CLLI_CODE	No

Table 5–9 Media Interface Field Mapping

Information Model Attribute	Information Model Support	MSS Inventory Field	Used for Discrepancy Detection
Id	Static	N/A	No
name	Static	Derived from EXTRACT.MV_PORT_ADDRESS.PORTADDR_SEQ	No
description	Static	N/A	No
ifType	Static	PTP, FTP, or CTP, according to the entity being modeled	No
interfaceNumber	Static	N/A	No
customerInterfaceNumber	Static	N/A	No
vendorInterfaceNumber	Static	N/A	No
nativeEmsName	Static	N/A	No
nativeEmsAdminServiceState	Static	N/A	No
nativeEmsServiceState	Static	N/A	No
mtuSupported	Static	N/A	No
mtuCurrent	Static	N/A	No

Table 5–9 (Cont.) Media Interface Field Mapping

Information Model Attribute	Information Model Support	MSS Inventory Field	Used for Discrepancy Detection
physicalAddress	Static	N/A	No
physicalLocation	Static	N/A	No
minSpeed	Static	N/A	No
maxSpeed	Static	N/A	No
nominalSpeed	Static	N/A	No
connectionState	Dynamic	EXTRACT.MV_CIRCUIT.STATUS	No
tpMappingMode	Dynamic	N/A	No
Direction	Dynamic	Bidirection	No
tpProtectionAssociation	Dynamic	N/A	No
edgePoint	Dynamic	N/A	No
userLabel	Dynamic	N/A	No
owner	Dynamic	N/A	No
activeEmsConnectorPresent	Static	N/A	No

Table 5–10 Pipe Field Mapping

Information Model Attribute	Information Model Support	MSS Inventory Field	Used for Discrepancy Detection
name	Static	V_CIRCUIT.EXCHANGE_CARRIER_CIRCUIT_ID	Yes
Id	Static	EXTRACT.MV_CIRCUIT.CIRCUIT_DESIGN_ID	No
gapPipe	Static	Hard-coded to **FALSE**	No
physicalLocation	Static	EXTRACT.V_CIRCUIT.LOC_A_CLLI_CODE	No
layerRate	Dynamic	EXTRACT.MV_CIRCUIT.RATE_CODE	No
Rerouted	Dynamic	Hard-coded to **FALSE**	No
partial	Dynamic	Derived: set to **FALSE** if the number of ports is greater than one, else it is **TRUE**.	No

Table 5–11 Transport Pipe Field Mapping

Information Model Attribute	Information Model Support	MSS Inventory Field	Used for Discrepancy Detection
name	Static	V_CIRCUIT.EXCHANGE_CARRIER_CIRCUIT_ID	Yes
Id	Static	EXTRACT.MV_CIRCUIT.CIRCUIT_DESIGN_ID	No
gapPipe	Static	Hard-coded to **FALSE**	No
physicalLocation	Static	EXTRACT.V_CIRCUIT.LOC_A_CLLI_CODE	No
layerRate	Dynamic	EXTRACT.MV_CIRCUIT.RATE_CODE	No
Rerouted	Dynamic	Hard-coded to **FALSE**	No
partial	Dynamic	Derived: set to **FALSE** if the number of ports is greater than one, else it is **TRUE**.	No

Table 5–12 STM Link Field Mapping

Information Model Attribute	Information Model Support	MSS Inventory Field	Used for Discrepancy Detection
name	Static	EXTRACT.V_CIRCUIT.EXCHANGE_CARRIER_CIRCUIT_ID	Yes
Id	Static	EXTRACT.MV_CIRCUIT.CIRCUIT_DESIGN_ID	No
gapPipe	Static	Hard-coded to **FALSE**	No
physicalLocation	Static	EXTRACT.V_CIRCUIT.LOC_A_CLLI_CODE	No
layerRate	Dynamic	EXTRACT.MV_CIRCUIT.RATE_CODE	No

