Oracle® Communications Network Integrity

Cisco Router and Switch UIM Integration Cartridge Guide

Release 7.3.2

E66037-01

May 2016

ORACLE®

Oracle Communications Network Integrity Cisco Router and Switch UIM Integration Cartridge Guide, Release 7.3.2

E66037-01

Contents

4 About Cartridge Modeling

5 About Design Studio Construction

6 About Design Studio Extension

A Characteristic Attributes

B About the absRelativePosition Value

Preface

This guide describes the functionality and design of the Oracle Communications Network Integrity Cisco Router and Switch UIM Integration cartridge.

Audience

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

Developers should have a good working knowledge of simple network management protocol (SNMP) and its operations, specifications, Network Integrity, Unified Inventory Management (UIM), and Oracle Communications Design Studio for both UIM and Network Integrity.

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

- *Network Integrity Concepts*
- *Network Integrity Developer's Guide*
- *Network Integrity MIB-II SNMP Cartridge Guide*
- *Network Integrity Cisco Router and Switch SNMP Cartridge Guide*
- *Network Integrity UIM Integration Cartridge Guide*

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
E66037-01	May 2016	Initial release.

1

Overview

This chapter provides an overview of the Oracle Communications Network Integrity Cisco Router and Switch UIM Integration cartridge.

About the Cisco Router and Switch UIM Integration Cartridge

The Cisco Router and Switch UIM Integration cartridge is an extension of the Network Integrity Cisco Router and Switch SNMP cartridge, and provides Cisco-specific functionality, including:

- Generic management information bean (MIB) MIB-II logical discovery and modeling
- Cisco logical discovery and modeling of Frame Relay, asynchronous transfer mode (ATM), and virtual local area network (VLAN) media configurations
- Cisco physical discovery and modeling

This cartridge produces both logical and physical device hierarchies that represent a discovered device. The logical hierarchy includes a logical device, child interfaces, subinterfaces (collectively called interfaces), and device interface configurations. The physical hierarchy includes physical device, equipment, equipment holders, and physical ports. In addition, this cartridge creates associations between the physical and logical hierarchies. The first association is at the device level, between the physical device and the logical device, and the second association is at the interface level between physical ports and Interfaces. See *Network Integrity Cisco Router and Switch SNMP Cartridge Guide* for more information about the discovery action.

The Cisco Router and Switch UIM Integration cartridge also enables you to discover devices based on their CPU utilization by setting a threshold value (in percentage) in the Discover Enhanced Cisco SNMP discovery scan. See "Creating a Discover Enhanced Cisco SNMP Discovery Scan" for more information. See *Network Integrity Cisco Router and Switch SNMP Cartridge Guide* for more information about the CPU utilization-enabled discovery.

In addition to a discovery action, this cartridge provides discrepancy detection, discrepancy resolution, and import actions for integration with Unified Inventory Management.

The import action allows a logical and physical device tree in Oracle Communications Unified Inventory Management (UIM) to be imported to Network Integrity to compare objects with what is discovered.

Discrepancy detection provides the mechanism to allow a filtered comparison of logical and physical device trees between what is discovered and what is imported from UIM.

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

The discrepancy resolution action enables the discovered logical and physical device trees to be created and updated in UIM.

About Cartridge Dependencies

This section provides information on dependencies that the Cisco Router and Switch UIM Integration cartridge has on other entities.

Run-time Dependencies

For the Cisco Router and Switch UIM Integration cartridge to work at run time:

- You must deploy the Address_Handlers and Cisco_SNMP cartridges to Network Integrity
- UIM must be installed and be accessible to Network Integrity

The following components must be deployed to UIM:

- ora_ni_uim_cisco_device_sample
- ora_ni_uim_ocim
- ora_ni_uim_device
- UIM Integration web service

Design-Time Dependencies

To load this cartridge into Oracle Communications Design Studio, the following cartridges are required:

- Address_Handlers
- Cisco_Model
- Cisco_SNMP_Cartridge
- MIB_II_SNMP_Cartridge
- NetworkIntegritySDK
- ora_ni_uim_cisco_device_sample
- ora_ni_uim_device
- ora_ni_uim_ocim
- ora_uim_model
- UIM_Integration_Cartridge

Opening the Cartridge Files in Design Studio

To review and extend the Cisco Router and Switch UIM Integration cartridge, download the Oracle Communications Cisco Router and Switch UIM Integration cartridge software from the Oracle software delivery web site:

https://edelivery.oracle.com

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

- **\UIM_Cartridge_Projects**
- **\Network_Integrity_Cartridge_Projects**
- **\SNMP_MIBs**

The **UIM_Cartridge_Projects** and **Network_Integrity_Cartridge_Projects** folders contain the extensible Design Studio files.

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

Building and Deploying the Cartridge

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

2

About the Cartridge Components

This chapter provides information about the components that comprise the Oracle Communications Network Integrity Cisco Router and Switch UIM Integration cartridge.

The Cisco Router and Switch UIM Integration cartridge is composed of the following actions:

- Discover Enhanced Cisco SNMP Action
- Import Cisco from UIM Action
- Detect Enhanced Cisco Discrepancies Action
- Resolve Cisco in UIM Action

Discover Enhanced Cisco SNMP Action

The Discover Enhanced Cisco SNMP action discovers Cisco devices and provides a physical and logical hierarchical model of what is discovered. This discovery action also models the associations between the physical and logical hierarchies.

This discovery action extends the Discover Generic Cisco SNMP action (from the Cisco Router and Switch SNMP cartridge) and inherits all its processors. For more information about the inherited processors, see *Network Integrity Cisco Router and Switch SNMP Cartridge Guide*.

