CA Performance Management Data Aggregator

Power User Certification Guide

2.4.1

ca technologies

CA Technologies Product References

This document references the following CA Technologies products:

- CA Performance Management Data Aggregator (Data Aggregator)
- Data Collector
- CA Performance Center

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

Contents

Chapter 1: Introduction

This section contains the following topics:

About this Guide

Data Aggregator provides a vendor self-certification interface that lets you expand CA Performance Management monitoring support. Self-certification lets you supplement the existing metric families in Data Aggregator with new or custom components and metrics from vendor MIBs.

You can create a vendor certification or modify existing device or component support. This guide discusses how to add new technology support with custom metric families, along with the custom vendor support.

This guide addresses advanced scenarios for creating custom vendor certifications and metric families. A basic knowledge of XML and schema files is required.

Note: Data Aggregator provides basic and advanced methods for creating custom vendor certifications and metric families. The basic method is a simpler process, consisting of adding vendor support for existing supported technologies (metric families), using the user interface. This method meets the requirements of many users. However, the advanced method is based on the factory certification format and exposes a complete set of capabilities. This guide explains the advanced certification method. For information about the basic certification method, see the *Data Aggregator Basic Self-Certification Guide*.

This guide follows a specific example throughout. As an administrator, you want to monitor Frame Relay permanent virtual circuits (PVCs). Out-of-the-box, Data Aggregator does not have support to monitor Frame Relays PVCs, but you can use self-certification to generate this support. In this guide, we provide the steps to define the component, create a custom metric family, and define a custom vendor certification to monitor Frame Relay permanent virtual circuits (PVCs).

How Device Support Works

Data Aggregator supports a vendor device using metric families and vendor certifications. Working together, these components determine how Data Aggregator collects configuration and operational metrics for a device. Understanding how device support works in Data Aggregator helps you determine whether your devices are properly supported in Data Aggregator. If they are not, understanding this process can help you adjust your settings to get the results you need.

Note: If needed, you can customize a metric family, vendor certification, or both to add support for a vendor device.

Data Aggregator supports devices using the following configuration features:

1. **Discovery Profile**—Determines which items in your environment Data Aggregator discovers, typically based on a range of IP addresses. The discovery process identifies the "type" for each item it finds.
2. **Device Collections**—Organizes your inventory into groups of related items. Based on the item type and IP address, items are automatically added to a device collection.
3. **Monitoring Profile**—Controls the polling rate for a device collection and determines which metric families to poll. Monitoring profiles can poll one or more metric families.

   Note: To ensure that your system is not overloaded with polling traffic, use monitoring profiles to adjust the polling rate for different sets of metrics.
4. **Metric Family**—Controls which metrics are gathered for a monitoring profile. Metric families are associated with one or more vendor certifications, which are listed in priority order.

   Note: Reuse metric families in your monitoring profiles to help ensure consistent data reporting.
5. **Vendor Certification**—Maps attributes from a vendor MIB to the metrics in a metric family. Also determines how metrics that are collected from an item are formatted for use in the CA Performance Center UI and reports. Metrics that are provided for an item can vary, depending on the item vendor . Mapping these values ensures that the metric values are reported consistently, regardless of the vendor. Multiple vendor certifications can be associated with a single metric family. In such cases, Data Aggregator maps metric values using a ranked list of vendor certifications. Data Aggregator calculates a metric value using the highest-priority vendor certification that matches the polled item.

   Note: MIBs, such as SNMP MIBs, can be imported into the system and compiled as part of building a vendor certification.

Example: Support for a Router Device

When running your discovery profile, Data Aggregator finds and identifies an item as a router. The router managed item is automatically added to the All Routers device collection. This device collection is associated with the Routers monitoring profile, which uses the CPU and Memory metric families to discover the CPU and Memory components on the device. These metric families also determine the vendor certification to use when calculating the metric values for these components. Based on this monitoring profile, Data Aggregator polls your router every 5 minutes for the metric data in these metric families. For example, the CPU metric family includes CPU idle utilization, CPU system utilization, and CPU nice utilization. Finally, the vendor certifications that are associated with a metric family determine how to calculate and format the raw metric data consistently. Data Aggregator stores the collected metric data for your router, which CA Performance Center uses in the UI and reports.

Device Support through Self-Certification

Data Aggregator supports common vendor devices using predefined *certifications*. Certifications specify how to collect configuration and operational metrics for a device. Data Aggregator uses the following methods for certification:

- Metric families
- Vendor certifications

How do you collect data when Data Aggregator does *not* provide a predefined certification for your device? You can *self-certify* support for a device.

Note: Metric families and vendor certifications are global (that is, *not* tenant aware). For more information about tenants, see the *CA Performance Center Administrator Guide*.

The *self-certification* support in Data Aggregator lets you create a custom vendor certification, a custom metric family, or both. Determine which method you need, as follows:

- **Vendor certification only**—A set of metrics you want are polled by default, but Data Aggregator does not support your device vendor MIB for these metrics. For example, Data Aggregator provides a CPU metric family to collect data such as CPU utilization. However, you want to gather CPU data for a server that Bargain Server Company manufactures. Using the MIB that the manufacturer provided, you can create a custom vendor certification for your server CPUs.
- **Metric family only**—Support for your device vendor MIB is included by default, but some metrics that the MIB supports are not polled. For example, your vendor MIB supports metrics for processes, but Data Aggregator does not provide a "Processes" metric family to gather that metric data.
- **Both methods**—Create both a metric family and vendor certification when Data Aggregator does not provide support for a device vendor MIB or its metrics.

Custom Certification Scenarios

Out of the box, CA Performance Management supports the common vendors, metrics, and components in your network infrastructure. The Technology Certification Portal lists out-of-the-box certifications by Data Aggregator version, vendor certification, and metric families:

http://serviceassurance.ca.com/im/ http://serviceassurance.ca.com/im/

However, you can maximize your infrastructure management power by creating custom CA Performance Management certifications. Customize when you need CA Performance Management support for:

- A new technology:
  - Define a new component.
  - Create a custom metric family.
- A new metric for an existing technology:
  - Create a custom metric family from a predefined (factory) metric family.
  - Define a custom vendor certification for a custom metric family.
- A new vendor:
  - Create a custom vendor certification for a factory metric family.
  - Create a custom vendor certification for a custom metric family.

Prerequisites

The following information describes the prerequisites for working with the vendor certification XML.

- CA Performance Management Data Aggregator Release 2.2.00 or later is required.
- A solid knowledge of "certification" as it is described in the *CA Performance Management Data Aggregator Self-Certification Guide* is required, and experience using the Vendor Certification wizard.
- An XML editor.

  An XML editor that validates the XML with a schema is recommended. Follow the guidelines of your XML editor for working with the XSD files.

  Note: The examples in this guide were created and validated with XML Notepad 2007 for Windows, which is available as a free download from the Microsoft® Download Center.
- A MIB browser. For example, the free version of iReasoning is supported.

- A connection to a CA Performance Management Data Aggregator server.
- To support the example that we use in this guide, the FRAMERELAY-RFC1315-MIB and the IF-MIB are required.

Chapter 2: Downloading the Schema and Example Files

Before you create your custom metric family and vendor certification XML files, download and review the related schema XSD files and the example XML files. You need the schema to validate your own XML files. Reviewing the example files before creating your own helps ensure the accuracy of your XML content.

The schema files that are provided with Data Aggregator have detailed information about element types, occurrence, and allowed lengths. The files also contain annotations that provide more information, such as allowed characters and naming conventions.

Download the schema and example XML files by entering the following URLs in your web browser Address field. *Hostname* is the Data Aggregator host, and the default *port* is 8581.

- http://*hostname*:*port*/resource/xsd/IMDBCertificationFacet.xsd
- http://*hostname*:*port*/resource/xsd/ComponentFacet.xsd
- http://*hostname*:*port*/resource/xsd/ItemSyncDefinition.xsd
- http://*hostname*:*port*/resource/xsd/SNMPCertificationFacet.xsd
- http://*hostname*:*port*/resource/xsd/CertificationFacet.xsd
- http://*hostname*:*port*/resource/xsd/CertificationFacet.xsd
- http://*hostname*:*port*/resource/xsd/webservices.xsd
- http://*hostname*:*port*/resource/xsd/basewebservices.xsd
- http://*hostname*:*port*/resource/xsd/datamodel.xsd

Chapter 3: Creating a Custom Component

This section contains the following topics:

Create a Component XML Template

To create an example XML file that you can use as a template for creating your custom component, use your preferred REST client.

Start by retrieving a list of existing component definitions. You can then verify whether the component that you require is already supported.

Follow these steps:

1. Set up a REST client with a connection to the Data Aggregator server.
2. Enter the following URL for the Data Aggregator web services API in the REST client:

   `http://`*`da_hostname`*`:8581/typecatalog/components`

   A list of all available components displays.
3. Verify whether the component that you require is already supported.

If the component that you require is not already supported, view and export a single component definition that is similar to the custom component that you want to create.

Follow these steps:

1. To retrieve a specific component that is similar to the component that you require, enter the following URL:

   `http://`*`da-hostname`*`:8581/typecatalog/components/`*`name`*

   name

   Is the name of an existing, specific component, such as NormalizedCPUInfo.
2. Select GET on the Method tab.

3. Run the method.

   The XML that is returned includes the component definition.

   You can use this XML as a template to create new components.

4. Copy the component XML into a text file and then modify it, as needed. For an example of the XML structure, see Understanding the Component XML (see page 18).

Understanding the Component XML Structure

A device component defines a class of component items that are associated with a device. Many factory-defined components are provided, but a custom component is typically defined for a custom metric family. Components can define an optional ItemSyncDefinition, which synchronizes component items to CA Performance Center. You can then view the components in Inventory lists, groups, and context pages.