Table 5–13 Trail Path Field Mapping

Information Model Attribute	Information Model Support	MSS Inventory Field	Used for Discrepancy Detection
name	Static	EXTRACT.V_CIRCUIT_POSITION.EXCHANGE_CARRIER_CIRCUIT_ID	Yes
gapPipe	Static	Hard-coded to **FALSE**	No
layerRate	Dynamic	EXTRACT.V_CIRCUIT_POSITION.RATE_CODE	No
channel	Dynamic	EXTRACT.V_CIRCUIT_POSITION.JKLM	Yes

Table 5–14 Trail Pipe Field Mapping

Information Model Attribute	Information Model Support	MSS Inventory Field	Used for Discrepancy Detection
name	Static	EXTRACT.V_CIRCUIT_POSITION.EXCHANGE_CARRIER_CIRCUIT_ID	Yes
AEnd	Static	Derived from the originating port name	No
ZEnd	Dynamic	Derived from the terminating port name	No
channel	Dynamic	EXTRACT.V_CIRCUIT_POSITION.JKLM	Yes

Table 5–15 Pipe Termination Point Field Mapping

Information Model Attribute	Information Model Support	MSS Inventory Field	Used for Discrepancy Detection
name	Static	Hierarchical name derived from of the following: ■ EXTRACT.V_PA_HIER.EQUIPMENT_ACRONYM_HIER ■ EXTRACT.V_PA_HIER.MTG_POS_NBR_HIER ■ EXTRACT.V_PA_HIER.PORTADDR_SEQ	Yes

Table 5–15 (Cont.) Pipe Termination Point Field Mapping

Information Model Attribute	Information Model Support	MSS Inventory Field	Used for Discrepancy Detection
physicalLocation	Static	EXTRACT.V_PA_HIER.CLLI_CODE	No
Device	Dynamic	EXTRACT.V_PA_HIER.NW_NODE_NAME	Yes
Directionality	Dynamic	EXTRACT.V_PA_HIER.A_Z_OTHER_CD	Yes

Data Import Algorithm

This section explains the various algorithms and logic used to model the imported MSS inventory data.

Import Equipment Hierarchy Algorithm

This algorithm uses spring framework pagination to incrementally retrieve node names from each page created by the Page Creator processor. This algorithm uses the following logic:

1. Gets all the node names from EquipmentExportDAO.
2. For each node:
   a. Creates a physical and logical device in the Information Model.
   b. Gets the root equipment (such as a shelf, or rack) from EquipmentExportDAO.
3. For each root equipment:
   a. Verifies that the occupiedMountingPositions attribute value is **0**:
      – If yes, the equipment is modeled as a rack.
      – If no, verifies whether the root equipment has a mounting position. If there is a mounting position, the equipment is modeled as a shelf.
   b. Associates the rack and shelf to the physical device and states the equipment type (rack or shelf) in the equipment name.
   c. Retrieves the child equipment hierarchy from EquipmentPositionHierDAO (child equipment are represented as EquipmentPositionHier entities).
   d. Queries the ports in the root equipment hierarchy from EquipmentPortAddressDAO (ports are represented as EquipmentPortAddress entities).
   e. Creates a list that maps card IDs to their ports.
   f. For each EquipmentPortAddress entity, verifies if leafEquipmentId equals EquipmentId.
      – If yes, models the port as a physical port (FTP) and associates it with the parent equipment. Models a media interface, associates it to the physical port, and sets the media interface as a child of the logical device.
      – If no, fills the card ID and all ports with the leafEquipmentId value as the list of associated ports.
   g. Builds the child equipment hierarchy in the Information Model with the EquipmentPositionHier entities.
   h. Saves the modeling information to the physical and logical trees.