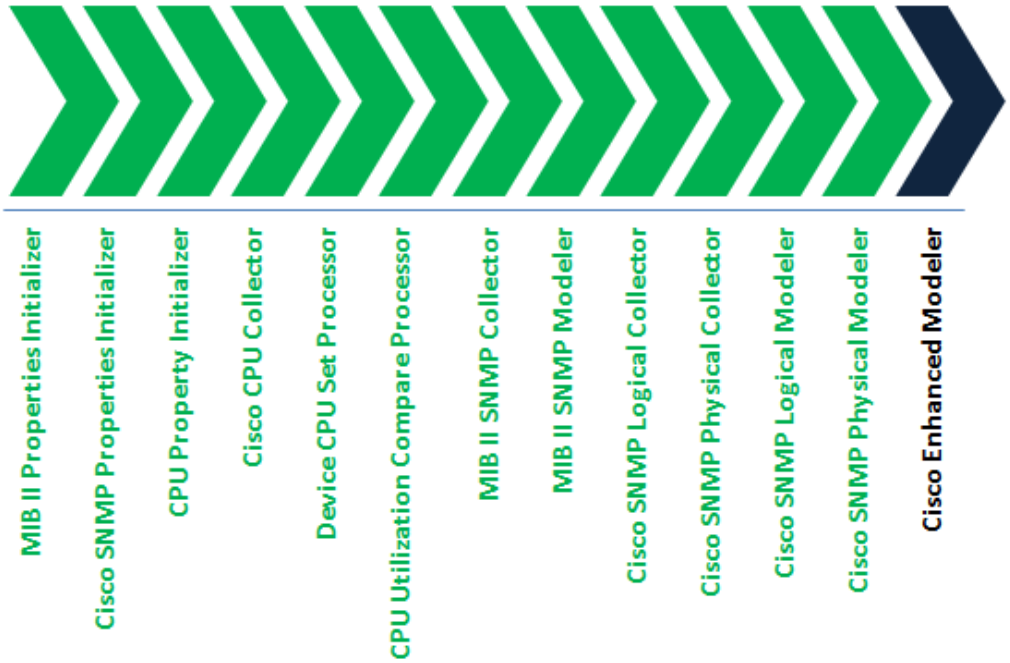
The Discover Enhanced Cisco SNMP action contains the following processors run in the following order:

1. MIB-II Properties Initializer (inherited)
2. Cisco SNMP Properties Initializer (inherited)
3. CPU Property Initializer (inherited)
4. Cisco CPU Collector (inherited)
5. Device CPU Set Processor (inherited)
6. CPU Utilization Compare Processor (inherited)
7. MIB-II SNMP Collector (inherited)
8. MIB-II SNMP Modeler (inherited)
9. Cisco SNMP Logical Collector (inherited)
10. Cisco SNMP Physical Collector (inherited)
11. Cisco SNMP Logical Modeler (inherited)

12. Cisco SNMP Physical Modeler (inherited)
13. Cisco Enhanced Modeler

Figure 2–1 illustrates the processor workflow of the Discover Enhanced Cisco SNMP action.

Figure 2–1 Discover Enhanced Cisco SNMP Action Processor Workflow



Cisco Enhanced Modeler

For discovering Cisco devices, Network Integrity implements the model that uses a generic specification to define its set of attributes. You can extend the generic specification to introduce new attributes. This model is built on the Information Model for Network Integrity, with modifications to support integration with Oracle Communications Unified Inventory Management (UIM).

Specifications are used to classify and categorize the entities that are found or produced. You can augment these entities with additional properties (attributes) using Design Studio. Specifications can be enhanced with characteristics to collect information about entities.

The Discover Enhanced Cisco SNMP action extends the Discover Generic Cisco SNMP action (from the Cisco Router and Switch SNMP cartridge) and inherits all its processors. The Discover Generic Cisco SNMP action creates generic entities and includes processors from MIB II Properties Initializer to Cisco SNMP Physical Modeler as shown in Figure 2–1.

In a typical Discover Enhanced Cisco SNMP Action processor workflow (illustrated in Figure 2–1), the processors run from MIB II Properties Initializer through Cisco SNMP Physical Modeler to retrieve matching entities (such as discovered physical device tree) and model a generic specification.

The Cisco Enhanced Modeler processor is used to programmatically walk the physical device tree and replace the generic specification with a specification that is consistent with Oracle Communications Unified Inventory Management (UIM).

Each physical entity has a field called modelName which is inspected for its value. The value is used as the specification name. The processor checks that the specification exists. If the specification exists, the processor applies that specification to the physical entity, overwriting the generic specification. If the specification is not found, the generic specification remains and the object does not become a UIM object. It is not staged for resolution to UIM. When you run a discovery scan and you no longer see generic specifications in your physical tree, you are ready to resolve the tree to UIM.

The advantage of extending the Discover Generic Cisco SNMP action is that you can use custom implementation to remodel the generic specification to be consistent with the specification of vendor-specific entities.

When a physical entity does not have any value for modelName and its name starts with *Artificial* (a result of model correction invocation), a lookup table (remodeler.properties) is used to determine what specification to apply. See *Cisco Router and Switch SNMP Cartridge Guide* for further information about model correction.

The lookup table allows you to specify (as an example) the key as "nativeEmsName and parent specification name" and value as the "specification name" to be applied. Model correction can produce many artificial entities, so using nativeEmsName and parent specification name in combination generally allows you to uniquely identify the intended entity in the hierarchy. A specification is applied to the physical entity when its nativeEmsName and parent specification name match.

Controlling modelName Content and Re-applied Specification

Systems Integrators can control the content of modelName, and so control the re-applied specification. The process is outlined as follows:

```
ciscoVendorType.properties sample
9 = cevModule
9.1 = cevModuleUnknownCard
9.2 = cevModuleCommonCards
9.3 = cevModuleC36xxType
9.4 = cevModuleVipPortAdapters
9.5 = cevModuleCpuType
9.5.1 = cevC7200Io1fe
9.5.2 = cevC7200Io
9.5.3 = cevCpuAS5300
9.5.7 = cevCpuRpm
9.5.8 = cevCpu2600
9.5.9 = cevCpu7200Npe300
9.5.10 = cevCpu1400
9.5.11 = cevCpu800
9.5.12 = cevCpuPSM1Gbps
```

The suffix of discoveredModelNumber can have up to three tokens delimited by a "."

If discoveredModelNumber contains 9.5.12, and the lookup table contains 9.5.12 cevCpuPSM1Gbps, this is yielded.

If 9.5.12 is not found, and 9.5 is found, cevModuleCpuType is yielded.

If 9.5.12 and 9.5 are not found, but 9 is found, cevModule is yielded.

If 9.5.12 and 9.5 and 9 are not found, then modelName is populated with "Unknown".

About remodeler.properties

The Discover Enhanced Cisco SNMP discovery action uses a generic remodeler to apply custom specifications to physical device results.

The process is data driven through a **remodeler.properties** file. Each line describes a remodeling rule in this format:

```
setspecification.entity type[.attribute name.attribute match criteria] =
[context/]specification
```

Where *entity type* is a type of entity, such as PhysicalDevice. The rule applies only to entities of this type.

The *attribute name.attribute match* criteria is optional. If present, the attribute with the specified name must match the attribute match criteria for the rule to apply. Match criteria has rudimentary substring matching where you can use the asterisk character to match one or more characters.