Here is an example of a component definition XML that supports our Frame-Relay PVC example. A new component, frPVC, has been added:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Auto-generated by the type catalog local manager.-->
<DataModel namespace="http://im.ca.com/inventory"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="ComponentFacet.xsd">
  <FacetType name="frPVC">
    <Documentation>A Frame Relay PVC</Documentation>
    <FacetOf namespace="http://im.ca.com/core" name="Item" />
    <Component>true</Component>
    <ItemSyncDefinition itemTypeName="component"
itemSubtypeName="frpvc" itemTypeLabel="FrameRelayPVC"
itemTypeLabelPlural="FrameRelayPVCs" categorize="false"
groupBy="false" context="true">
    </ItemSyncDefinition>
  </FacetType>
</DataModel>
```

Set the context in the ItemSyncDefinition to “true” to enable a link to a custom context page. You can navigate to this page from the frPVC device component where it appears in the Inventory, Device Components list. Setting it to “true” also enables you to select the metric family as a “context,” which makes your custom metric family available in the Dynamic Trend chart type. For more information, see ItemSyncDefinition (see page 20).

Basic Properties

The basic properties of your custom component help to distinguish it from other custom components that you create.

FacetType/name

Specifies the component name. Each component must have a unique name that identifies it internally within the system. Carefully choose a name with a minimal possibility of naming conflicts with future similar components. For example, define a naming scheme that helps ensure that these component names are unique.

Note: This name is never exposed externally. To display a component name in the user interface, use the ItemSyncDefinition, itemTypeLabel, and itemTypeLabelPlural elements.

Can be updated: No

Possible values: Alphanumeric and underscore. Dot and dash are not permitted.

Documentation

Specifies internal comments for the component. To help make these comments useful, we recommend that you describe why and when you added or changed the component.

Can be updated: Yes

Possible values: Plain text

Effect of updating: None

When does the update take effect: Immediately

Required Actions for updates to take effect: None

Component

Asserts that this item is a component.

Can be updated: No

Possible values: true

FacetOf

Asserts that this component is an item.

Can be updated: No

Possible values: namespace="http://im.ca.com/core" name="Item"

ItemSyncDefinition

The ItemSyncDefinition attribute is optional. This attribute specifies how the component items are synchronized and displayed in CA Performance Center. If the component items are not specified, they do not display in the CA Performance Center Inventory lists (for example, Device Components). But, they can still be reported on in custom views.

ItemSyncDefinition/itemTypeName

Specifies the item type. For custom components, this value must be “component.”

Can be updated: No

Possible values: Component

ItemSyncDefinition/itemSubtypeName

Specifies the internal name of the component in CA Performance Center. This value must be unique for all components. Use a naming convention that avoids conflicts with future factory and custom components, such as using a prefix representing your organization, 'acmeFan'.

Can be updated: No

Possible values: Alphanumeric, unique for all components

ItemSyncDefinition/itemTypeLabel

Specifies the user interface label that is used when displaying a single component of this type. For example, this value is used in the Inventory, Device Components UI “Type” column.

Can be updated: Yes

Possible values: Plain text, unique for all components

Effect of updating: Label is displayed in CA Performance Center Inventory user interfaces.

When does the update take effect: Allow for resync to occur and up to 15 minutes to complete updates.

Required actions for updates to take effect: None

ItemSyncDefinition/itemTypeLabelPlural

Specifies the user interface label that is used when displaying multiple components of this type. Used by the Inventory menu (see **groupBy**) and the Group name (see **categorize**).

Can be updated: Yes

Possible values: Plain text, unique for all components

Effect of updating: Label is displayed in CA Performance Center Inventory UIs.

When does the update take effect: Allow for resync to occur and up to 15 minutes to complete updates.

Required actions for updates to take effect: None

ItemSyncDefinition/categorize

Instructs CA Performance Center to create an inventory group under Inventory, All Items. This group contains all items of this component type. The group is named '*{itemTypeLabelPlural}'.*

Note: The inventory group that is created cannot be used for reporting dashboards. Instead, this group is for inventory purposes only. If the group is selected for reporting, then no data displays. Other, device-based inventory groups (under Inventory, All Items) can be used for reporting, such as Routers and Servers. However, component-based inventory groups cannot, such as Device Components.

Can be updated: Yes

Possible values: true, false

Effect of updating: Creates or removes an Inventory group in CA Performance Center. For CA Performance Center group administrators, on the Manage Groups page, the group is created under Inventory, All Items, *{itemTypeLabelPlural}*.

When does the update take effect: Allow for resync to occur and up to 30 minutes to complete updates.

Required actions for updates to take effect: Items typically appear in the group within 30 minutes. If they do not, manually resync the Data Aggregator datasource. Be sure to select 'Perform a Full Resynchronization' before clicking the Resync confirmation button.

ItemSyncDefinition/groupBy

Instructs CA Performance Center to create an inventory menu item (under Inventory) for viewing all items of this component item type. The menu is named "*{itemTypeLabelPlural}.*" This attribute also causes the component type show up in the Context Type drop-down menu when setting a view context. When false, the components are listed in the Inventory, Device Components table with the type of "*{itemTypeLabel}*." The groupBy property does not create a group (see **categorize**).

Can be updated: Yes

Possible values: true, false

Effect of updating: The menu item is created when true, or removed when false.

When does the update take effect: Allow for resynchronization to occur and allow up to 15 minutes to complete updates.

Required actions for updates to take effect: None

ItemSyncDefinition/context

Makes each component item name a context hyperlink in the Inventory component views that, when clicked, navigate to the individual component context page.

Can be updated: Yes/No

Possible values: Plain text

Effect of updating: Makes each component item name a context hyperlink in the Inventory component views.

When does the update take effect: Allow for resynchronization to occur and allow up to 15 minutes to complete updates.

Required actions for updates to take effect: None

Remove an ItemSyncDefinition

Completely removing an ItemSyncDefinition requires a specific procedure.

Follow these steps:

1. Remove the ItemSyncDefinition section, including the <ItemSyncDefinition> start and end tags.
2. After you apply the component change to Data Aggregator, log in to CA Performance Center as the administrator.
3. Select the DataAggregator datasource on the Data Sources page.

4. Click the Resync button.
5. Check the “Perform a Full Resynchronization” option, then click the Resync confirmation button.

   The resync process begins. Allow up to 15–30 minutes for the changes to synchronize.

   When this process is complete, all CA Performance Center behaviors that the ItemSyncDefinition defined for the component are removed.

Properties Not Supported for Custom Components

The following properties are *not* supported for custom components:

- Attribute
- WebService
- ItemSyncDefinition/isDeviceComponent*
- ItemSyncDefinition/mapped*
- ItemSyncDefinition/ItemProperty

* These properties *can* be present in your XML, but they cannot be set to true.

Import a Custom Component

After you import your custom component into Data Aggregator, a new component is available to support a custom metric family.

Data Aggregator provides a web service to support the import of a new component. Use any REST client. We recommend using the RESTClient with CA Performance Management. If you do not have it installed, you can get the RESTClient GUI JAR file from http://code.google.com/p/rest-client/.

When using the RESTClient, consider the following information:

- To launch it, double-click the JAR file.
- When you POST XML, make sure that the Charset is set to UTF-8.

  To view and verify this setting, click the Edit Content-Type & Charset button.
- You can also auto-indent the Response Body, as follows:
  1. Click Options on the Tools menu.
  2. Select the Etc. tab.
  3. Select Auto-indent Response Body and click OK.

Follow these steps:

1. Enter http://*da-hostname*:8581/typecatalog/components as the URL.
2. Select POST as the Method.
3. Select 'application/xml' as the 'Body Content-type' in the Body settings.

   Important! Failing to set the Content-type results in a 404 error.
4. Copy and paste your custom component XML into the Body field.
5. Run the method.

   Your custom component is imported. If no errors occur, the Status field in the HTTP Response section displays:

   ```
   HTTP/1.1 200 OK
   ```

   Note: Any other return code indicates that an error occurred while updating your custom component. Fix the error and retry updating the component by doing another POST.

   Important! To avoid possible data loss, always back up your deploy directory each time you create or update a vendor certification, metric family, or component.

Verify the Custom Component Results

After you have imported your custom component XML, verify the results. In our example, we verify that our frPVC component has been imported successfully.

Follow these steps:

1. Enter the following URL in your REST client:

   `http://da_hostname:8581/typecatalog/components/frPVC`

   A page displays the custom Frame-Relay component XML.

You can also discover the new component. After synchronization, you can then use the Inventory, Device Components list to view the component.

Update a Custom Component

You can update an existing custom component.

Note: For information about the effects of updating a tag or attribute when updating your components, review the custom component XML details. In particular, review the specific attribute descriptions.

Follow these steps:

1. Enter the following address in the URL field:

   ```
   http://da_hostname:8581/typecatalog/components/name
   ```

   Name

   Is the name of your custom component to update.

2. Select PUT in the Method tab.

3. Copy and paste your updated component XML into the Edit field on the Body tab, and set the Content-type to application/xml.

   Important! Failure to set the Content-Type results in a 404 error.

4. Click the Go button next to the URL field.

   Your custom component is updated. If no errors occur, the Status field in the HTTP Response section displays:

   ```
   HTTP/1.1 200 OK
   ```

   Note: Any other return code indicates that an error occurred while updating your custom component. Fix the error and retry updating the component by doing another PUT.

   Important! To avoid possible data loss, always back up your deploy directory each time you create or update a vendor certification, metric family, or component.

More information:

Understanding the Component XML Structure (see page 18)

Chapter 4: Creating a Custom Metric Family

Upon a successful import, a new metric family is available for supporting a custom vendor certification. The metric family is also available to associate with a monitoring profile for discovery and polling.

This section contains the following topics:

About Custom Metric Families

Factory metric families define the most common metric attributes to monitor. However, you can create a custom metric family when you want to collect data for new metric attributes. For example, if a metric family does not exist for collecting process data, you can create one. Then using your new metric family, create a vendor certification to monitor the metric attributes for processes.

Create a Metric Family XML Template

Use your preferred REST client to create an example XML file that you can use as a template for creating your custom metric family.

Start by retrieving a list of existing metric families. You can then verify whether the metric family that you require is already supported.

Follow these steps:

1. Set up a REST client with a connection to the Data Aggregator server.
2. Enter the following URL for the Data Aggregator web services API in the REST client:

   `http://`*`da_hostname`*`:8581/typecatalog/metricfamilies`

   A list of all available metric families displays.
3. Verify whether the metric family that you require is already supported.

If the metric family that you require is not already supported, view and export a metric family that is similar to the custom metric family that you want to create.

Follow these steps:

1. To retrieve a specific metric family that is similar to the metric family that you require, enter the following URL:

   `http://`*`da_hostname`*`:8581/typecatalog/metricfamilies/`*`name`*

   name

   Is the name of a specific, existing metric family, such as NormalizedCPUInfo.

2. Select GET in the Method tab.
3. Run the method.

   The XML that is returned includes the metric family information.

   You can use this XML as a template to create custom metric families.

4. Copy the metric family XML into a text file and then modify it, as needed. For an example of the XML structure, see Understanding the Metric Family XML (see page 28).

Understanding the Metric Family XML Structure

A metric family defines the set of metrics to collect and report on for a given technology. These metrics are normalized so that reporting is uniform regardless of the vendor (data source). Not all vendors provide a value for every metric in a metric family, and not all metrics are required. Metrics are "null" when the vendor does not provide a value. Also, any report views based on the null metrics are empty.

A metric family also defines attributes that are captured during discovery, like the item name and index. There can also be discovery rules defined that reconcile component matching. You include a metric family in a monitoring profile. The set of metric families in a monitoring profile determines which metrics to collect for the devices in each device collection that is associated with the profile.

Here is an example of a metric family that supports our Frame-Relay PVC example. Notice how our example custom component, frPVC, is included in the ComponentFacets section (in bold, for example purposes):

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Auto-generated by the type catalog local manager.  -->
<DataModel namespace="http://im.ca.com/normalizer"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="IMDBCertificationFacet.xsd">
  <FacetType name="frPVCInfo"
descriptorClass="com.ca.im.core.datamodel.certs.NormalizedFacetDes
criptorImpl">
    <Documentation>Frame Relay Permanent Virtual
Circuit</Documentation>
    <FacetOf namespace="http://im.ca.com/core" name="Item" />
    <AttributeGroup name="AttributeGroup" external="true"
list="true">
      <Documentation />
      <Attribute name="Indexes" type="ObjectID[]">
         <Documentation />
         <Polled>false</Polled>
         <Baseline>false</Baseline>
         <IsDbColumn>false</IsDbColumn>
         <Variance>false</Variance>
         <StandardDeviation>false</StandardDeviation>
         <Minimum>false</Minimum>
         <Maximum>false</Maximum>
         <WriteOnPoll>false</WriteOnPoll>
         <RollupStrategy />
         <AttributeDisplayName />
         <Percentile>0</Percentile>
       </Attribute>
       <Attribute name="Names" type="String">
         <Documentation>The name of the frame relay
circuit</Documentation>
         <Polled>false</Polled>
         <Baseline>false</Baseline>
         <IsDbColumn>false</IsDbColumn>
         <Variance>false</Variance>
         <StandardDeviation>false</StandardDeviation>
         <Minimum>false</Minimum>
         <Maximum>false</Maximum>
         <WriteOnPoll>false</WriteOnPoll>
         <RollupStrategy />
         <AttributeDisplayName />
         <Percentile>0</Percentile>
        </Attribute>
        <Attribute name="Description" type="String">
          <Documentation>A description for the frame relay
circuit</Documentation>
```

```
          <Polled>false</Polled>
          <Baseline>false</Baseline>
          <IsDbColumn>false</IsDbColumn>
          <Variance>false</Variance>
          <StandardDeviation>false</StandardDeviation>
          <Minimum>false</Minimum>
          <Maximum>false</Maximum>
          <WriteOnPoll>false</WriteOnPoll>
          <RollupStrategy />
          <AttributeDisplayName />
          <Percentile>0</Percentile>
        </Attribute>
        <Attribute name="BECNIn" type="Double">
          <Documentation>Backward congestion since the virtual
circuit was created</Documentation>
          <Polled>true</Polled>
          <Baseline>false</Baseline>
          <IsDbColumn>true</IsDbColumn>
          <Variance>false</Variance>
          <StandardDeviation>false</StandardDeviation>
          <Minimum>false</Minimum>
          <Maximum>false</Maximum>
          <WriteOnPoll>false</WriteOnPoll>
          <RollupStrategy>Sum</RollupStrategy>
          <AttributeDisplayName />
          <Percentile>0</Percentile>
        </Attribute>
        <Attribute name="FECNIn" type="Double">
          <Documentation>Forward congestion since the virtual
circuit was created</Documentation>
          <Polled>true</Polled>
          <Baseline>false</Baseline>
          <IsDbColumn>true</IsDbColumn>
          <Variance>false</Variance>
          <StandardDeviation>false</StandardDeviation>
          <Minimum>false</Minimum>
          <Maximum>false</Maximum>
          <WriteOnPoll>false</WriteOnPoll>
          <RollupStrategy>Sum</RollupStrategy>
          <AttributeDisplayName />
          <Percentile>0</Percentile>
        </Attribute>
        <Attribute name="FramesIn" type="Double">
          <Documentation>Frames received since the virtual circuit
was created</Documentation>
          <Polled>true</Polled>
          <Baseline>false</Baseline>
          <IsDbColumn>true</IsDbColumn>
          <Variance>false</Variance>
          <StandardDeviation>false</StandardDeviation>
          <Minimum>false</Minimum>
          <Maximum>false</Maximum>
          <WriteOnPoll>false</WriteOnPoll>
          <RollupStrategy>Sum</RollupStrategy>
          <AttributeDisplayName />
          <Percentile>0</Percentile>
        </Attribute>
        <Attribute name="FramesOut" type="Double">
          <Documentation>Frames sent since the virtual circuit was
created</Documentation>
          <Polled>true</Polled>
          <Baseline>false</Baseline>
          <IsDbColumn>true</IsDbColumn>
          <Variance>false</Variance>
          <StandardDeviation>false</StandardDeviation>
          <Minimum>false</Minimum>
          <Maximum>false</Maximum>
          <WriteOnPoll>false</WriteOnPoll>
          <RollupStrategy>Sum</RollupStrategy>
          <AttributeDisplayName />
          <Percentile>0</Percentile>
        </Attribute>
        <Attribute name="BytesIn" type="Double">
          <Documentation>Bytes received since the virtual circuit was
created</Documentation>
          <Polled>true</Polled>
          <Baseline>false</Baseline>
          <IsDbColumn>true</IsDbColumn>
          <Variance>false</Variance>
          <StandardDeviation>false</StandardDeviation>
          <Minimum>false</Minimum>
          <Maximum>false</Maximum>
          <WriteOnPoll>false</WriteOnPoll>
          <RollupStrategy>Sum</RollupStrategy>
          <AttributeDisplayName />
          <Percentile>0</Percentile>
        </Attribute>
        <Attribute name="BytesOut" type="Double">
          <Documentation>Bytes sent since the virtual circuit was
created</Documentation>
          <Polled>true</Polled>
          <Baseline>false</Baseline>
          <IsDbColumn>true</IsDbColumn>
          <Variance>false</Variance>
          <StandardDeviation>false</StandardDeviation>
          <Minimum>false</Minimum>
          <Maximum>false</Maximum>
          <WriteOnPoll>false</WriteOnPoll>
          <RollupStrategy>Sum</RollupStrategy>
          <AttributeDisplayName />
          <Percentile>0</Percentile>
        </Attribute>
      </AttributeGroup>
      <Attribute name="SourceFacetTypes" cached="true" list="true"
persistent="true" type="QName">
        <Documentation />
      </Attribute>
      <DisplayName>Frame Relay PVC</DisplayName>
      <Expressions>
        <ExpressionGroup destCert="{http://im.ca.com/core}Item">
          <Expression destAttr="Name">Names</Expression>
        </ExpressionGroup>
        <ExpressionGroup
destCert="{http://im.ca.com/inventory}DeviceComponent">
          <Expression destAttr="IndexList">Indexes</Expression>
        </ExpressionGroup>
        </Expressions>
        <TableName>FR_PVC_INFO</TableName>
        <ComponentFacets>
         <Facet>{http://im.ca.com/inventory}frPVC</Facet>
        </ComponentFacets>
        <Protocol>IMDB</Protocol>
        <Normalized>true</Normalized>
       </FacetType>
     </DataModel>
```