Build Equipment Hierarchy Algorithm

The input for this algorithm is the list of EquipmentPositionHier entities and the map of cards-to-ports. Each EquipmentPositionHier is an immediate child (either a card or shelf) of the root equipment. This algorithm uses the following logic:

1. Gets the list of mounting position numbers down the hierarchy for each EquipmentPositionHier entity.
2. For each mounting position number:
   a. Models a shelf object as an equipment entity if its parent is a rack object and the shelf is not yet built in this hierarchy and associates the child with its parent.
   b. Models a card object as an equipment entity if the shelf is already built. Models a slot object as an equipment holder entity for the card. If the slot is already built, models a sub-slot object as an equipment holder entity. Associates children objects with their parent.
   c. Gets the list of ports associated with the card from the map of cards-to-ports.
   d. Models each port as a physical port and a media interface entity. Associates the media interface and the physical port with the logical device.
   e. Adds the EquipmentPortAddressDAO entities to each port as a collection.

Import Circuit Hierarchy Algorithm

MSS and the MSS Integration cartridge distinguish between the following types of circuits:

- STM (physical circuit, such as STM1 or STM4)
- HOT (logical circuit, such as VC4)
- LOP (logical circuit, such as E1, E3, or E4)

Table 5–16 shows the relationship between the imported MSS data, physical TMF814 entities, and physical Information Model entities.

Table 5–16 MSS Circuit Data Mapping to Information Model

MSS Data	TMF814 Entity	Information Model Entity
STM-Type Circuit	STM Link	Link
LOP-Type Circuit	Customer Circuit or LOP	Pipe
HOT-Type Circuit	HOT	Transport Pipe

MSS organizes logical circuits as children to physical circuits. The input for this algorithm is the list of EquipmentPortAddress entities produced by the Build Equipment Hierarchy algorithm. This algorithm uses the following logic:

1. For each port entity from the EquipmentPortAddress table:
   a. Gets the CircuitPortAddress instance containing the circuit ID corresponding to an STM link passing through that port.
   b. Queries the CircutExport table to get the STM link with the circuit ID.
   c. Obtains the aPort and zPort for the STM link from the CircuitPortAddress table.

d. Models the STM link as an optical topological link entity, models its ports as pipe termination point entities, and associates the ports to their link.

e. Adds the modeled STM link circuit ID to the stmSet collection.

2. Identifies all VC4 HOT circuits for all the STM circuit IDs in the stmSet collection by querying the trail from CircuitPositionDAO. For each trail:

a. Identifies E4 customer circuits by counting its children in the CircuitPosition table. For each E4 customer circuit:

- Queries the CircuitExport table for the circuit ID and models the CircuitExport objects as a pipe entities.
- Queries the ports for the circuit from the CircuitPortAddress table, models them as PTPs, and associates them to their pipe entity.
- Queries the trail path from the CircuitPosition table, models them as trail path entities, and associates them to their pipe entity.
- Verifies the JKLM value for the trail path, and corrects it if necessary.

b. Identifies VC4 HOT circuits by evaluating the layer rate code. For each VC4 HOT circuit:

- Models them as transport pipe entities.
- Queries the CircuitExport table for circuit ID and models the CircuitExport object as a transport pipe entity.
- Queries the parent STM link from the CircuitPosition table, and queries the STM link ports from the CircuitPortAddress table.
- Determines the start-port and end-port from the STM link ports, models them as pipe termination points entities, and associates them to their transport pipe entity.
- Queries the trail paths from the CircuitPosition table, models them as trail path entities, and associates them to their pipe entity.

c. Adds the modeled transport pipe circuit ID to the vc4sForLops list.

3. Queries E1 and E3 trail circuits for each VC4 circuit in the vc4sForLop list. For each trail circuit:

- Queries circuits from the CircuitExport table and models them as pipe entities.
- Queries customer circuit ports from the CircuitPortAddress table, models them as pipe termination point entities, and associates the ports to the pipe.
- Queries trail paths from the CircuitPosition table, models them as trail path entities, and associates them to the pipe.
- Verifies the JKLM value for the trail path, and corrects it if necessary.

About Discrepancy Resolution Modeling

The Discrepancy Resolution action uses different field mappings depending on the type of entity being resolved.

Discrepancy Resolution Field Mapping for Equipment

The MSS Integration cartridge uses MSS CORBA API methods to resolve equipment discrepancies. Each API method runs a Type Java object. Table 5–17 explains the field mapping for physical entities mapping to circuits in the Information Model.