The *context* criteria is optional. If present, the context describes the specification that must be present on the parent entity for the rule to apply. For example, **cevContainerPowerSupplyBay/artificial6509PowerSupplyHolderCard** only matches an entity if the entity's parent has the specification **cevContainerPowerSupplyBay**. The context can describe the specifications for multiple ancestors, and not just the parent. In this case, specifications for ancestors are separated by "/".

specification is the name of the specification to apply if the entity matches the rule.

The remodeler walks the physical device tree. For each entity, it looks for a matching rule and applies that specification. Rules are processed in the order that they appear in the property file. Once a rule is applied, processing of that entity stops.

Note: Standard Java property file format must be followed. In particular, spaces and some other special characters on the left side of the equals must be escaped. For a full description of the Java property file format, consult the java doc for the load() method of the java.util.Properties class.

Supported Devices

The Cisco Router and Switch UIM Integration cartridge is built to support three devices for integration with UIM:

- cisco3640
- cisco7206
- cat6509

It does not support a full library of specifications for all hardware permutations found on these devices. If you try to discover these device instances, you may find certain hardware is not supported. To resolve to UIM, you may have to build up a library of specifications on Network Integrity and UIM to model these devices.

About Cisco Switches

Cisco Switches sometimes run CAT OS not IOS, and entityPhysicalVendorType is sparingly populated, so it cannot serve as the mechanism to identify hardware. To model Cisco switches with specifications staged for UIM, Systems Integrators must make extensive use of the remodeler.properties file, creating a mapping between discovered SNMP values and desired specifications.

The recommended SNMP variable is entityPhysicalDesc which is populated into the description field for Equipment. However, different versions of CAT OS may yield varying values for entityPhysicalDesc for the same hardware, so multiple entries may be required if dealing with various OS versions.

An example of the use of remodeler.properties is displayed in the following code. The class CiscoEnhancedModelerProcessorImpl contains the following code:

```
if (discoveredSysObjId.equals(SYSOBJECTID_6509)) {
       new CiscoRemodeler(getClass(),"remodeler.properties").remodelDevice(physicalDevice);
} else {
       new CiscoRemodeler().remodelDevice(physicalDevice);
}
```

In this example, remodeler.properties is invoked for the Cisco 6509 to handle artificial objects created by model correction.

To handle a particular Cisco switch, add another *else if* statement to identify it, and then invoke a custom remodeler.properties on it.

The *else* statement invokes default properties to use the field modelName as the source for identifying the specification to apply via the mapping table ciscoVendorTypesMap.

Tips for Creating Specifications

This section identifies some integration tips.

- Discovery fills in the field discoveredModelNumber from a SNMP variable entityPhysicalVendorType.
- discoveredModelNumber is interrogated, and the suffix is mapped to an entry in the table ciscoVendorTypesMap yielding modelName. modelName is thereby controlled using the content of the ciscoVendorTypesMap table.
- modelName as default is used to determine what specification to apply.
- If entity nativeEmsName starts with *Artificial*, and the device is Cisco 6509, remodeler.properties controls what specification to apply.
- If modelName is blank or unknown, you must either update ciscoVendorTypesMap or make use of remodeler.properties to model the device staging it for UIM.

If you discover a new device or a version of the three devices the cartridge supports out of the box, the specifications are applied and are visible in the entityType column in the physical tree in the Network Integrity UI.

If you see a generic specification in the entityType column in the physical tree as opposed to one that starts with "cev", then you must create a specification for this entity in both Network Integrity and UIM to fully support the device and have it staged for UIM.

For example, if you discover a new device, the modelName field is probably filled with a mapped value from ciscoVendorTypesMap, with a generic specification applied. Inspect modelName for each entity in the tree to ascertain what specification to create and what lineage (parent/child) must be set for UIM. When you run a discovery scan and you no longer see generic specifications in your physical tree, you are ready to resolve the tree to UIM.

Specifications in Network Integrity and UIM must exist, and the names must be equivalent. UIM has more requirements on specifications, in that additional properties must be set including:

- Parent child relationship and cardinality.

- Number of slots the Equipment card occupies.

Modeling in UIM for Discovery

The Network Integrity to UIM web service does not consider that equipment could be modeled in UIM such that they occupy multiple equipment holders (for example, a single equipment occupies two equipment holders).

In cases such as this, the web service operations treat the child of each equipment holder as having a unique equipment. Consequently, Network Integrity assumes that two unique equipment exist, when it fact there is only one.

This is a theoretical issue because, in Network Integrity, you must model equipment specifications in UIM in accordance with how Network Integrity discovers a device. When executing Cisco SNMP discovery, if an equipment instance occupies two equipment holders, this is not discoverable, and the device reports that an equipment instance occupies one equipment holder while the other equipment holder appears empty, even though the equipment physically occupies two equipment holder.

To modify discovery and modeling so that the true representation is rendered, custom handling is required in the cartridges and web service.

Import Cisco from UIM Action

The Import Cisco from UIM action imports logical and physical device trees from UIM for Cisco devices.

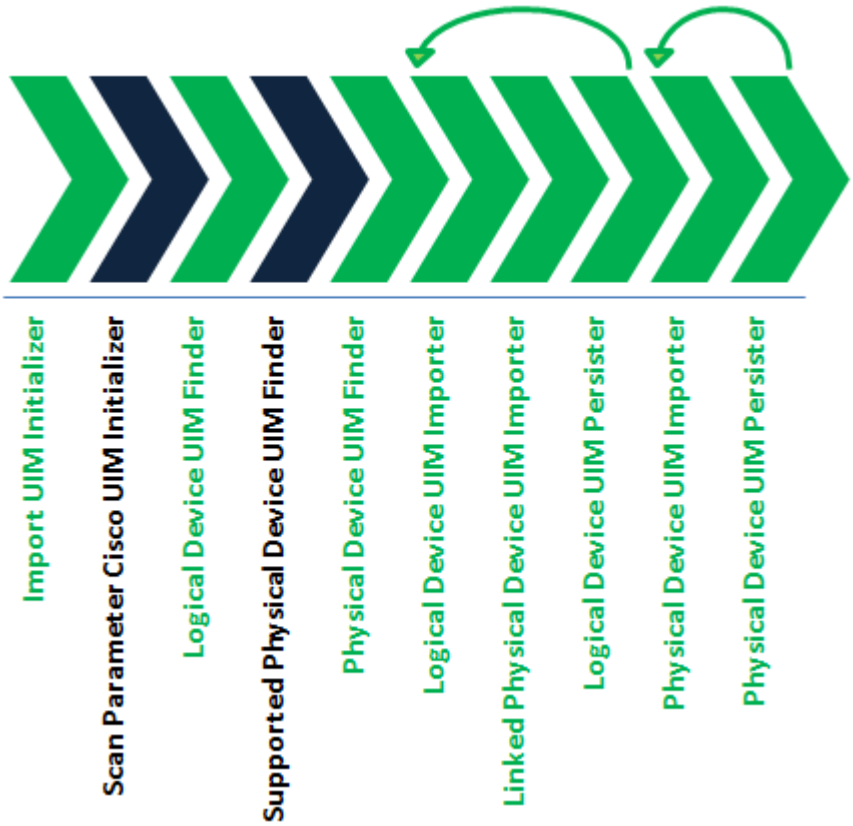
This import action extends the Abstract Import from UIM action (from the UIM Integration cartridge) and inherits all its processors. For information about the inherited processors, see *Network Integrity UIM Integration Cartridge Guide*.