Basic Properties

The basic properties of your custom metric family help to distinguish it from other custom metric families you create.

Consider the following restrictions when you determine basic properties:

- The FacetType/name, DisplayName, and TableName properties must be unique for each metric family.
- The Protocol tag is always IMDB.
- The Normalized tag is always true.
- Set the FacetType/descriptorClass property and all DataModel and FacetOf properties.

FacetType/name

Specifies the metric family name. For each metric family, the name must be a unique name that identifies it internally within the system. Carefully choose a name with a minimal possibility of naming conflicts with future similar metric families. For example, define a naming scheme that helps ensure that these metric family names are unique.

Note: This name is never exposed externally. To display a metric family name in the user interface, use the DisplayName element.

Can be updated: No

Possible values: Alphanumeric and underscore. Dot and dash are not permitted. Must be unique across all metric families.

DisplayName

Specifies the metric family name that displays in the user interface.

Can be updated: Yes

Possible values: Plain text. Must be unique across all metric families.

Effect of updating: Change to the name in the administrator user interface.

When does the update take effect: Immediately

Required actions for updates to take effect: Refresh the user interface.

Documentation

Specifies the external description for the metric family. To help make these comments useful, we recommend that you describe why and when you added or changed the metric family.

Can be updated: Yes

Possible values: Plain text

Effect of updating: None

TableName

Specifies the database table name that is used to store the metrics that the metric family collects.

Can be updated: Yes

Possible values: Uppercase alphanumeric and underscore. Must begin with a letter. Must be unique across all metric families.

Example: PROCESS_STATS

Effect of updating: Poll data is stored in a new set of database tables.

Important! When you update the TableName, the old poll data is lost. Old report views are broken.

When does the update take effect: Immediately. Before new views can be created, there is a delay of up to 5 minutes while CA Performance Center loads the new MIB files.

Required actions for updates to take effect: Views must be recreated.

ComponentFacets

The ComponentsFacets section lists the facets that are created during discovery. Discovery identifies items as device components or creates a hierarchy relationship between items.

Facet

Specifies a facet that is attached to the component item during component discovery.

Can be updated: Yes

Possible values: QName of the facet

Effect of updating: If the component facet is synchronized to CA Performance Center, the component is visible in CA Performance Center.

When does the update take effect: Rediscover

Required actions for updates to take effect: Delete the device and rediscover.

More information:

Hierarchy (see page 45)
ComponentReconciliation (see page 46)

ItemFacets

Important! ItemFacets is a new section that will likely change to support future, complex metric family structures. Its use is discouraged.

ItemFacets lists the facets that are created during discovery that identify items as devices.

Facet

Specifies a facet that is attached to the item during discovery.

Can be updated: Yes

Possible values: QName of the facet

Effect of updating: Component is visible on the REST service for the specified facet. If the component facet is synchronized to CA Performance Center, the component is visible in CA Performance Center.

When does the update take effect: Rediscover

Required actions for updates to take effect: Delete the device and rediscover.

Example:

```
<ItemFacets>
    <Facet>{http://im.ca.com/inventory}Host</Facet>
    <Facet>{http://im.ca.com/inventory}Device</Facet>
    <Facet>{http://im.ca.com/inventory}ConsolidatedAndDiscoveredMetricFamilyHisto
    ry</Facet>
    <Facet>{http://im.ca.com/core}Syncable</Facet>
    <!-- The IPDomainID attribute will be filled in by discovery  -->
    <Facet>{http://im.ca.com/core}IPDomainMember</Facet>
</ItemFacets>
```

More information:

ItemReconciliation (see page 47)

SourceFacetTypes Attribute

A SourceFacetTypes attribute is required for discovery and must be defined.

Use these required values:

- Name: SourceFacetTypes
- Type: QName
- Cached: true
- Persistent: true
- List: true

Example: <Attribute name="SourceFacetTypes" type="QName" cached="true" persistent="true" list="true" />

Can be updated: No

AttributeGroup (Metric Family)

An AttributeGroup is a collection of item discovery attributes and metric attributes. The item discovery attributes are set during discovery, like item descriptions. The metric attributes are collected during polling. The following information describes the elements that you use in the AttributeGroup section.

Set the AttributeGroup/list and AttributeGroup/external properties to true. These properties specify that each attribute represents a list of values that is obtained from an external source. Customize the following XML elements:

AttributeGroup/name

Specifies the attribute group name. Conform to the "<FacetType/name>Group" naming scheme.

Can be updated: Yes

Possible values: Plain text

Effect of updating: None

Documentation

(Optional) Specifies the description for the attribute group.

Can be updated: Yes

Possible values: Plain text

Effect of updating: None

General Attributes (Metric Family)

The general attributes for all metric families are as follows:

Attribute/name

Specifies the unique, internal name. For metrics, this name is also used for naming the database column.

Note: This name is never exposed externally. To display an attribute name in the user interface, use the AttributeDisplayName element.

Can be updated: Yes

Possible values: Alphanumeric and underscore.

Effect of updating: For metrics, the values for this attribute are stored in a new database column corresponding to the updated name. The user loses the historical data that is collected for this metric (with the older name). The custom reports reporting on this metric fails.

When does the update take effect: Next poll

Required actions for updates to take effect: None

Attribute/type

Indicates the data type of this attribute. The most frequently used data types are Int, Long, Double, String, or ObjectID. The database stores metric attributes as a float. Therefore, these attributes must use a numeric type (we recommend a Double). Other types are used for item attributes.

Can be updated: Yes

Possible values: Boolean, Int, Long, Double (floating point), BigInteger, String, DateTime, IPAddress, MACaddress, IPSubnet, OctetString (hex representation), ObjectID, ItemID, QName (Qualified Name)

Note: The type names are case insensitive, for example, "boolean" is the same as Boolean.

Effect of updating: For metrics, none. All metrics are stored in the database as a float. For item attributes, the device must be deleted and rediscovered.

When does the update take effect: For metrics, next poll. For item attributes, on rediscover.

Required actions for updates to take effect: For metrics, none. For item attributes, delete the device and rediscover.

AttributeDisplayName

Specifies the value that appears in operator and administrator interfaces.

Can be updated: Yes

Possible values: Alphanumeric, space, and underscore.

Effect of updating: The metric reflects the updated AttributeDisplayName in the Metric Families UI and custom reports.

When does the update take effect: Immediately

Required actions for updates to take effect: None

Documentation

Displays the attribute description in the user interface. The documentation is also displayed in tool tips when you hover the cursor over the attribute name.

Can be updated: Yes

Possible values: Plain text

Effect of updating: Hovering the cursor over the attribute name shows the updated documentation.

When does the update take effect: Immediately

Required actions for updates to take effect: None

Polled

Indicates whether the attribute is polled. If it is set to false, it is only accessed during discovery.

Can be updated: Yes

Possible values: true, false

Effect of updating: If set to false, the OIDs corresponding to this attribute/metric are not polled when no other polled attribute/metric is using that OID in its expression. If set to true, the OIDs corresponding to this attribute/metric are polled.

When does the update take effect: Next poll

Required actions for updates to take effect: None

IsDbColumn

Stores its value in the database table. IsDbColumn is used for metric attributes. Set the IsDbColumn value to true when Polled is set to true.

Can be updated: Yes

Possible values: true, false

Effect of updating: If set to false, the data for this attribute/metric is not stored in the database. If set to true, the data for this attribute/metric is stored in the database.

When does the update take effect: Immediately

Required actions for updates to take effect: None

Discovery Attributes

For many attributes only the value that is retrieved during discovery is stored in the database. No further polling or processing, such as an evaluation of a baseline, is performed.

The Indexes and Names attributes must exist for all metric families. The Descriptions attribute is optional.

```
<Attribute name="Indexes" type="ObjectID[]"  />
<Attribute name="Names" type="String"  />
<Attribute name="Descriptions" type="String"  />
```

The metric families supporting Hierarchy must include these attributes:

```
<Attribute name="ItemUniqueIDs" type="String" />
<Attribute name="ParentUniqueIDs" type="String" />
```

Polled and Baseline Attributes

The following information describes the polled and baseline attribute elements:

Baseline

Indicates whether to calculate a mean value for this attribute. If it is set to true, a corresponding BaselineList definition must be defined.

Note: The Baseline attribute requires that the StandardDeviation attribute is set to true.

Can be updated: Yes

Possible values: true, false

Effect of updating: Baseline values are calculated when true.

When does the update take effect: Next poll

Required actions for updates to take effect: None

Maximum

Indicates whether to calculate the maximum of this attribute during the rollup. Creates a 'max_' column in the database table. If RollupStrategy is defined, this attribute must also be defined.

Can be updated: Yes

Possible values: true, false

Effect of updating: True provides a calculation of, and a reporting field for, “Maximum.”

When does the update take effect: Next poll

Required actions for updates to take effect: None

Minimum

Indicates whether to calculate the minimum of this attribute during the rollup. Creates a 'min_' column in the database table. If RollupStrategy is defined, this attribute must also be defined.

Can be updated: Yes

Possible values: true, false

Effect of updating: True provides a calculation of, and a reporting field for, “Minimum.”

When does the update take effect: Next poll

Required actions for updates to take effect: None

StandardDeviation

Indicates whether to calculate the standard deviation of this attribute during the rollup. Creates a 'std_' column in the database table. If RollupStrategy is defined, this attribute must also be defined.

Can be updated: Yes

Possible values: true, false

Effect of updating: True provides a calculation of, and a reporting field for, “Standard Deviation.”