Table 5–17 Equipment Resolution Field Mapping

MSS CORBA API	Information Model Attribute	API Type Field
MetaSolv.CORBA.WDIEquipmentTypes.EquipSpecQuery (Equipment entity)	▪ discoveredVendorName ▪ discoveredPartNumber	▪ Manufacturer ▪ partNumber
MetaSolv.CORBA.WDIEquipmentTypes_v2.EquipmentInstallation (Equipment entity)	▪ serialNumber ▪ name ▪ serviceState ▪ hardwareRev	▪ EquipmentModification.serialNumber ▪ startingMountingPosition (derived) ▪ Status ▪ ConfigurationModificationSeq.hardwareVersion
MetaSolv.CORBA.WDIEquipmentTypes_v2.EquipmentUpdate (Equipment entity)	▪ serialNumber ▪ hardwareRev	▪ EquipmentModification.serialNumber ▪ ConfigurationModificationSeq.hardwareVersion
MetaSolv.CORBA.WDIEquipmentTypes_v2.EquipInstallQuery (Equipment entity)	▪ Name ▪ physicalLocation ▪ discoveredPartNumber ▪ modelName	▪ acronym ▪ installedAtLocationCode ▪ partNumber ▪ type
MetaSolv.CORBA.WDIEquipmentTypes_v2.NetworkElementQuery (Equipment entity)	▪ Name ▪ physicalLocation	▪ Name ▪ networkLocation
MetaSolv.CORBA.WDIEquipmentTypes_v2.NetworkElementCreate (PhysicalDevice entity)	▪ Name ▪ physicalLocation ▪ Description	▪ Name ▪ locId ▪ Description
MetaSolv.CORBA.WDIEquipmentTypes_v2.NetworkElementResult (PhysicalDevice entity)	▪ ID	▪ networkNodeId
MetaSolv.CORBA.WDINetworkLocationTypes_v2.NetworkLocationQuery (PhysicalDevice entity)	▪ physicalLocation	▪ locationCode

Discrepancy Resolution Field Mapping for Circuits

Table 5–18 explains the field mapping for circuit entities mapping to circuits in the Information Model.

Table 5–18 Circuit Resolution Field Mapping

Information Model Entity	Information Model Attribute	Connection Field
Pipe	Specification Name	Ratecode
Pipe	Termination points	Ports
Pipe	Channel	Circuit positions
Pipe	PipeTerminationPoint.Originating.location	ALocation
Pipe	PipeTerminationPoint.Terminating.location	ZLocation

6

About Design Studio Construction

This chapter provides information on the composition of the Oracle Communications Network Integrity MSS Integration cartridge from the Oracle Communications Design Studio perspective.

Model Collections

The MSS Integration cartridge models imported data to the TMF814 Generic specification. See *Network Integrity Optical TMF814 CORBA Cartridge Guide* for more information.

Actions

The following tables outline the Design Studio construction of the MSS Integration cartridge and associated components:

- Table 6–1, " Actions Design Studio Construction"
- Table 6–2, " Cartridge Scan Parameter Groups Design Studio Construction"
- Table 6–3, " Import Processors Design Studio Construction"
- Table 6–4, " Equipment Discrepancy Detection Processors Design Studio Construction"
- Table 6–5, " Circuit Discrepancy Detection Processors Design Studio Construction"
- Table 6–6, " Discrepancy Resolution Processors Design Studio Construction"

Table 6–1 Actions Design Studio Construction

Action	Result Category	Scan Parameter Groups	Processors
Import from MSS	Device	See Table 6–2	See Table 6–3
Detect Equipment Discrepancies	Device	N/A	See Table 6–4
MSS Circuit Discrepancy Detection	Circuit	N/A	See Table 6–5
Resolve in MSS	Device	N/A	See Table 6–6