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

1. Import UIM Initializer (inherited)
2. Scan parameter Cisco UIM Initializer
3. Logical Device UIM Finder (inherited)
4. Supported Physical Device UIM Finder
5. Physical Device UIM Finder (inherited)
6. Logical Device UIM Importer (inherited)
7. Linked Physical Device UIM Importer (inherited)
8. Logical Device UIM Persister (inherited)
9. Physical Device UIM Importer (inherited)
10. Physical Device UIM Persister (inherited)

Figure 2–2 illustrates the processor workflow of the Import Cisco from UIM action.

Figure 2–2 Import Cisco from UIM Action Processor Workflow



Scan parameter Cisco UIM Initializer

This processor initializes the scan parameter import filters.

Table 2–1 shows the filters for the Import Cisco from UIM action.

Table 2–1 Filters Used with Import Cisco from UIM Action

Filter	Pattern	Example
Name	Supports comma separated list for multiple values and wildcards	rot3640-11
Management IP Address	Supports comma separated list for multiple values and wildcards	10.10.10.10
Inventory State	▪ INSTALLED ▪ UNAVAILABLE	N/A
Network Location / Entity Code	Supports comma separated list for multiple values and wildcards	NYHQ1.D3

Supported Physical Device UIM Finder

This processor:

- Retrieves all the logical device IDs that match the filter criteria and have the *deviceGeneric* specification.
- Iterates over each ID to:
  - Retrieve the logical device tree and the associated physical device tree from UIM, while caching the ID of the physical device.

  - Persist the logical and physical device trees.
- Retrieves all the physical device IDs that match the filter criteria (if LogicalDeviceId is not set).
- Iterates over each physical ID, if it is not already cached:
  - Retrieve the physical device tree from UIM.
  - Persist the physical device trees.

This processor provides scan parameters that allow you to set filters when creating an import scan. The filters determine the set of entities included in the import scan.

Table 2–2 shows scan filter values used when configuring the filters in the base class.

Table 2–2 Import Scan Parameters (not set by the user)

Filter	Value
Query Physical Devices	true
Import Related Physical or Logical Device	true
Logical Device Specification	deviceGeneric

Detect Enhanced Cisco Discrepancies Action

The Detect Enhanced Cisco Discrepancies action detects discrepancies between Enhanced Cisco SNMP Discovery scan results and data imported from UIM.

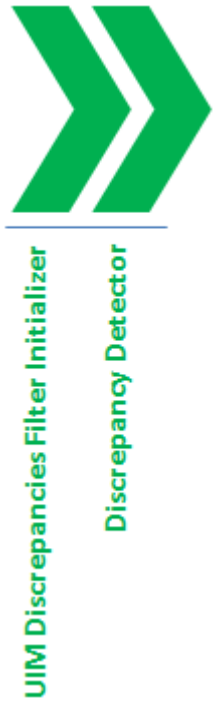
This discrepancy detection action extends the Abstract Detect UIM Discrepancies action (from the UIM Integration cartridge) and inherits all its processors. For information about the inherited processors, see *Network Integrity UIM Integration Cartridge Guide*.

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

1. UIM Discrepancies Filter Initializer (inherited)
2. Discrepancy Detector (inherited)

Figure 2–3 illustrates the processor workflow of the Detect Enhanced Cisco Discrepancies action.

Figure 2–3 Detect Enhanced Cisco Discrepancies Action Processor Workflow



Resolve Cisco in UIM Action

The Resolve Cisco in UIM action resolves discrepancies on logical and physical hierarchies and associations between logical and physical entities in UIM.

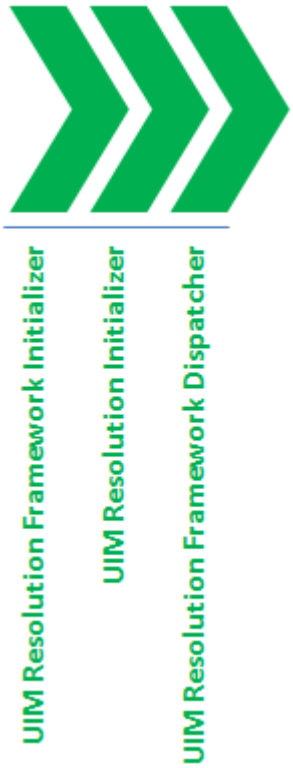
The discrepancy resolution action extends the Abstract Resolve in UIM action (from the UIM Integration cartridge) and inherits all its processors. For information about the inherited processors, see *Network Integrity UIM Integration Cartridge Guide*.

The Resolve Cisco in UIM action contains the following processors run in the following order:

1. UIM Resolution Framework Initializer (inherited)
2. UIM Resolution Initializer (inherited)
3. UIM Resolution Framework Dispatcher (inherited)

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

Figure 2–4 Resolve Cisco in UIM Action Processor Workflow



3

Using the Cartridge

This chapter provides instructions for using the Oracle Communications Network Integrity Cisco Router and Switch UIM Integration cartridge in Network Integrity.

Creating a Discover Enhanced Cisco SNMP Discovery Scan

The Discover Enhanced Cisco SNMP discovery scan discovers network information from IPv4, IPv6, FrameRelay, FrameRelayData, FrameRelayExtendedData, ATM Media, VirtualChannelLinkData, and ATMInterface networks.

To create a Discover Enhanced Cisco SNMP discovery scan:

1. Create a new scan.

   See the Network Integrity online Help for more information.

2. On the **General** tab, do the following:
   - From the **Scan Action** list, select **Discover Enhanced Cisco SNMP**.