When does the update take effect: Next poll

Required actions for updates to take effect: None

DeviationFromBaseline

Requires that the Baseline attribute is set to true. Provides two extra reporting fields, “Average Baseline” and “Percent Deviation,” calculated using baseline data. These fields are not available for building custom views. No changes are made to the database table.

Can be updated: Yes

Possible values: true, false

Effect of updating: True provides the “Average Baseline” and “Percent Deviation” fields for the internal report development.

When does the update take effect: Immediately

Required actions for updates to take effect: None

Percentile

Indicates whether to calculate the 95th percentile of this attribute during the rollup. Creates a 'pct_' column in the database table. If RollupStrategy is defined, this attribute must also be defined.

Can be updated: Yes

Possible values: 0 (zero), 95

Effect of updating: A value of 95 provides a calculation of, and a reporting field for, “95th Percentile.” Zero means that no calculation is performed, and the reporting field is not available.

When does the update take effect: Next poll

Required actions for updates to take effect: None

RollupStrategy

Specifies the operation that is performed every cycle during the rollup of the individually polled values. When Polled and IsDbColumn are set to true, this element is required.

Can be updated: Yes

Possible values: Sum (a summation for counters), Avg (an average for gauges)

Effect of updating: The specified strategy is used to perform rollup calculations.

When does the update take effect: Next poll

Required actions for updates to take effect: None

Rate

Provides an extra reporting field, “Average Rate,” calculated as AVG (metric value / time). No changes are made to the database table.

Note: The Rate is available for reporting but not for use when monitoring the profile event rules.

Can be updated: Yes

Possible values: true, false

Effect of updating: Provides the “Average Rate” field for reporting.

When does the update take effect: Immediately

Required actions for updates to take effect: None

Units

Specifies the name of the units label used in reports. The actual label that is displayed is translated according to the language setting of the report.

Can be updated: Yes

Possible values: Percent, Packets, PacketsPerSecond, DiscardedPackets, DiscardedPacketsPerSecond, ErroredPackets, ErroredPacketsPerSecond, Bits, BitsPerSecond, Bytes, BytesPerSecond, Microseconds, Milliseconds, UnixTime

Effect of updating: The specified units label is displayed in reports.

When does the update take effect: Immediately

Required actions for updates to take effect: None

Example: Using polled and baseline attribute elements

```
<Attribute name="Utilization" type="double">
    <AttributeDisplayName>Utilization</AttributeDisplayName>
    <AttributeAbbreviation>Utilization</AttributeAbbreviation>
    <IsDbColumn>true</IsDbColumn>
    <Baseline>true</Baseline>
    <Minimum>true</Minimum>
    <Maximum>true</Maximum>
    <RollupStrategy>Avg</RollupStrategy>
    <StandardDeviation>true</StandardDeviation>
    <DeviationFromBaseline>true</DeviationFromBaseline>
    <Percentile>95</Percentile>
    <Polled>true</Polled>
    <Units>Percent</Units>
</Attribute>
```

BaselineDefinitions

The BaselineDefinitions section contains the baseline definitions to calculate for this metric family. A baseline definition must be specified for each metric in the AttributeGroup section whose Baseline property is set to true.

You can define two types of baselines: Hourly (required) and Daily (optional). Hourly baselines are used both for event processing and for displaying baselines in reports. Daily baselines are used for displaying baselines in reports with a time frame of one month or greater.

The following information describes the baseline elements used:

Name

Specifies the type of baseline definition for a metric. The type is either hourly or daily.

Can be updated: No

Possible values: HourlyBaseline, DailyBaseline

ID

Specifies a value that is no longer used. However, the field must be specified as a positive integer and must be unique across all hourly and daily baseline definitions within this metric family.

Can be updated: Yes

Possible values: Any unique and positive integer.

Effect of updating: None

PerformanceMetric

Specifies the name (case sensitive) of the metric for which the baseline is calculated. Set the Polled and Baseline properties for the metric attribute to true.

Can be updated: Yes

Possible values: A valid metric name (case sensitive).

Effect of updating: Baseline calculations are performed for the metric.

When does the update take effect: Next baseline calculation, either hourly or daily.

Required actions for updates to take effect: None

Period

Specifies the type of baseline calculation, meaning either an hourly or daily baseline. Specify the value "1 Hour" for an HourlyBaseline Name or "1 Day" for a DailyBaseline Name.

Can be updated: Yes

Possible values: 1 Hour, 1 Day

Effect of updating: Baseline calculations are performed either hourly or daily.

When does the update take effect: Next baseline calculation, either hourly or daily

Required actions for updates to take effect: None

Window

Specifies a value that is no longer used. However, the field must be specified as "30 Days" for hourly baselines and "90 Days" for daily baselines.

Can be updated: No

Possible values: 30 Days, 90 Days

StartDate, EndDate, DaysOfWeek

Specifies more values that are not used, but they must be specified as 0 (zero).

Can be updated: No

Possible values: 0

Expressions

The Expressions section is composed of ExpressionGroup tags that are used for component discovery. During the component discovery, the values for the component item properties (such as the IndexList, Name, and Description) are calculated. The vendor certification expressions supporting the metric family expressions are used for this calculation.

Note: Do not confuse the metric family and vendor certification ExpressionGroup tags.

The ExpressionGroup tags for the following DestCert URIs must exist:

DestCert	**DestAttr**
{http://im.ca.com/core}Item	Name
{http://im.ca.com/core}Item	Description
{http://im.ca.com/inventory} DeviceComponent	IndexList

ExpressionGroup/name

(Optional) Specifies the expression group name.

Can be updated: Yes

Possible values: Plain text

Effect of updating: None

ExpressionGroup/destCert

Specifies the component facet that contains the destAttrs to populate. The facet name typically comes from the ComponentFacets section, except the Item and DeviceComponent facets.

Can be updated: Yes

Possible values: Facets that are defined in ComponentFacets, or the Item, DeviceComponent facet.

Effect of updating: Changes permissible expression destAttr

When does the update take effect: Component Rediscovery

Required actions for updates to take effect: None

ExpressionGroup/Expression

Specifies the expression for the component facet attribute.

Can be updated: Yes

Possible values: Any valid metric

Effect of updating: Changes permissible expression destAttr

When does the update take effect: Component Rediscovery

Required actions for updates to take effect: None

ExpressionGroup/Expression/destAttr

Specifies the component facet attribute name.

Can be updated: Yes

Possible values: Any valid attribute from that component facet.

Effect of updating: Changes the attribute name

When does the update take effect: Component Rediscovery

Required actions for updates to take effect: None

Hierarchy

A hierarchy, or parent-child relationship, can be defined between items of different metric families, for example, Interface and CBQoS Classmap. In the metric family definitions, the Hierarchy must be specified in the child metric family with:

- The Hierarchy QName in the ComponentFacets
- The ItemUniqueID and ParentUniqueID destAttr values in the Hierarchy ExpressionGroup
- The ItemUniqueIDs and ParentUniqueIDs attributes in the AttributeGroup

The supporting expressions are defined in the vendor certifications (see page 75).

The Hierarchy ExpressionGroup tags for the following DestCert URIs must exist:

DestCert	DestAttr
{http://im.ca.com/inventory}Hierarchy	ItemUniqueID
{http://im.ca.com/inventory}Hierarchy	ParentUniqueID

Example:

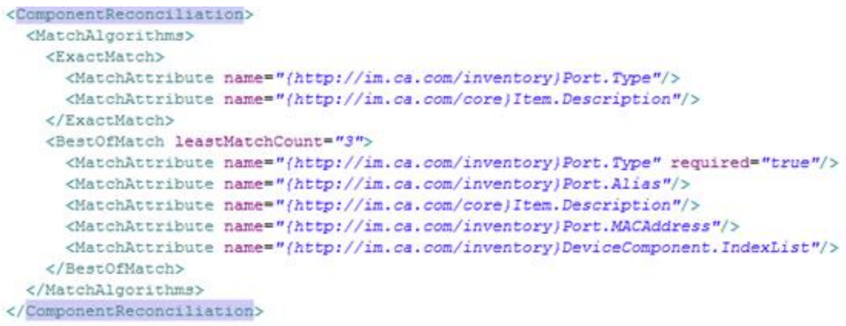
```
<ComponentFacets>
    <Facet>{http://im.ca.com/inventory}QoSClassMap</Facet>
    <Facet>{http://im.ca.com/inventory}Hierarchy</Facet>
</ComponentFacets>

<ExpressionGroup name="Hierarchy"
destCert="{http://im.ca.com/inventory}Hierarchy">
    <Expression destAttr="ItemUniqueID">ItemUniqueIDs</Expression>
    <Expression destAttr="ParentUniqueID">ParentUniqueIDs</Expression>
</ExpressionGroup>

<AttributeGroup name="QosCosGroup" list="true" external="true">
      <Attribute name="ItemUniqueIDs" type="String" />
      <Attribute name="ParentUniqueIDs" type="String" />
      ...
</AttributeGroup>
```

More information:

ComponentFacets (see page 34)

ComponentReconciliation

The following information defines the component reconciliation logic that is used in component discovery. First, this information determines whether the system has already discovered this component or not. Then, the reconciliation logic determines whether to update an existing component or create a new one.



```
<ComponentReconciliation>
  <MatchAlgorithms>
    <ExactMatch>
      <MatchAttribute name="{http://im.ca.com/inventory}Port.Type"/>
      <MatchAttribute name="{http://im.ca.com/core}Item.Description"/>
    </ExactMatch>
    <BestOfMatch leastMatchCount="3">
      <MatchAttribute name="{http://im.ca.com/inventory}Port.Type" required="true"/>
      <MatchAttribute name="{http://im.ca.com/inventory}Port.Alias"/>
      <MatchAttribute name="{http://im.ca.com/core}Item.Description"/>
      <MatchAttribute name="{http://im.ca.com/inventory}Port.MACAddress"/>
      <MatchAttribute name="{http://im.ca.com/inventory}DeviceComponent.IndexList"/>
    </BestOfMatch>
  </MatchAlgorithms>
</ComponentReconciliation>
```

There can be multiple ordered algorithms per metric family. If a metric family does not define a reconciliation algorithm, a default one with match attribute Item.Name is applied.

More information:

ComponentFacets (see page 34)
MatchAlgorithms (see page 48)

ItemReconciliation

Important! ItemReconciliation is a new section that will likely change to support future, complex metric family structures. Its use is discouraged.

The following information defines the item reconciliation logic that is used in item discovery. The logic determines whether the system has already discovered an item or not. Based on this determination, an existing item is updated or a new item is created. Item reconciliation is similar to component reconciliation. However, item reconciliation is used for items that are not components, such as virtual hosts. The ItemFacets are added to any new items or to any matching items (if the facets do not exist).

Example:

```
<ItemReconciliation>
    <SourceAgentScopedReconciliation>
        <MatchAlgorithms>
           <ExactMatch>
              <MatchAttribute
        name="{http://im.ca.com/inventory}SourceAgentInfo.SourceAgentIndexes" />
           </ExactMatch>
        </MatchAlgorithms>
    </SourceAgentScopedReconciliation>
    <GlobalScopedReconciliation matchDevices="true" />
</ItemReconciliation>
```

SourceAgentScopedReconciliation

Defines the match algorithms that are used to reconcile items.

Can be updated: Yes

Effect of updating: Changes the item reconciliation logic.

When does the update take effect: Rediscovery

Required actions for updates to take effect: Update the metric family or change the vendor certification priority.

GlobalScopedReconciliation

Defines the match algorithms that are used when items could not be reconciled for the source agent. The GlobalScopedReconciliation algorithms are used to locate items that have been created for other agents but match the potential new items. If the matchDevices property is set to true, the system default (built-in, not visible in XML) ComponentReconciliationInfo match algorithm is used. The match algorithm is based on the device primary IP address and host name.

Can be updated: Yes

Effect of updating: Changes the item reconciliation logic.

When does the update take effect: Rediscovery

Required actions for updates to take effect: Update the metric family or change the vendor certification priority.

More information:

ItemFacets (see page 34)
MatchAlgorithms (see page 48)

MatchAlgorithms

Component reconciliation and item reconciliation use match algorithms. Two match algorithms are supported:

- ExactMatch—All specified attributes must be matched to say the item matches with the new data.
- BestOfMatch—Users must specify the least number of attributes to be matched by using the "leastMatchCount" value. Also, each attribute has a "required" key property. If the required property is set to true, that attribute must be matched to be considered a match.

The algorithm has a match precedence when multiple algorithms are provided for a metric family. The order of the algorithms determines the precedence. The algorithm at the top has the highest precedence. The bottom one has the lowest precedence.