Table 6–2 Cartridge Scan Parameter Groups Design Studio Construction

Characteristic Name	Type	Description	UI Label
NetworkLocation	Text box	The network location. Enter either the common language location identifier (CLLI) code, or the coded location from MSS.	Network Location
Status	Drop down	List: All, Installed equipment, Equipment under maintenance The status of the root equipment.	Status
NodeNameQualifier	Drop down	Works in combination with the NodeName parameter to filter the imported nodes by name and qualifier.	Node Name Qualifier
NodeName	Text box	The device name or a list of device names.	Node Name
NodeId	Text box	The node ID or a list of node IDs.	Node Id
Scope	Drop down	List: ■ Equipment, STM Links, and Circuits ■ Equipment and STM Links only ■ Equipment only The scope of data to import from MSS.	Scope
RunMSSExtract	Drop down	Boolean to determine whether to run MSS incremental extraction procedure before the scan run.	Run MSS Extract

Table 6–3 Import Processors Design Studio Construction

Processor Name	Variable
Equipment DAOs Initializer	Input: N/A Output: ■ daoLocator The data access object (DAO) locator class that performs data lookup on the views.
Page Initializer	Input: daoLocator Output: ■ pageCountList An iterable list object for each page created. ■ pageSize The size of each page. ■ filterString The configurations entered in the Network Integrity UI for filtering the imported data.
Page Creator	Input: ■ daoLocator, pageSize, filterString ■ pageIndex An instance of the pageCountList iterable object. Output: ■ nodeNameList A list of imported node names corresponding to the filtered UI configurations.

Table 6–3 (Cont.) Import Processors Design Studio Construction

Processor Name	Variable
Node Collector	Input: daoLocator, nodeNameList Output: ■ nodesMapByNodeName A map of node names to a list of corresponding root equipment. ■ nodeSet The list of node names.
Device Modeler	Input: ■ nodesMapByNodeName ■ node An entry from the nodeSet object. Output: ■ collectedPortsUnderNode A list of ports belonging to the current node object. ■ logicalDevice A modeled logical device. ■ physicalDevice A modeled physical device. ■ rootEquipments A list of root equipment objects for the current node object.
Equipment Hierarchy Collector	Input: ■ daoLocator, logicalDevice, physicalDevice, nodesMapByNodeName ■ rootEquipment An entry from the current rootEquipments object. Output: ■ cardsToPortsMap A list mapping ports to their corresponding cards. ■ equipmentHierarchyDetails The equipment hierarchy details for the imported data.
Equipment Hierarchy Modeler	Input: cardsToPortsMap, equipmentHierarchyDetails, rootEquipment, collectedPortsUnderNode, physicalDevice, logicalDevice Output: N/A
Hierarchy Persister	Input: logicalDevice, physicalDevice Output: N/A

Table 6–3 (Cont.) Import Processors Design Studio Construction

Processor Name	Variable
STM Link Discoverer	Input: collectedPortsUnderNode, daoLocator, physicalDevice Output: ■ stmList A complete list of synchronous transport modules (STMs).
VC4 Circuit Discoverer	Input: stmList, daoLocator, physicalDevice Output: ■ igForCircuits The circuits inventory group. ■ vc4sForLops A list of VC4 circuits from which lower order pipes (LOPs) are collected.
VC3 VC12 LOP Discoverer	Input: daoLocator, igForCircuits, PhysicalDevice, vc4sForLops Output: N/A

Table 6–4 Equipment Discrepancy Detection Processors Design Studio Construction

Processor Name	Variable
Equipment Filters Initializer	Input: N/A Output: N/A
Discrepancy Detector	This processor is imported from the NetworkIntegritySDK cartridge project.
Discrepancy Filter	Input: N/A Output: N/A
Check Auto Resolution Selected	This processor is imported from the NetworkIntegritySDK cartridge project.
MSS Auto Resolve Selected Discrepancies	Input: autoResolutionManager Output: N/A
Identify Auto Resolving Discrepancies	This processor is imported from the NetworkIntegritySDK cartridge project.
Prepare Resolving Discrepancies	This processor is imported from the NetworkIntegritySDK cartridge project.