     The **Scan Type** field displays **Discovery**.
   - In the **Scan Action Parameters** section, configure the scan with appropriate credentials and version details.
   - (Optional) To enable the scan to discover devices based on CPU utilization, do the following:
     - In the **Scan Action Parameters** section, from the **Select Parameter Group** list, select **CPU Utilization Parameters**.
     - In the **CPU Utilization %** field, specify a value between 1 to 99.
3. On the **Scope** tab, specify addresses.
4. Save and run the scan.

   The scan discovers and models logical and physical devices.

The scan creates device model for each logical and physical device. The physical device structure uses cat6509, cevChasis3640, and various references defined in the Discover Enhanced Cisco SNMP action. For more information, see "Model Collections".

Creating an Import Cisco from UIM Scan

The Import Cisco from UIM scan imports and models Cisco SNMP logical and physical devices from Oracle Communications Unified Inventory Management (UIM).

To create an Import Cisco from UIM scan:

1. Create a new scan.

   See the Network Integrity online Help for more information.

2. On the **General** tab of the Create Scan page, do the following:
   - From the **Scan Action** list, select **Import Cisco From UIM**.

     The **Scan Type** field displays **Import**.
   - In the Scan Action Parameters section, leave all the fields with their default values.

3. Save and run the scan.

   The scan creates logical and physical device models for each device. For more information, see "Model Collections".

Populating UIM with Discovered Data

This procedure describes steps to populate UIM with network data discovered by the Discover Enhanced Cisco SNMP discovery action.

To populate UIM with discovered network data:

1. Create a new scan.

   See the Network Integrity online Help for more information.

2. On the **General** tab of the Create Scan page, do the following:
   - From the **Scan Action** list, select **Discover Enhanced Cisco SNMP**.

     The **Scan Type** field displays **Discovery**.
   - Select **Detect Discrepancies**.
   - In the Scan Action Parameters area, make any necessary configurations.

3. Save the scan.
4. Run the discovery scan.

   The scan generates Entity+ discrepancies for each discovered device.

5. Right-click on the discrepancies you want to populate into UIM and select **Correct in UIM**.
6. Click **Submit**.
7. Verify that UIM is populated with the discovered data.
8. In Network Integrity, configure and run an import scan.

   The import scan imports data from UIM.

9. Run the discovery scan again.

   The scan does not detect any discrepancies and UIM should now contain discovered data.

4

About Cartridge Modeling

This chapter provides information on modeling the Oracle Communications Network Integrity Cisco Router and Switch UIM Integration cartridge.

For information about the following, see *Network Integrity Cisco Router and Switch SNMP Cartridge Guide*:

- Cisco UIM Integration cartridge UML Representation
- Hierarchy Mapping
- Oracle Communications Information Model Information
- Field Mapping
- Logical Mapping
- Physical Mapping
- Model Correction

Shared Specifications

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

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

Oracle Communications Design Studio features are used to tag certain characteristics as specific to UIM. These characteristics appear on specifications deployed to UIM, but they do not appear on specifications deployed to Network Integrity.

This cartridge references **ora_ni_uim_device** for the shared physicalDeviceGeneric and equipmentGeneric specifications.

The following tables identify which characteristics on the shared specifications are specific to UIM.

deviceGeneric Characteristics

Table 4–1 shows deviceGeneric characteristics.

Table 4–1 deviceGeneric Characteristics

Characteristic Name	Specific to UIM
mgmtIpAddress	No
sysObjectId	No
nativeEmsAdminServiceState	Yes
nativeEmsServiceState	Yes
nativeEmsName	Yes

interfaceGeneric Characteristics

Table 4–2 shows interfaceGeneric characteristics.

Table 4–2 interfaceGeneric Characteristics

Characteristic Name	Specific to UIM
ifAlias	No
ifName	No
ifPromiscuousMode	No
ifType	Yes
minSpeed	Yes
maxSpeed	Yes
nominalSpeed	Yes
physicalAddress	Yes
physicalLocation	Yes
nativeEmsConnectorPresent	Yes
mtuCurrent	Yes
mtuSupported	Yes
nativeEmsAdminServiceState	Yes
nativeEmsServiceState	Yes
nativeEmsName	Yes

physicalDeviceGeneric Characteristics

Table 4–3 shows physicalDeviceGeneric characteristics.

Table 4–3 physicalDeviceGeneric Characteristics

Characteristic Name	Specific to UIM
mgmtIpAddress	No
modelName	No
discoveredModelNumber	No
discoveredPartNumber	No
discoveredVendorName	No
hardwareRev	No

Table 4–3 (Cont.) physicalDeviceGeneric Characteristics

Characteristic Name	Specific to UIM
softwareRev	No
nativeEmsName	Yes

equipmentGeneric Characteristics

Table 4–4 shows equipmentGeneric characteristics.

Table 4–4 equipmentGeneric Characteristics

Characteristic Name	Specific to UIM
modelName	No
discoveredModelNumber	No
discoveredPartNumber	No
discoveredVendorName	No
hardwareRev	No
nativeEmsName	Yes

equipmentHolderGeneric Characteristics

Table 4–5 shows equipmentHolderGeneric characteristics.

Table 4–5 equipmentHolderGeneric Characteristics

Characteristic Name	Specific to UIM
modelName	No
discoveredModelNumber	No
discoveredPartNumber	No
discoveredVendorName	No
hardwareRev	No
nativeEmsName	Yes

physicalPortGeneric Characteristics

Table 4–6 shows physicalPortGeneric characteristics.

Table 4–6 physicalPortGeneric Characteristics

Characteristic Name	Specific to UIM
modelName	No
discoveredModelNumber	No
discoveredPartNumber	No
discoveredVendorName	No
hardwareRev	No
nativeEmsName	Yes

5

About Design Studio Construction

This chapter provides information on the composition of the Oracle Communications Network Integrity Cisco Router and Switch UIM Integration cartridge from the Oracle Communications Design Studio perspective.

Model Collections

See *Network Integrity Cisco Router and Switch SNMP Cartridge Guide* for information about the MIB-II and Cisco model collections.

Specifications

Table 5–1 shows the specifications included in the Cisco Router and Switch UIM Integration cartridge.