Each algorithm must have at least one matching attribute. When data matches to multiple items with the same algorithm, the item with the most matched attributes wins. When multiple matched items have the same number of matched attributes, the winner is picked at random from these items.

Examples: How the reconciliation works

Two match algorithms are provided for a metric family: alg1 and alg2. Alg1 has higher precedence than alg2. The metric family has three existing component items: 1, 2 and 3. Rediscovering the metric family finds three entries: A, B, and C. Now, we apply the two algorithms to determine which entry is new, changed, and unchanged.

```
   Reconciliation Meta Data                     New Data         Existing Components
<ComponentReconciliation>                           A                    1
  <MatchAlgorithms>                                 B                    2
    <MatchAlgorithm1>                               C                    3
      <MatchAttribute name="attr1"/>
      <MatchAttribute name="attr2"/>
    </MatchAlgorithm1>

    <MatchAlgorithm2>
      <MatchAttribute name="attr1"/>
      <MatchAttribute name="attr3"/>
      <MatchAttribute name="attr4"/>
    </MatchAlgorithm2>

  </MatchAlgorithms>
</ComponentReconciliation>
```

<MatchAlgorithm1> and <MatchAlgorithm2> can be either <ExactMatch> or <BestOfMatch>. The order of the two match algorithms tells us that MatchAlgorithm1 has a higher precedence than MatchAlgorithm2.

Case 1: Unique 1-to-1 Match

Entry A matches to item 1, and item 1 does not have any other match.

```
A -----> 1
```

This example is the simplest case. This match is unique, so it does not matter if it also matches alg1 or alg2. Entry A matches item 1.

A good match algorithm produces more unique matches.

Case 2: One entry has multiple matches

Entry A matches to item 1 by alg1 and also matches to 2 by alg2.

```
           ---> 1 (alg1)  (1 wins)
      /
    A                 Since alg1 has higher precedence, item 1 wins the match.
      \
           ---> 2 (alg2)
```

Case 3: Multiple entries match to the same item with different algorithms

Entry A matches to 1 by alg1 and entry B also matches to item 1 by alg2.

```
A ------> 1 (alg1)  (A wins)
B ------> 1 (alg2)
```

Since alg1 has higher precedence, entry A wins.

Case 4: Multiple entries match to the same item with same algorithm but different numbers of matched attributes

Both A and B match to 1 by alg1.

```
A ------> 1 (alg1, # of matched attrs: 2)  (A wins)
B ------> 1 (alg1, # of matched attrs: 1)
```

Because A has more matched attributes, A wins.

If the number of match attributes is the same, the winner is randomly picked and a warning is generated.

Case 5: Mixed match 1

```
 alg1
  A ------------------------------------> 1
    /  alg2(match attr count: 3)
  B
    \  alg2(match attr count: 2)
           --------------------------------> 2
```

A matches to 1 because it matches with a higher precedence algorithm.

B matches to 2 because 1 has matched to A.

Case 6: Mixed match 2

```
      ---------> 3
    /  alg1
  A                ==> A wins 3 because alg1 has a higher matching precedence
    \  alg2
      ---------> 1
    /  alg2
  B                ==> B wins 2 because alg1 has a higher matching precedence
    \  alg1
      ---------> 2
    /  alg2
  C                ==> C has no match because 2 is matched to B and 3 is matched
to A
    \  alg2
      ---------> 3
```

Entry C is treated as a new component. 1 is considered as an unmatched item.

The more match case 1 (unique match), the better the match algorithm is.

Can be updated: Yes

Effect of updating: Changes the component reconciliation logic.

When does the update take effect: Rediscovery

Required actions for updates to take effect: Update metric family or change vendor certification priority.

More information:

ComponentReconciliation (see page 46)
ItemReconciliation (see page 47)

ReconfigDetectionAttr

This element defines a metric family attribute that is used for change detection. You can enable the change detection on a monitoring profile. Data Aggregator polls that attribute only to check whether the target device has changed, instead of doing a complete rediscovery. This feature helps performance and helps reduce the network traffic.

Can be updated: Yes

Possible values: The full name of the metric family attribute. The specified metric family attribute flag should have the cached, persistent, and external flags set to true.

Effect of updating: Change to component reconfiguration detection.

When does the update take effect: After you rediscover.

Required actions for updates to take effect: Update metric family or change vendor certification priority.

Properties Not Supported for Custom Metric Families

The following properties are *not* supported for custom metric families:

- Variance
- RollupExpression

Important! If you use an exported factory metric family as a template, remove all unsupported elements or set them to false in your custom metric family.

Import a Custom Metric Family

To put your custom metric family into effect, import it into Data Aggregator.

CA Performance Center does not yet offer a user interface to support the import of a metric family that is based on the IMDBCertificationFacet.xsd schema directly. Therefore, leverage a web service that Data Aggregator exposes to accomplish this task.

Use any REST client. We recommend using the RESTClient with CA Performance Management. If you do not have it installed, you can get the RESTClient GUI JAR file from http://code.google.com/p/rest-client/.

When using the RESTClient, consider the following information:

- To launch it, double-click the JAR file.
- When you POST XML, make sure that the Charset is set to UTF-8.

  To view and verify this setting, click the Edit Content-Type & Charset button.
- You can also auto-indent the Response Body, as follows:
  1. Click Options on the Tools menu.
  2. Select the Etc. tab.
  3. Select Auto-indent Response Body and click OK.

Follow these steps:

1. Enter http://*da-hostname*:8581/typecatalog/metricfamilies as the URL.
2. Select POST as the Method.
3. Select 'application/xml' as the 'Body Content-type' in the Body settings.

   Important! Failing to set the Content-type results in a 404 error.
4. Copy and paste your custom metric family XML into the Body tab.
5. Run the method.

   Your custom metric family is imported. If no errors occur, the Status field in the HTTP Response section displays:

   ```
   HTTP/1.1 200 OK
   ```

   Note: Any other return code indicates that an error occurred while updating your custom metric family. Fix the error and retry updating the metric family by doing another POST.

   Important! To avoid possible data loss, always back up your deploy directory each time you create or update a vendor certification, metric family, or component.

Verify the Custom Metric Family Results

After you have imported your custom metric family XML, verify the results. In our example, we verify that the Frame Relay metric family has been imported successfully.

Follow these steps:

1. Enter the following URL in your REST client:

   `http://da_hostname:8581/typecatalog/metricfamilies/name`

   name

      Is the name of your custom metric family. In this example, it is FrameRelay.

   A page displays the custom Frame Relay metric family XML.

You can also verify that your custom metric family is listed on the Metric Families tab on the Monitoring Configuration menu for a Data Aggregator data source.

Additional Metrics Support

There may be situations where you want to add additional metrics to an existing custom metric family. Continuing our example, you want to add Bits In and Bits Out metrics.

Here is an example of what the updated frPVCInfo custom metric family looks like. The changes that we made in this procedure are indicated in bold:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Auto-generated by the type catalog local manager.-->
<DataModel namespace="http://im.ca.com/normalizer"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="IMDBCertificationFacet.xsd">
  <FacetType name="frPVCInfo"
descriptorClass="com.ca.im.core.datamodel.certs.NormalizedFacetDes
criptorImpl">
   <Documentation>Frame Relay Permanent Virtual
Circuit</Documentation>
   <FacetOf namespace="http://im.ca.com/core" name="Item" />
   <AttributeGroup name="AttributeGroup" external="true"
list="true">
     <Documentation />
     <Attribute name="Indexes" type="ObjectID[]">
       <Documentation />
       <Polled>false</Polled>
       <Baseline>false</Baseline>
       <IsDbColumn>false</IsDbColumn>
       <Variance>false</Variance>
       <StandardDeviation>false</StandardDeviation>
       <Minimum>false</Minimum>
       <Maximum>false</Maximum>
       <WriteOnPoll>false</WriteOnPoll>
       <RollupStrategy />
       <AttributeDisplayName />
       <Percentile>0</Percentile>
     </Attribute>
     <Attribute name="Names" type="String">
        <Documentation>The name of the frame relay
circuit</Documentation>
        <Polled>false</Polled>
        <Baseline>false</Baseline>
        <IsDbColumn>false</IsDbColumn>
        <Variance>false</Variance>
        <StandardDeviation>false</StandardDeviation>
        <Minimum>false</Minimum>
        <Maximum>false</Maximum>
        <WriteOnPoll>false</WriteOnPoll>
        <RollupStrategy />
        <AttributeDisplayName />
        <Percentile>0</Percentile>
      </Attribute>
      <Attribute name="Description" type="String">
        <Documentation>A description for the frame relay
circuit</Documentation>
        <Polled>false</Polled>
        <Baseline>false</Baseline>
        <IsDbColumn>false</IsDbColumn>
        <Variance>false</Variance>
        <StandardDeviation>false</StandardDeviation>
        <Minimum>false</Minimum>
        <Maximum>false</Maximum>
        <WriteOnPoll>false</WriteOnPoll>
        <RollupStrategy />
        <AttributeDisplayName />
        <Percentile>0</Percentile>
      </Attribute>
      <Attribute name="BECNIn" type="Double">
         <Documentation>Backward congestion since the virtual
circuit was created</Documentation>
         <Polled>true</Polled>
         <Baseline>false</Baseline>
         <IsDbColumn>true</IsDbColumn>
         <Variance>false</Variance>
         <StandardDeviation>false</StandardDeviation>
         <Minimum>false</Minimum>
         <Maximum>false</Maximum>
         <WriteOnPoll>false</WriteOnPoll>
         <RollupStrategy>Sum</RollupStrategy>
         <AttributeDisplayName />
         <Percentile>0</Percentile>
       </Attribute>
       <Attribute name="FECNIn" type="Double">
          <Documentation>Forward congestion since the virtual
circuit was created</Documentation>
          <Polled>true</Polled>
          <Baseline>false</Baseline>
          <IsDbColumn>true</IsDbColumn>
          <Variance>false</Variance>
          <StandardDeviation>false</StandardDeviation>
          <Minimum>false</Minimum>
          <Maximum>false</Maximum>
          <WriteOnPoll>false</WriteOnPoll>
          <RollupStrategy>Sum</RollupStrategy>
          <AttributeDisplayName />
          <Percentile>0</Percentile>
        </Attribute>
        <Attribute name="FramesIn" type="Double">
           <Documentation>Frames received since the virtual circuit
was created</Documentation>
           <Polled>true</Polled>
           <Baseline>false</Baseline>
           <IsDbColumn>true</IsDbColumn>
           <Variance>false</Variance>
           <StandardDeviation>false</StandardDeviation>
           <Minimum>false</Minimum>
           <Maximum>false</Maximum>
           <WriteOnPoll>false</WriteOnPoll>
           <RollupStrategy>Sum</RollupStrategy>
           <AttributeDisplayName />
           <Percentile>0</Percentile>
         </Attribute>
         <Attribute name="FramesOut" type="Double">
           <Documentation>Frames sent since the virtual circuit was
created</Documentation>
           <Polled>true</Polled>
           <Baseline>false</Baseline>
           <IsDbColumn>true</IsDbColumn>
           <Variance>false</Variance>
           <StandardDeviation>false</StandardDeviation>
           <Minimum>false</Minimum>
           <Maximum>false</Maximum>
           <WriteOnPoll>false</WriteOnPoll>
           <RollupStrategy>Sum</RollupStrategy>
           <AttributeDisplayName />
           <Percentile>0</Percentile>
         </Attribute>
         <Attribute name="BytesIn" type="Double">
           <Documentation>Bytes received since the virtual circuit
was created</Documentation>
           <Polled>true</Polled>
           <Baseline>false</Baseline>
           <IsDbColumn>true</IsDbColumn>
           <Variance>false</Variance>
           <StandardDeviation>false</StandardDeviation>
           <Minimum>false</Minimum>
           <Maximum>false</Maximum>
           <WriteOnPoll>false</WriteOnPoll>
           <RollupStrategy>Sum</RollupStrategy>
           <AttributeDisplayName />
           <Percentile>0</Percentile>
         </Attribute>
         <Attribute name="BytesOut" type="Double">
           <Documentation>Bytes sent since the virtual circuit was
created</Documentation>
           <Polled>true</Polled>
           <Baseline>false</Baseline>
           <IsDbColumn>true</IsDbColumn>
           <Variance>false</Variance>
           <StandardDeviation>false</StandardDeviation>
           <Minimum>false</Minimum>
           <Maximum>false</Maximum>
           <WriteOnPoll>false</WriteOnPoll>
           <RollupStrategy>Sum</RollupStrategy>
```