Table 6–5 Circuit Discrepancy Detection Processors Design Studio Construction

Processor Name	Variable
Circuit Discrepancy Name Filter Initializer	Input: N/A Output: isTopLevel

Table 6–5 (Cont.) Circuit Discrepancy Detection Processors Design Studio Construction

Processor Name	Variable
Missing Entity Filter Initializer	Input: isTopLevel Output: N/A This processor extends the Optical Circuit Discrepancy Detection action on the Optical Circuit Assimilation cartridge.
Partial Circuit Discrepancy Filter	Input: N/A Output: N/A
Discrepancy Detector	Input: N/A Output: N/A This processor extends the Base Detection cartridge.

Table 6–6 Discrepancy Resolution Processors Design Studio Construction

Processor Name	Variable
CORBA Property Initializer	Input: N/A Output: ■ corbaSeed A JavaBean that holds properties related to the CORBA cartridge, for CORBA connectivity. See *Network Integrity CORBA Cartridge Guide* for more information.
MSS CORBA Property Initializer	Input: corbaSeed Output: ■ corbaSeed ■ mssCORBAConnectionDetails The property group containing the MBean configuration required to establish CORBA connectivity with MSS. ■ mssEJBConnectionDetails The property group containing the MBean configuration required to establish EJB connectivity with MSS.
CORBA Connection Manager	Input: corbaSeed Output: ■ namingServer The Naming context for the MSS system. ■ orb The object request broker (ORB) instance.
Resolution Framework Initializer	Input: mssCORBAConnectionDetails, mssEJBConnectionDetails Output: ■ baseResolutionElement An instance of the data structure used to run resolution actions in MSS.
MSS Resolution Initializer	Input: mssCORBAConnectionDetails, mssEJBConnectionDetails, namingServer, orb, baseResolutionElement Output: mssCORBAConnectionDetails, mssEJBConnectionDetails
Resolution Framework Dispatcher	Input: mssCORBAConnectionDetails, mssEJBConnectionDetails, baseResolutionElement Output: mssCORBAConnectionDetails, mssEJBConnectionDetails

7

About Design Studio Extension

This chapter provides examples and explanations on how to extend certain aspects of the Oracle Communications Network Integrity MSS Integration cartridge using Oracle Communications Design Studio. See *Network Integrity Developer's Guide* for more information. See *Network Integrity Concepts* for guidelines and best practices for extending cartridges.

Importing Additional Information from MSS

The Import from MSS action imports equipment and circuit information from Oracle Communications MetaSolv Solution (MSS) from specific fields in the MSS materialized views.

You can extend the Import from MSS action to:

- Import additional information from other fields in the MSS Extract Schema.
- Import additional information from fields outside the MSS Extract Schema.

To import additional information from the MSS Extract Schema:

1. Identify the additional fields and the corresponding entities from the MSS Extract Schema to add to the scope of the Import from MSS action.
2. Determine the required API mapping and the corresponding TMF814 entities for the additional information. See *Network Integrity Optical TMF814 CORBA Cartridge Guide* for more information.
3. Identify the processor that models the additional TMF814 entities.
4. Extend the Import from MSS action to import and model the additional entities.

To import additional information from outside the MSS Extract Schema:

1. Identify the additional fields and the corresponding entities from MSS to add to the scope of the Import from MSS action.
2. Determine the MSS Extract Schema views for the additional information.
3. Use the custom fields in each of the MSS Extract Schema views to populate the additional fields for each entity.
4. Extend the MSS extract process to populate the additional information in the custom fields for each entity. See "Extending the MSS Extract Process" for more information.
5. Identify the additional fields and the corresponding entities from the MSS Extract Schema to add to the scope of the Import from MSS action.

6. Determine the required API mapping and the corresponding TMF814 entities for the additional information. See *Network Integrity Optical TMF814 CORBA Cartridge Guide* for more information.
7. Identify the processor that models the additional TMF814 entities.
8. Extend the Import from MSS action to import and model the additional entities.