Table 5–1 Cisco Router and Switch UIM Integration Cartridge Specifications

Specification	Information Model Entity Type
cisco3640	Physical Device
cevChassis3640	Equipment
cat6509	Physical Device
cevChassisCat6509	Equipment
artificial6509ContainerGbic	Equipment
artificial6509MSFCCard	Equipment
artificial6509PowerSupplyHolderCard	Equipment
artificial6509PowerSupplySlot	Equipment Holder
cisco7206VXR	Physical Device
cevChassis7206Vxr	Equipment
cevBackplaneCat6000	Equipment
cevBackplaneCat6500	Equipment
cevCat6kWsc6000cl	Equipment
cevCat6kWsc6kvtt	Equipment
cevCat6kWsf6kPfc	Equipment
cevCat6kWsx6248Rj45	Equipment
cevCat6kWsx6348Rj45	Equipment

Table 5–1 (Cont.) Cisco Router and Switch UIM Integration Cartridge Specifications

Specification	Information Model Entity Type
cevCat6kWsxSup1a2ge	Equipment
cevContainerClock	Equipment Holder
cevContainerDaughterCard	Equipment Holder
cevContainerFanTraySlot	Equipment Holder
cevContainerGbic	Equipment Holder
cevContainerPowerSupplyBay	Equipment Holder
cevContainerSlot	Equipment Holder
cevContainerVtt	Equipment Holder
cevCpu7200Npe300	Equipment
cevCpuCat6kMsfc	Equipment
cevCpuCat6kWsxSup1a2ge	Equipment
cevFanWSC6k9SlotFan	Equipment
cevModuleCat6000Type	Equipment
cevModuleUnknownCard	Equipment
cevModuleVipPortAdapters	Equipment
cevPa8e	Equipment
cevPaAtmdxMmOc3	Equipment
cevPaE3MuxCbr120e1	Equipment
cevPmCpm2e2w	Equipment
cevPmM4t	Equipment
cevPortAMDP2	Physical Port
cevPortDCUATMPort	Physical Port
cevPortFe	Physical Port
cevPortFEIP	Physical Port
cevPortGigBaseLH	Physical Port
cevPortMueslix	Physical Port
cevPortQuiccSerial	Physical Port
cevPortUnknown	Physical Port
cevPowerSupplyAC1360W	Equipment
cevPowerSupplyC7200AC	Equipment
cevSensorClock	Equipment
cevSensorFanTrayStatus	Equipment
cevSensorModuleDeviceTemp	Equipment
cevSensorModuleInletTemp	Equipment
cevSensorModuleOutletTemp	Equipment
cevSensorModulePowerOutputFail	Equipment
cevSensorPSFan	Equipment

Table 5–1 (Cont.) Cisco Router and Switch UIM Integration Cartridge Specifications

Specification	Information Model Entity Type
cevSensorPSInput	Equipment
cevSensorPSOutput	Equipment
cevSensorVtt	Equipment
cevWicSerial1t	Equipment

Logical Specification Lineage

See *Network Integrity Cisco Router and Switch SNMP Cartridge Guide* for Logical Specification Lineage.

Physical Specification Lineage

This section displays the specification lineages for the three out-of-the-box supported devices integrated with Oracle Communications Unified Inventory Management (UIM):

- cisco3640
- cisco7206VXR
- cat6509

cisco3640

Example 5–1 shows a specification lineage for cisco3640. This lineage shows the intended relationship between specifications.

Example 5–1 cisco3640 Specification Lineage

```
cisco3640
   cevChassis3640
      cevContainerSlot
         cevPmCpm2e2w
            cevPortAMDP2
            cevContainerDaughterCard
               cevWicSerial1t
                  cevPortQuiccSerial
         cevPmM4t
            cevPortMueslix
```

cisco7206VXR

Example 5–2 shows a specification lineage for cisco7206VXR. This lineage shows the intended relationship between specifications.

Example 5–2 cisco7206VXR Specification Lineage

```
cisco7206VXR
    cevChassis7206Vxr
        cevCpu7200Npe300
        cevContainerSlot
           cevModuleUnknownCard
              cevPortFEIP
           cevPaAtmdxMmOc3
```

```
            cevPortUnknown
          cevModuleVipPortAdapters
           cevPortMueslix
          cevPaE3MuxCbr120e1
           cevPortUnknown
           cevPortDCUATMPort
          cevPowerSupplyC7200AC
          cevPa8e
           cevPortAMDP2
```

cat6509

Example 5–3 shows a specification lineage for cat6509. This lineage shows the intended relationship between specifications.

Example 5–3 cat6509 Specification Lineage

```
cat6509
   cevChassisCat6509
      cevBackplaneCat6000
         cevContainerVtt
            cevCat6kWsc6kvtt
               cevSensorVtt
         cevSensorVtt
         cevContainerClock
            cevCat6kWsc6000cl
               cevSensorClock
         cevSensorClock
      cevBackplaneCat6500
         cevContainerVtt
            cevCat6kWsc6kvtt
               cevSensorVtt
         cevSensorVtt
         cevContainerClock
             cevCat6kWsc6000cl
                cevSensorClock
         cevSensorClock

   cevContainerFanTraySlot
      cevFanWSC6k9SlotFan
      cevSensorFanTrayStatus
   cevContainerPowerSupplyBay
      artificial6509PowerSupplyHolderCard
         artificial6509PowerSupplySlot
            cevPowerSupplyAC1360W
               cevSensorPSFan
               cevSensorPSInput
               cevSensorPSOutput
   cevContainerSlot
      cevCat6kWsxSup1a2ge
      cevCpuCat6kWsxSup1a2ge
         cevSensorModuleOutletTemp
         cevSensorModulePowerOutputFail
         cevSensorModuleInletTemp
         cevSensorModuleDeviceTemp
         cevContainerDaughterCard
            cevWicSerial1t
               cevPortQuiccSerial
            cevCat6kWsf6kPfc
               cevSensorModuleOutletTemp
```

```
                  cevSensorModuleInletTemp
               artificial6509MSFCCard
                  cevModuleCat6000Type
                     cevSensorModuleOutletTemp
                     cevSensorModuleInletTemp
                  cevCpuCat6kMsfc
      artificial6509ContainerGbic
         cevPortGigBaseLH
cevCat6kWsx6248Rj45
   cevSensorModulePowerOutputFail
   cevSensorModuleInletTemp
   cevSensorModuleDeviceTemp
   cevPortFe
cevCat6kWsx6348Rj45
   cevSensorModulePowerOutputFail
   cevSensorModuleInletTemp
   cevSensorModuleDeviceTemp
   cevPortFe
```

Discovery Action

Table 5–2 shows details about the Discover Enhanced Cisco SNMP discovery action.