```
           <AttributeDisplayName />
           <Percentile>0</Percentile>
         </Attribute>
         <Attribute name="BitsIn" type="Double">
            <Documentation>Bits received since the virtual circuit
was created</Documentation>
            <Polled>true</Polled>
            <Baseline>false</Baseline>
            <IsDbColumn>true</IsDbColumn>
            <Variance>false</Variance>
            <StandardDeviation>false</StandardDeviation>
            <Minimum>false</Minimum>
            <Maximum>false</Maximum>
            <WriteOnPoll>false</WriteOnPoll>
            <RollupStrategy>Sum</RollupStrategy>
            <AttributeDisplayName />
            <Percentile>0</Percentile>
          </Attribute>
          <Attribute name="BitsOut" type="Double">
            <Documentation>Bits sent since the virtual circuit was
created</Documentation>
            <Polled>true</Polled>
            <Baseline>false</Baseline>
            <IsDbColumn>true</IsDbColumn>
            <Variance>false</Variance>
            <StandardDeviation>false</StandardDeviation>
            <Minimum>false</Minimum>
            <Maximum>false</Maximum>
            <WriteOnPoll>false</WriteOnPoll>
            <RollupStrategy>Sum</RollupStrategy>
            <AttributeDisplayName />
            <Percentile>0</Percentile>
          </Attribute>
        </AttributeGroup>
        <Attribute name="SourceFacetTypes" cached="true"
list="true" persistent="true" type="QName">
          <Documentation />
        </Attribute>
        <DisplayName>Frame Relay PVC</DisplayName>
        <Expressions>
          <ExpressionGroup destCert="{http://im.ca.com/core}Item">
             <Expression destAttr="Name">Names</Expression>
          </ExpressionGroup>
          <ExpressionGroup
destCert="{http://im.ca.com/inventory}DeviceComponent">
            <Expression destAttr="IndexList">Indexes</Expression>
          </ExpressionGroup>
        </Expressions>
        <TableName>FR_PVC_INFO</TableName>
        <ComponentFacets>
          <Facet>{http://im.ca.com/inventory}frPVC</Facet>
        </ComponentFacets>
        <Protocol>IMDB</Protocol>
        <Normalized>true</Normalized>
      </FacetType>
</DataModel>
```

We are assuming that you made these changes after you imported the custom metric family. Therefore, after you make the changes, update the metric family. After you update the metric family, you must also update the vendor certification that the metric family is associated with.

More information:

Update a Custom Vendor Certification (see page 90)
Update a Custom Metric Family (see page 58)

Update a Custom Metric Family

You can update an existing custom metric family. In this example, you previously added metrics to your custom metric family, frPVCInfo. Now, update the metric family for the changes to take effect.

Note: For information about the effects of updating a tag or attribute when updating your custom metric family, review the custom metric family XML details. In particular, review the specific attribute descriptions.

Follow these steps:

1. Enter the following address in the URL field:

   `http://`*`da_hostname`*`:8581/typecatalog/metricfamilies/`*`name`*

   Name

      Is the name of the custom metric family to update. For our example, the metric family is named frPVCInfo.

2. To verify that the metric family exists at that URL, select GET in the Method tab.

   Note: If the metric family does not exist, you can Import a Custom Metric Family (see page 52).

3. After you verify that the metric family exists at the URL, select PUT in the Method tab.

4. Copy and paste your updated custom metric family XML into the Body tab (Edit field) and set the Content-type to application/xml.

   Important! Failing to set the Content-type results in a 404 error.

5. Click the Go button next to the URL field.

   Your custom metric family is updated. If no errors occur, the Status field in the HTTP Response section displays the following message:

   ```
   HTTP/1.1 200 OK
   ```

   Any other return code indicates that an error occurred while updating your custom metric family. Fix the error and retry updating the metric family by doing another PUT.

   Important! To avoid possible data loss, always back up your deploy directory each time you create or update a vendor certification, metric family, or component.

6. Next, update the custom vendor certification that this metric family is associated with (see page 90).

More information:

Understanding the Metric Family XML Structure (see page 28)
Additional Metrics Support (see page 53)

Chapter 5: Creating a Custom Vendor Certification

Upon a successful import, the new vendor certification is automatically added to the end of the priority list for the specified metric family. You can modify the priority list and can move your vendor certification higher in priority when needed. The vendor certification is used when discovering device support for the specified metric family.

Note: Data Aggregator provides basic and advanced methods for creating custom vendor certifications and metric families. The basic method is a simpler process, consisting of adding vendor support for existing supported technologies (metric families), using the user interface. This method meets the requirements of many users. However, the advanced method is based on the factory certification format and exposes a complete set of capabilities. This guide explains the advanced certification method. For information about the basic certification method, see the *Data Aggregator Self-Certification Guide*.

This section contains the following topics:

Create a Vendor Certification XML Template

Use your preferred REST client to create an example XML file that you can use as a template for creating your custom vendor certification.

Start by retrieving a list of existing vendor certifications. You can then verify whether the vendor certification that you require is already supported.

Follow these steps:

1. Set up a REST client with a connection to the Data Aggregator server.
2. Enter the following URL for the Data Aggregator web services API in the REST client:

   ```
   http://da_hostname:8581/typecatalog/certifications/snmp
   ```

   A list of all available vendor certifications displays.
3. Verify whether the vendor certification that you require is already supported.

If the vendor certification that you require is not already supported, view and export a vendor certification that is similar to the custom vendor certification that you want to create.

Follow these steps:

1. Enter the following URL to retrieve a specific vendor certification that is similar to the metric family that you require:

   ```
   http://da_hostname:8581/typecatalog/certifications/snmp/name
   ```

   name

   Is the name of your vendor certification, such as CiscoCPUMibRev.

2. Select GET in the Method tab.
3. Run the method.

   The XML that is returned includes the vendor certification information.

   You can use this XML as a template to create custom vendor certifications.

4. Copy the vendor certification XML into a text file and then modify it, as needed. For an example of the XML structure, see Understanding the Vendor Certification XML (see page 62).

Understanding the Vendor Certification XML Structure

A vendor certification maps vendor- and device-specific data to performance metrics and configuration data defined in a metric family. Mapping this data from various sources to the "normalized" metric family values helps Data Aggregator uniformly report on this data, regardless of the device vendor.

Here is an example of a custom vendor certification XML that supports our Frame-Relay PVC example. Notice how our example custom metric family, frPVCInfo, is included in the ExpressionGroup section (in bold, for example purposes):

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Auto-generated by the type catalog local manager.-->
<DataModel namespace="http://im.ca.com/certifications/snmp"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="SNMPCertificationFacet.xsd">
  <FacetType name="frPVCInfoCustom"
descriptorClass="com.ca.im.core.datamodel.certs.CertificationFacet
DescriptorImpl">
    <Documentation>Frame Relay PVC Vendor
Certification</Documentation>
    <FacetOf namespace="http://im.ca.com/core" name="Item" />
    <AttributeGroup name="AttributeGroup" external="true"
list="true">
      <Documentation />
      <Attribute name="INDEX" type="ObjectID">
      <Documentation />
      <Source>1.3.6.1.2.1.10.32.2.1.4</Source>
      <IsIndex>true</IsIndex>
      <IsKey>false</IsKey>
      <NeedsDelta>false</NeedsDelta>
    </Attribute>
    <Attribute name="frCircuitReceivedBECNs" type="Long">
      <Documentation />
      <Source>1.3.6.1.2.1.10.32.2.1.5</Source>
      <IsIndex>false</IsIndex>
      <IsKey>true</IsKey>
      <NeedsDelta>true</NeedsDelta>
    </Attribute>
    <Attribute name="frCircuitSentFrames" type="Long">
      <Documentation />
      <Source>1.3.6.1.2.1.10.32.2.1.6</Source>
      <IsIndex>false</IsIndex>
      <IsKey>true</IsKey>
      <NeedsDelta>true</NeedsDelta>
    </Attribute>
    <Attribute name="frCircuitSentOctets" type="Long">
      <Documentation />
      <Source>1.3.6.1.2.1.10.32.2.1.6</Source>
      <IsIndex>false</IsIndex>
      <IsKey>true</IsKey>
      <NeedsDelta>true</NeedsDelta>
    </Attribute>
    <Attribute name="frCircuitReceivedFrames" type="Long">
      <Documentation />
      <Source>1.3.6.1.2.1.10.32.2.1.8</Source>
      <IsIndex>false</IsIndex>
      <IsKey>true</IsKey>
      <NeedsDelta>true</NeedsDelta>
    </Attribute>
    <Attribute name="frCircuitReceivedOctets" type="Long">
      <Documentation />
      <Source>1.3.6.1.2.1.10.32.2.1.9</Source>
      <IsIndex>false</IsIndex>
      <IsKey>true</IsKey>
      <NeedsDelta>true</NeedsDelta>
    </Attribute>
  </AttributeGroup>
  <Protocol>SNMP</Protocol>
  <DisplayName>Frame Relay PVC Certification</DisplayName>
  <Expressions>
    <ExpressionGroup
destCert="{http://im.ca.com/normalizer}frPVCInfo"
name="frPVCInfoDS">
      <Expression destAttr="Indexes">INDEX</Expression>
      <Expression destAttr="Names">"Frame Relay " +
INDEX</Expression>
      <Expression
destAttr="FECNIn">frCircuitReceivedFECNs</Expression>
      <Expression
destAttr="BECNIn">frCircuitReceivedBECNs</Expression>
      <Expression
destAttr="FramesIn">frCircuitReceivedFrames</Expression>
      <Expression
destAttr="FramesOut">frCircuitSentFrames</Expression>
      <Expression
destAttr="BytesIn">frCircuitReceivedOctets</Expression>
      <Expression
destAttr="BytesOut">frCircuitSentOctets</Expression>
    </ExpressionGroup>
  </Expressions>
  <MIB>RFC1315-MIB</MIB>
 </FacetType>
</DataModel>
```

Basic Properties

The basic properties of your custom vendor certification help to distinguish it from other custom vendor certifications you create. Also, these properties indicate from which vendor MIB you are collecting metric data.

Consider the following restrictions when you determine basic properties:

- The FacetType/name and FacetType/DisplayName properties must be unique for each vendor certification.
- The Protocol tag is always SNMP.
- Set the FacetType/descriptorClass property and all DataModel and FacetOf properties as shown in the example XML in the previous illustration.

FacetType/name

Uniquely identifies a vendor certification.

Recommendation: Conform to "<MibName><TableName>Mib."

Can be updated: No

Possible values: Alphanumeric and underscore. Dot and dash are not permitted.

The FacetType section manifests a particular vendor certification. The same XML document can contain multiple FacetType sections when those vendor certifications expose various aspects of the vendor-specific device such as TCP and UDP statistics from a MIB-2 implementation.

The FacetType section contains some basic properties. For example, this section contains the name of the vendor MIB, followed by one or more AttributeGroup sections. These AttributeGroup sections define which attributes this certification uses from the MIB. Finally, there are one or more ExpressionGroup sections that map attributes from the AttributeGroup sections to the metrics specified in a metric family.

FacetType/Documentation

Describes what is certified with the vendor certification.

Recommendation: Include the details about the vendor, MIB name, and table name.

Can be updated: Yes

Possible values: Plain text

Effect of updating: None

FacetType/MIB

Specifies the name of the MIB, which the DEFINITIONS clause defines in the ASN.1 file.

Recommendation: Conform to "<MibName>”

Can be updated: Yes

Possible values: Plain text

Effect of updating: Change to the “SNMP MIB Name” column in the Vendor Certification tab of the Administrator user interface.

When does the update take effect: Immediately

Required actions for updates to take effect: Refresh the user interface.

FacetType/DisplayName

Specifies the name of the vendor certification as it displays in CA Performance Center.

Recommendation: Start with the vendor name and include the MIB and functionality information.

Can be updated: Yes

Possible values: Plain text

Effect of updating: A change to the name in the Administrator user interface.

When does the update take effect: Immediately

Required actions for updates to take effect: Refresh the user interface.

AttributeGroup (Vendor Certification)

The following example illustrates the AttributeGroup section of your custom vendor certification. This section identifies the attributes (variable OIDs) of a particular table in the vendor MIB that are used to map raw device data. This data is mapped to the performance metrics and the configuration data that is defined in a metric family.

You set the AttributeGroup/list and AttributeGroup/external properties to true, as shown in the example XML in the previous illustration. These properties specify that each attribute represents a list of values that is obtained from an external source (a MIB table). The following information summarizes the XML elements to customize.

AttributeGroup/name

Specifies the attribute group name.

Recommendation: Conform to "<FacetType/name>Group."

Can be updated: Yes

Possible values: Plain text

Effect of updating: None

Documentation

(Optional) Specifies the description for the attribute group.

Can be updated: Yes

Possible values: Plain text

Effect of updating: None

UseIndex

Specifies the name of the attribute to be used as the index for this attribute group for joining multiple MIB tables.

Recommendation: Set to the value of the AttributeGroup/name property.

Can be updated: Yes

Possible values: Plain text

Effect of updating: None

More information:

AttributeGroup (Multiple MIB Tables) (see page 88)
IndexTagList (see page 76)

General Attributes (Vendor Certification)

The general attributes for all vendor certifications are as follows:

Attribute/name

Specifies the attribute name.

Recommendation: Set to the MIB variable name, which the OBJECT-TYPE clause defines in the ASN.1 file.

Can be updated: Yes

Possible values: Alphanumeric and underscore. Dot and dash are not permitted.

Effect of updating: You must update any expressions that reference this attribute.

When does the update take effect: Immediately

Required actions for updates to take effect: None

Attribute/type

Specifies the data type of the attribute.

Recommendation: Use the attribute type that best matches the variable type that the SYNTAX clause defines in the ASN.1 file.

Can be updated: Yes

Possible values: Boolean, Int, Long, Double, BigInteger, String, DateTime, IPAddress, MACaddress, IPSubnet, OctetString, ObjectID

Effect of updating: Polled SNMP data is converted to this type.

When does the update take effect: Next poll

Required actions for updates to take effect: None

Documentation

(Optional) Specifies the description for the attribute, which documents the semantics (such as the unit) of the MIB variable.

Recommendation: Use the descriptions that are taken from the MIB ASN.1 file.

Can be updated: Yes

Possible values: Plain text

Effect of updating: None

IsKey

(Optional) Uses a flag to indicate whether the MIB variable is key for determining support for a table. When multiple fields are specified as keys, all of the fields are considered together as a compound key.

Default: false

Recommendation: Set to true if it is a key MIB object for component discovery.

Can be updated: Yes

Possible values: true, false

Effect of updating: Components could change to a new vendor certification.

When does the update take effect: After component rediscovery.

Required actions for updates to take effect: Update the metric family or change the vendor certification priority.

IsIndex

(Optional) Uses a flag to indicate whether this variable is an index to the MIB table.

Default: false

Recommendation: Set to true for an index attribute.

Can be updated: Yes

Possible values: true, false

Effect of updating: Component indexing could change.

When does the update take effect: After component rediscovery.

Required actions for updates to take effect: Update the metric family or change the vendor certification priority.

NeedsDelta

(Optional) Uses a flag to indicate whether to delta (that is, store the difference between current and last poll for Counters) the MIB variable.

Default: false

Recommendation: Set to true if the variable is defined as a Counter, Counter32, Counter64, or TimeTicks quantity in the MIB.

Can be updated: Yes

Possible values: true, false

Effect of updating: The polled data changes.

When does the update take effect: Next poll

Required actions for updates to take effect: None

Source

Specifies the ObjectID of the attribute.

Recommendation: Set to the fully qualified MIB variable OID that the OBJECT-TYPE defines.

Can be updated: Yes

Possible values: Dot-separated numbers (for example, 1.3.6.1.4.1…)

Effect of updating: Data is polled from the specified OID.

When does the update take effect: Next poll

Required actions for updates to take effect: None

Version

Specifies the version of the vendor certification. When a vendor certification is installed during an upgrade, the installer checks the version of the certification. If the version is newer than the one in the installer package, the installer does not change the existing vendor certification.

Default: 1.0

Can be updated: Yes

Possible values: Dot-separated numbers (for example, 1.3.6.1.4.1…)

Effect of updating: The version attribute is updated.

When does the update take effect: Immediately

Required actions for updates to take effect: None

Author

Specifies the creator of the vendor certification.

Default: "Custom"

Can be updated: Yes

Possible values: Any alphanumeric string.

Effect of updating: The author attribute is updated.

When does the update take effect: Immediately

Required actions for updates to take effect: None

The list of attributes specifies the set of data that a metric family collects when supported by this vendor certification. Typically, this data falls into two categories:

- Configuration data of the device component (such as name or indices) that is collected only at discovery time.
- Performance data that is collected every poll cycle.

More information:

Multi-MIB Table Support (see page 84)
Filtering Support (see page 81)

Configuration Data Attributes

An attribute with the name INDEX and type ObjectID is mapped to the Indexes attribute of the target metric family. You can set the value for the Source tag to any variable OID. However, you typically use one of the variables that are listed in the INDEX clause of the table. For example, consider ifIndex in the interfaces table of MIB-2. This variable serves as the index for the other variables in the same MIB table. In addition, the IsIndex tag (and typically also the IsKey tag) for this attribute is set to true.

In this example, attributes such as ifDesc or ifType provide more configuration information on an interface. Therefore, these attributes are useful for the Names and Description attributes of the target metric family.

Performance Data Attributes

These attributes provide the raw data for performance metrics in the target metric family. Consider the following points:

- You can directly map one of these attributes to a metric family performance metric, or
- You can use the attribute in an expression along with other attributes to compute a value for the metric.

ExpressionGroup

The ExpressionGroup maps attributes as follows:

- From the AttributeGroup (that defines how to get a metric from an SNMP MIB)
- To the metrics specified in a metric family (that defines how an attribute is stored in the database)

You can store a MIB value in the database as it is received from the device or after some normalization operations are performed. For example, normalization operations include dividing or multiplying with 1024 to transform to/from kilobytes.

ExpressionGroup/name

(Optional) Specifies the expression group name.

Can be updated: Yes

Possible values: Plain text

Effect of updating: None

ExpressionGroup/destCert

Specifies the metric family that contains the destAttrs to populate.

Can be updated: Yes

Possible values: Any valid metric family

Effect of updating: Changes the permissible expression destAttr.

When does the update take effect: Immediately

Required actions for updates to take effect: None

ExpressionGroup/Filter

(Optional) Specifies which components are discovered. Using the Filter reduces the number of components that are managed.

Can be updated: Yes

Possible values: Boolean MVEL expression using available Attributes

Effect of updating: Changes which components are discovered.

When does the update take effect: After component rediscovery.

R**equired actions for updates to take effect:** Update the metric family or change the vendor certification priority.

Filters in ExpressionGroup

The following list describes filter examples.

Example 1:

```
<Filter> hrStorageType.toString() == "1.3.6.1.2.1.25.2.1.4" &amp;&amp;
         hrStorageSize != 0
</Filter>
```

In this example only partition instances that have a size <> 0 and a storageType == hrStorageFixedDisk are created in the data repository.

Example 2:

```
<Filter> (jnxOperatingCPUIndex.toString() == "9.2.0.0" ||
          jnxOperatingCPUIndex.toString() == "9.1.0.0" ) &amp;&amp;
         (jnxOperatingState != 1 || jnxOperatingCPU > 0 )
</Filter>
```

Example 3:

```
<Filter> (rttMonCtrlAdminRttType==9) &amp;&amp;
         ( !(rttMonCtrlAdminOwner.toString() contains "Network Health") )
</Filter>
```

Example 4:

```
<Filter> (snmpOIDParser(sonetIndex,2,2).toString()=="1") </Filter>
```

This example uses a vendor certification utility function to parse the OID.

Note: For a complete list of functions, see the appendix in this guide.

Expression/destAttr Metrics

The following information describes the Expression/destAttr metrics:

Indexes

Specifies to use the vendor certification attributes of the ObjectID to define the MVEL expression to provide the value for the Indexes metric family attribute.

Recommendation: Set to INDEX.

Can be updated: Yes

Possible values: Any attribute that has <IsIndex>true</IsIndex>.

Effect of updating: Component indexing could change.

When does the update take effect: After component rediscovery

Required actions for updates to take effect: Update the metric family or change the vendor certification priority.

Names

Specifies to use the vendor certification attributes to collect configuration data. This configuration data helps define the MVEL expression to provide the value for the Names metric family attribute.

Recommendation: Include as much information as necessary to identify an instance uniquely.

Can be updated: Yes

Possible values: String MVEL expression using available Attributes

Effect of updating: Component name change

When does the update take effect: After component rediscovery

Required actions for updates to take effect: Update the metric family or change the vendor certification priority.

Descriptions

(Optional) Specifies to use the vendor certification attributes to collect configuration data. This configuration data helps define the MVEL expression to provide the value for the Descriptions metric family. Not all metric families support a Descriptions attribute.

Recommendation: Include as much information that is available to describe an instance.

Can be updated: Yes

Possible values: String MVEL expression using available Attributes

Effect of updating: Component description change

When does the update take effect: Component rediscovery

Required actions for updates to take effect: Update the metric family or change the vendor certification priority.

Other Metrics

Specifies to use the vendor certification attributes to collect configuration or performance data. This data is used to define the MVEL expression to provide the value for the metric family attribute.

Can be added: Yes, if the destAttr exists in the metric family.

Can be updated: Yes

Possible values: MVEL expression using available Attributes, producing a value that matches the type of the destination attribute.

Effect of updating: Polled value changes

When does the update take effect: Next poll

Required actions for updates to take effect: None

The metric family exposes URIs (such as, {http://im.ca.com/normalizer}*FamilyName*.*AttributeName*), which are separately referred to in the ExpressionGroup. The ExpressionGroup/destCert property is set to the URI (for example, {http://im.ca.com/normalizer}*FamilyName*), and the Expression/destAttr is set to *AttributeName*.

More information:

Configuration Data Attributes (see page 71)

HierarchyList

The following information defines the hierarchy behavior.

Hierarchy/ParentFacet

Specifies the QName of the facet that is used to find the candidate parent items.

Can be updated: Yes

Possible values: Any valid facet

Effect of updating: Changes the hierarchy construction.

When does the update take effect: After component rediscovery

Required actions for updates to take effect: Update the metric family or change the vendor certification priority.

Hierarchy/ParentAttribute

Specifies the QName of the attribute that is used to identify the specific parent item.

Can be updated: Yes

Possible values: Any valid attribute QName

Effect of updating: Changes the hierarchy construction.

When does the update take effect: After component rediscovery

Required actions for updates to take effect: Update the metric family or change the vendor certification priority.

Hierarchy/ChildAttribute

Specifies the QName of the attribute on the child item that is used to match the ParentAttribute on the parent item.

Can be updated: Yes

Possible values: Any valid attribute QName

Effect of updating: Changes the hierarchy construction.

When does the update take effect: After component rediscovery

Required actions for updates to take effect: Update the metric family or change the vendor certification priority.

IndexTagList

To poll attributes from multiple MIB tables, we need an attribute group per MIB table containing these attributes. The index tag list provides a mechanism to relate two attribute groups (or MIB tables) with different indexes. The groups are related such that one item (row) of one table is linked to a corresponding row in a second table.

PrimaryTag

References the primary Attribute group (that is, the group that defines an index attribute with the ObjectID type). The value of this element must equal the 'UseIndex' tag of the attribute group for the primary group.

Can be updated: Yes

Possible values: The ‘UseIndex’ tag of the attribute group corresponding to the primary attribute group.

Effect of updating: Changes indexing

When does the update take effect: After component rediscovery

Required actions for updates to take effect: Update the metric family or change the vendor certification priority.

IndexTag

Defines how to relate rows of the primary group (or MIB table) to rows in the secondary group. This element relates the rows by specifying attributes of both groups that must match.

IndexTag/Name

References the secondary group (or MIB table). The value of this element must equal the 'UseIndex' tag of the secondary attribute group that you are trying to relate with the primary one.

Can be updated: Yes

Possible values: The ‘UseIndex’ tag of the secondary attribute group.

Effect of updating: Changes indexing

When does the update take effect: After component rediscovery

Required actions for updates to take effect: Update the metric family or change the vendor certification priority.

IndexTag/PrimaryKeyExpression

Specifies an MVEL expression containing attributes of the primary attribute group or an attribute group corresponding to any of the previously defined IndexTag. The calculated value is matched up with the 'ThisTagKeyExpression'. If there is a match, the rows of both attribute groups (or MIB tables) are linked. Then, these attributes can be used together in an 'Expression' backing a destAttr (or Metric).

Can be updated: Yes

Possible values: A valid MVEL expression

Effect of updating: Changes indexing

When does the update take effect: After component rediscovery

Required actions for updates to take effect: Update the metric family or change the vendor certification priority.

IndexTag/ThisTagKeyExpression

Specifies an MVEL expression containing attributes of the secondary attribute group. The calculated value is matched up with the 'PrimaryKeyExpression'. If there is a match, the rows of both groups (or MIB tables) are linked. Then, these attributes can be used together in an 'Expression' backing a destAttr (or Metric).