Table 5–2 Discover Enhanced Cisco SNMP Action

Result Category	Address Handler	Scan Parameters	Model	Processors
Device	IPAddressHandler	■ version ■ port ■ snmpReadCommunity ■ snmpTimeout ■ snmpRetries ■ username ■ contextName ■ authProtocol ■ authPassword ■ privacyProtocal ■ privacyPassword	■ MIB-II Model ■ Cisco Model ■ Cisco UIM Model	■ MIB-II Properties Initializer ■ Cisco SNMP Properties Initializer ■ MIB-II SNMP Collector ■ MIB-II SNMP Modeler ■ Cisco SNMP Logical Collector ■ Cisco SNMP Physical Collector ■ Cisco SNMP Logical Modeler ■ Cisco SNMP Physical Modeler ■ Cisco Enhanced Modeler

Discovery Processors

Table 5–3 shows information about the processors of the Discover Enhanced Cisco SNMP action.

Table 5–3 Discover Enhanced Cisco SNMP Action Processors

Processor Name	Variable
MIB-II Properties Initializer	Input: N/A Output: ■ snmpIfTypeMap Property map containing listing of ifTypes to string name. ■ snmpVendorNameMap Property map containing listing of sysObjectId suffixes to vendorName.
Cisco SNMP Properties Initializer	Input: N/A Output: ■ ciscoProductsMap A mapping from the vendor specific portion of the sysObjectId to Cisco Device model name. ■ ciscoVendorNumbers A list containing supported Cisco vendor numbers. ■ ciscoVendorTypesMap A mapping from entPhysicalVendorType to Cisco equipment part name.
CPU Property Initializer	Input: N/A Output: ■ cpuProperties This class is used to initialize and set the default CPU value for the device.
Cisco CPU Collector	Input: N/A Output: ■ cpmCPUTotal5secRev ■ cpmCPUTotal1minRev ■ cpmCPUTotal5minRev
Device CPU Set Processor	Input: N/A ■ ciscoCPUCollectorResponseDocument SNMP discovered data produced by the Cisco CPU Collector. ■ cpuProperties This class is used to initialize and set the default CPU value for the device. Output: N/A
CPU Utilization Compare Processor	Input: N/A ■ cpuProperties This class is used to initialize and set the default CPU value for the device. Output: N/A
MIB-II SNMP Collector	Input: N/A Output: ■ mibiisnmpCollectorResponseDocument (implicit) Polled SNMP data. For information about poll lists, see *Cisco Router and Switch SNMP Cartridge Guide*.

Table 5–3 (Cont.) Discover Enhanced Cisco SNMP Action Processors

Processor Name	Variable
MIB-II SNMP Modeler	Input: ■ mibiisnmpCollectorResponseDocument SNMP discovered data produced by the MIB-II SNMP Collector. ■ snmpIfTypeMap Property map containing listing of ifTypes to string name. Output: ■ deviceInterfaceMap A map that contains interfaces with IfIndex as key. ■ logicalDevice This is the logical device that was created in the MIB-II Modeler.
Cisco SNMP Logical Collector	Input: N/A Output: ■ ciscoSNMPLogicalCollectorResponseDocument (implicit) Polled SNMP data. For information about poll lists, see *Cisco Router and Switch SNMP Cartridge Guide*.
Cisco SNMP Physical Collector	Input: N/A Output: ■ ciscoSNMPLogicalCollectorResponseDocument (implicit) Polled SNMP data. For information about poll lists, see *Cisco Router and Switch SNMP Cartridge Guide*.

Table 5–3 (Cont.) Discover Enhanced Cisco SNMP Action Processors

Processor Name	Variable
Cisco SNMP Logical Modeler	Input: ■ ciscoSNMPLogicalCollectorResponseDocument SNMP discovered data produced by the Cisco SNMP Logical Collector. ■ ciscoVendorNumbers A list containing supported Cisco vendor numbers. ■ deviceInterfaceMap A map that contains interfaces with IfIndex as key. ■ logicalDevice This is the logical device that was created in the MIB-II Modeler. ■ mibiisnmpCollectorResponseDocument SNMP discovered data produced by the MIB-II SNMP Collector. Output: ■ ciscoLogicalDevice This is the logical device that was created in the Cisco SNMP Logical Modeler.
Cisco SNMP Physical Modeler	Input: ■ ciscoSNMPPhysicalCollectorResponseDocument SNMP discovered data produced by the Cisco SNMP Physical Collector. ■ ciscoProductsMap A mapping from the vendor specific portion of the sysObjectId to Cisco Device model name. ■ ciscoVendorTypesMap A mapping from entPhysicalVendorType to Cisco equipment part name. ■ logicalDevice This is the logical device that was created in the MIB-II Modeler. ■ mibiisnmpCollectorResponseDocument SNMP discovered data produced by the MIB-II SNMP Collector. ■ snmpVendorNameMap Property map containing listing of sysObjectId suffixes to vendorName. Output: N/A
Cisco Enhanced Modeler	Input: ■ logicalDevice This is the logical device that was created in the MIB-II Modeler. Output: N/A

Import Cisco from UIM Action

Table 5–4 shows details about the Import Cisco from UIM import action.

Table 5–4 Import Cisco from UIM Action

Result Category	Scan Parameters	Model	Processors
Device	▪ name ▪ mgmtIPAddress ▪ adminState ▪ networkLocationEntityCode	▪ MIB-II Model ▪ Cisco Model ▪ Cisco UIM Model	▪ This import action extends the Abstract Import from UIM action included in the UIM Integration cartridge. See *Network Integrity UIM Integration Cartridge Guide* for information about the processors in this action. ▪ Scan Parameter Cisco UIM Initializer ▪ Supported Physical Device UIM Finder

Detect Enhanced Cisco Discrepancies Action

Table 5–5 shows details about the Detect Enhanced Cisco Discrepancies action.

Table 5–5 Detect Enhanced Cisco Discrepancies Action

Result Category	Results Source	Scan Parameters	Model	Processors
Device	Discover Enhanced Cisco SNMP action	N/A	N/A	▪ This action extends the Abstract Detect UIM Discrepancies action included in the UIM Integration cartridge. See *Network Integrity UIM Integration Cartridge Guide*. ▪ Detect Enhanced Cisco Discrepancies

Resolve Cisco in UIM Action

Table 5–6 shows details about the Resolve Cisco in UIM discrepancy resolution action.

Table 5–6 Resolve Cisco in UIM Action

Result Category	Results Source	Result Category	Processors
Device	Discover Enhanced Cisco SNMP action	All	This action extends the Abstract Resolve in UIM action included in the Network Integrity UIM Integration cartridge. See *Network Integrity UIM Integration Cartridge Guide*.

6

About Design Studio Extension

This chapter provides scenarios for the extensibility of Oracle Communications Network Integrity using Oracle Communications Design Studio.

Adding a New Vendor

For this example, Cisco introduces a new vendor type to represent a new equipment part. This cartridge defines a map called ciscoVendorTypesMap that contains the equipment part name indexed by the vendor type number, which is a portion of the entPhysicalVendorType OID. The Cisco SNMP Properties Initializer processor produces this map and makes it available for other processors. To update the map to include a new vendor type number and corresponding equipment part name, you can extend the Discover Generic Cisco SNMP action and add a new Cisco SNMP Post Properties Initializer processor. This initializer processor takes as input the map (for example, ciscoVendorTypesMap) produced by the Cisco SNMP Properties Initializer processor. The implementation can then update the map.

For more details regarding extensibility, see *Network Integrity Developer's Guide*.

A

Characteristic Attributes

This appendix describes how to determine the values of the modelName and discoveredVendorName fields.

Determining the Value of the modelName Field

You determine the value of the modelName field for PhysicalDevice using the following algorithm:

- The input is the sysObjectId value in its raw form (for example, .1.3.6.1.4.1.9.1.110) stored in the discoveredModelNumber field.
- It then parses the ninth digit from sysObjectId (for example, 110).
- It then uses the parsed value (for example, 110) as the key to look up the model name from the ciscoProductsMap that is output from the Cisco SNMP Properties Initializer.
- If the key does not exist, discoveredVendorName is set to **Unknown** *(x)* where *x* is the key; for example, **Unknown (110)**.

You determine the value of the modelName field for Equipment, EquipmentHolder, and PhysicalPort using the following algorithm:

- The input is the entPhysicalVendorType OID value in its raw form (for example, .1.3.6.1.4.1.9.12.3.1.9.3.2) stored in the discoveredModelNumber field. This value is in the form of .1.3.6.1.4.1.9.12.3.1.*vvvv.xxxx.yyyy.n.n*….. where *vvvv.xxxx.yyyy* is the vendor type number.
- It then parses out the vendor type number from the entPhysicalVendorType value (for example, 9.3.2).
- It then uses the vendor type number (for example, 9.3.2) as the key to look up the model name from the ciscoVendorTypesMap that is output from the Cisco SNMP Properties Initializer.
- If the key does not exist, the cartridge trims the last digit and try again (for example, from 9.3.2 to 9.3). If the key still does not exist, it trims once more (for example, to 9) and try again. If this fails as well, discoveredVendorName is set to **Unknown** *(x)* where *x* is the original key; for example, 9.3.2.

Determining the Value of the discoveredVendorName Field

You determine the value of the discoveredVendorName field using the following algorithm:

- The input is the sysObjectId value in its raw form (for example, .1.3.6.1.4.1.9.1.110) stored in the discoveredModelNumber field.
- It then parses the seventh digit from sysObjectId (for example, 9).
- It then uses the vendor number (for example, 9) as the key to look up the vendor name from the snmpVendorNameMap that is output from the MIB II Properties Initializer.
- If the key does not exist, discoveredVendorName is set to **Unknown** *(x)* where *x* is the key; for example, **Unknown (9)**.

B

About the absRelativePosition Value

This appendix explains how to generate the absRelativePosition variable value.

Generating the absRelativePosition Value

The field contains a variable called absRelativePosition. absRelativePosition is used to generate a unique value for an entity in a given tree. absRelativePosition is a programmatically generated value and is composed of a prefix and suffix. The suffix is always derived from entPhysicalParentRelPos.

The following is an example of a physical device tree:

PD

E1

PP1

E2

PP2

The prefix of each entity is derived from the absolute relative position from the root; for example:

PD takes the prefix 0

The child E1 takes the prefix 0:0

The child E2 takes the prefix 0:1, uniquely identifying itself from its sibling.

PP1 takes the prefix 0:0:0

PP2 takes the prefix 0:1:0

Some devices (due to a device reporting error) show that multiple equipment holders or physical ports have entPhysicalParentRelPos value that occupy the same relative position within the same parent.

At a high level, the algorithm works by resolving conflicts by increasing the relative position of the subsequent duplicates by 1. This is best described using an example.

Table B–1 shows the applicable SNMP attributes that are used in determining the correct relative position.

Table B–1 SNMP Attributes Used to Determine Correct Relative Position

Index	entPhysicalDesr	entPhysicalParentRelPos	entPhysicalContainedIn
1	3640 chassis, Hw Serial#: 621974280, Hw Revision: 00	-1	0

Table B–1 (Cont.) SNMP Attributes Used to Determine Correct Relative Position

Index	entPhysicalDesr	entPhysicalParentRelPos	entPhysicalContainedIn
2	3640 Chassis Slot	0	1
3	Ethernet/WAN	0	2
12	AmdP2	0	3
13	AmdP2	0	3
14	AmdP2	1	3
15	AmdP2	1	3

The column headings in Table B–1 have the following meaning:

- Index: a numeric value used to represent a physical entity and must be unique.
- entPhysicalDesr: a string description of the physical entity.
- entPhysicalParentRelPos: the relative position within the parent.
- entPhysicalContainedIn: the Index of the entity's parent denoting the current entity as a child. 0 indicates the root of the physical entity tree.

From Table B–1, notice that AmdP2 at index 12 and 13, and AmdP2 at index 14 and 15 have the same entPhysicalParentRelPos values (that is, 0 and 1 respectively) within the parent, which is Ethernet/WAN.

- Index 12 and 13 would both generate the name AmdP2::0.
- Index 14 and 15 would both generate the name AmdP2::1.

To correct this, the algorithm briefly described above is executed as follows:

- Index 12 is processed first and would generate the name "AmdP2::0".
- Index 13 is processed and it is determined that the entPhysicalParentRelPos is a duplicate. The cartridge increments the value by 1 and generates the name "AmdP2::1". It also flags this entity's position as artificially generated.
- Index 14 is processed and it is determined that an entity already exists with entPhysicalParentRelPos 1, however it was artificially generated. Therefore, Index 14 gets the name "AmdP2::1" and Index 13 gets name "AmdP2::2".
- Index 15 is processed and it is determine that the entPhysicalParentRelPos is a duplicate. The cartridge increments the last artificially generated value by 1 thereby creating name "AmdP2::3".

  The result is as shown in Table B–2.

Table B–2 End Result

Index	entPhysicalDesr	Name
12	AmdP2	AmdP2::0
13	AmdP2	AmdP2::2
14	AmdP2	AmdP2::1
15	AmdP2	AmdP2::3