Can be updated: Yes

Possible values: A valid MVEL expression

Effect of updating: Changes indexing

When does the update take effect: After component rediscovery

Required actions for updates to take effect: Update the metric family or change the vendor certification priority.

Import a Custom Vendor Certification

To put your vendor certification into effect, import it into Data Aggregator. There are two ways to import a vendor certification:

- Use a REST Client to Import a Custom Vendor Certification (see page 78).
- Use the Custom Vendor Certification Installer (see page 79)

Use a REST Client to Import a Custom Vendor Certification

Use any REST client to import a vendor certification. We recommend using the RESTClient to perform GET, POST, PUT, and other methods to the RESTful web services within CA Performance Management. If you do not have it installed, you can get the RESTClient GUI JAR file from http://code.google.com/p/rest-client/.

When using the RESTClient, consider the following information:

- To launch it, double-click the JAR file.
- When you POST XML, make sure that the Charset is set to UTF-8.

  To view and verify this setting, click the Edit Content-Type & Charset button.
- You can also auto-indent the Response Body, as follows:
  1. Click Options on the Tools menu.
  2. Select the Etc. tab.
  3. Select Auto-indent Response Body and click OK.

Follow these steps:

1. Enter http://*da-hostname*:8581/typecatalog/certifications/snmp as the URL.
2. Select POST on the Method tab.
3. Select 'application/xml' as the 'Body Content-type' in the Body settings.

   Important! Failing to set the Content-type results in a 404 error.
4. Copy and paste your custom vendor certification XML into the Body tab.
5. Run the method.

   Your vendor certification is imported. If no errors occur, the Status field in the HTTP Response section displays:

   ```
   HTTP/1.1 200 OK
   ```

Note: Any other return code indicates that an error occurred while updating your custom component. Fix the error and retry updating the component by doing another POST.

Important! To avoid possible data loss, always back up your deploy directory each time you create or update a vendor certification, metric family, or component.

Use the Custom Vendor Certification Installer

Use the CA Performance Management Certification Installer, which is on the Data Aggregator, to import a certification. The installer lets you install certifications in three ways:

- Single certification
- Collection of certifications in the form of a ZIP file
- All of the certifications in a directory (must point directly to the folder containing the certification)

Follow these steps:

1. Download and run the installer.

   The installer is available at the following URL:
   `http://<DA>:8581/cert/install.htm`

   <DA>

   The hostname of your Data Aggregator.

   Note: The default REST port is 8581.

2. Select a language for the installer.

3. Enter the following information when the corresponding prompts appear:

   - DA Hostname
   - DA REST port

4. Select whether you want to install a single certification, a collection (ZIP) of certifications, or all of the certifications in a directory.

   Select the folder in which your certifications are located when installing from a directory. The certification installer does not search any sub-directories for certifications.

5. The installer checks the version of each certification during the installation process.

   If the Data Aggregator indicates that a certification is already installed, the certification installer tries to upgrade the certification. If the installer fails to upgrade the certification, an error appears to specify the reason for the failure.

6. When the installer has finished installing certifications, any encountered errors appear.

   You can review the error details in the log files that the installer creates in the Home directory of your computer.

Note: You may also install a certification using the Linux command-line or UI installer.

Verify the Custom Vendor Certification Results

After you have imported your custom vendor certification XML, verify the results. In our example, we verify that the rPVCInfoCustom custom vendor certification has been imported successfully.

Follow these steps:

1. Enter the following URL in your REST client:

   `http://`*`da_hostname`*`:8581/typecatalog/certifications/snmp/`*`name`*

   name

   Is the name of your custom vendor certification. In this example, it is frPVCInfoCustom.

   A page displays the custom frPVCInfoCustom vendor certification XML.

2. Enter the following URL in the REST client:

   `http://`*`da-hostname`*`:8581/rest/vendorpriorities`

3. Search for your vendor certification in the XML document that is displayed.

4. Verify that the MetricFamilyVendorPriority XML lists both your custom vendor certification and metric family.

You are now ready to verify your custom vendor certification using the Data Aggregator user interface.

Follow these steps:

1. Verify that your custom vendor certification is available in Data Aggregator:

   a. Click Vendor Certifications on the Monitoring Configuration menu for a Data Aggregator data source.

      Verify that your custom vendor certification is listed (or search for your vendor certification).

   b. Select your custom vendor certification from the list of vendor certifications.

      Verify that your custom metric family is shown in the Metric Families view.

c. Click your metric family in the Metric Families view.

d. Click the Vendor Certification Priorities tab.

   Verify that your custom vendor certification is associated with the All Manageable Devices device collection.

2. Configure monitoring profiles and device collections for the metric family that your custom vendor certification supports:

   a. Verify that a discovery creates the expected and appropriate items.

   b. Verify that the items poll and collect the metric data you specified in your metric family XML.

   c. Verify that the collected data is correct.

   d. Verify that the item data synchronizes correctly with CA Performance Center, if the item synchronization is configured.

   e. Run a discovery again and verify that the information updates correctly.

Filtering Support

The Filter tag of the ExpressionGroup section is used to apply filtering on the component discovery process.

Continuing with our example, you want to monitor only active Frame Relay PVCs. You can do this by applying the filter tag to your frPVCInfoCustom custom vendor certification to monitor these specific components. Filters are added to the ExpressionGroup.

Note: You can use MVEL syntax in Filter expressions. For more information on the MVEL syntax, vendor certification utility functions, and vendor certification global variables, see the appendix in this guide.

Here is an example of what the updated frPVCInfoCustom custom vendor certification looks like. The changes that we made in this procedure are indicated in bold.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Auto-generated by the type catalog local manager.-->
<DataModel namespace="http://im.ca.com/certifications/snmp"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="SNMPCertificationFacet.xsd">
  <FacetType name="frPVCInfoCustom"
descriptorClass="com.ca.im.core.datamodel.certs.CertificationFacet
DescriptorImpl">
    <Documentation>Frame Relay PVC Vendor
Certification</Documentation>
    <FacetOf namespace="http://im.ca.com/core" name="Item" />
    <AttributeGroup name="AttributeGroup" external="true"
list="true">
      <Documentation />
      <Attribute name="INDEX" type="ObjectID">
        <Documentation />
        <Source>1.3.6.1.2.1.10.32.2.1.4</Source>
        <IsIndex>true</IsIndex>
        <IsKey>false</IsKey>
        <NeedsDelta>false</NeedsDelta>
      </Attribute>
      <Attribute name="frCircuitReceivedFECNs" type="Long">
        <Documentation />
        <Source>1.3.6.1.2.1.10.32.2.1.4</Source>
        <IsIndex>false</IsIndex>
        <IsKey>true</IsKey>
        <NeedsDelta>true</NeedsDelta>
      </Attribute>
      <Attribute name="frCircuitReceivedBECNs" type="Long">
        <Documentation />
        <Source>1.3.6.1.2.1.10.32.2.1.5</Source>
        <IsIndex>false</IsIndex>
        <IsKey>true</IsKey>
        <NeedsDelta>true</NeedsDelta>
      </Attribute>
      <Attribute name="frCircuitSentFrames" type="Long">
        <Documentation />
        <Source>1.3.6.1.2.1.10.32.2.1.6</Source>
        <IsIndex>false</IsIndex>
        <IsKey>true</IsKey>
        <NeedsDelta>true</NeedsDelta>
      </Attribute>
      <Attribute name="frCircuitSentOctets" type="Long">
        <Documentation />
        <Source>1.3.6.1.2.1.10.32.2.1.6</Source>
        <IsIndex>false</IsIndex>
        <IsKey>true</IsKey>
        <NeedsDelta>true</NeedsDelta>
      </Attribute>
      <Attribute name="frCircuitReceivedFrames" type="Long">
        <Documentation />
        <Source>1.3.6.1.2.1.10.32.2.1.8</Source>
       <IsIndex>false</IsIndex>
       <IsKey>true</IsKey>
       <NeedsDelta>true</NeedsDelta>
      </Attribute>
      <Attribute name="frCircuitReceivedOctets" type="Long">
        <Documentation />
        <Source>1.3.6.1.2.1.10.32.2.1.9</Source>
        <IsIndex>false</IsIndex>
        <IsKey>true</IsKey>
        <NeedsDelta>true</NeedsDelta>
      </Attribute>
      <Attribute name="frCircuitState" type="int">
        <Documentation />
        <Source>1.3.6.1.2.1.10.32.2.1.3</Source>
        <IsIndex>false</IsIndex>
        <IsKey>false</IsKey>
        <NeedsDelta>false</NeedsDelta>
      </Attribute>
     </AttributeGroup>
     <Protocol>SNMP</Protocol>
     <DisplayName>Frame Relay PVC Certification</DisplayName>
     <Expressions>
       <ExpressionGroup
destCert="{http://im.ca.com/normalizer}frPVCInfo"
name="frPVCInfoDS">
         <Filter>(frCircuitState==2)</Filter>
         <Expression destAttr="Indexes">INDEX</Expression>
         <Expression destAttr="Names">"Frame Relay " +
INDEX</Expression>
         <Expression
destAttr="FECNIn">frCircuitReceivedFECNs</Expression>
         <Expression
destAttr="BECNIn">frCircuitReceivedBECNs</Expression>
         <Expression
destAttr="FramesIn">frCircuitReceivedFrames</Expression>
         <Expression
destAttr="FramesOut">frCircuitSentFrames</Expression>
         <Expression
destAttr="BytesIn">frCircuitReceivedOctets</Expression>
         <Expression
destAttr="BytesOut">frCircuitSentOctets</Expression>
         <Expression
destAttr="BitsIn">frCircuitReceivedOctets*8</Expression>
```

```
         <Expression
destAttr="BitsOut">frCircuitSentOctets*8</Expression>
       </ExpressionGroup>
     </Expressions>
     <MIB>RFC1315-MIB</MIB>
    </FacetType>
</DataModel>
```

We are assuming that we made these changes after we imported the custom vendor certification. Therefore, after you make the changes, update the vendor certification.

More information:

Update a Custom Vendor Certification (see page 90)

Multi-MIB Table Support

When certifying a vendor MIB, situations exist where you must collect the raw data for a particular metric family from two or more tables.

CA Performance Management provides support for vendor certifications that require access to multiple MIB tables. This support is based on enhancements to the XML document structure of a single-table certification, which allow joining data that is collected from multiple tables using a common key (index). Continuing with our example, you want to monitor only active Frame Relay PVCs. Modify your frPVCInfoCustom custom vendor certification to add multi-MIB table support.

In this example, the Frame Relay PVCs can be named using a combination of the ifName MIB object in the ifXTable and the frCircuitDlci object that provides the data link connection identifier (DLCI) for this PVC. This kind of naming convention is useful for determining which Frame Relay interface the PVC is layered onto.

Modify your custom vendor certification, to add the following information:

- Add a *new* attribute to the existing AttributeGroup to represent the frCircuitDlci MIB object.
- The ifName MIB object that you want to use is coming from a MIB that is not included in your custom vendor certification. Add a new AttributeGroup (in this case, ifXTable), and then add the new attribute (ifName).

Here is an example of what the updated frPVCInfoCustom custom vendor certification looks like. The changes that we made in this procedure are indicated in bold:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Auto-generated by the type catalog local manager.-->
<DataModel namespace="http://im.ca.com/certifications/snmp"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="SNMPCertificationFacet.xsd">
   <FacetType name="frPVCInfoCustom"
descriptorClass="com.ca.im.core.datamodel.certs.CertificationFacet
DescriptorImpl">
     <Documentation>Frame Relay PVC Vendor
Certification</Documentation>
     <FacetOf namespace="http://im.ca.com/core" name="Item" />
     <AttributeGroup name="ifXTableGroup" external="true"
list="true">
     <Documentation>This pulls data from the ifXTable so that the
ifName corresponding to the PVC can be referenced</Documentation>
     <UseIndex>ifXIndexTag</UseIndex>
     <Attribute name="ifXTableIndex" type="ObjectID">
       <Documentation />
       <IsKey>false</IsKey>
       <IsIndex>true</IsIndex>
       <Source>1.3.6.1.2.1.31.1.1.1.1</Source>
       <Polled>false</Polled>
     </Attribute>
     <Attribute name="ifName" type="OctetString">
       <Documentation />
       <IsKey>false</IsKey>
       <IsIndex>false</IsIndex>
       <Source>1.3.6.1.2.1.31.1.1.1.1</Source>
       <Polled>false</Polled>
      </Attribute>
     </AttributeGroup>
     <IndexTagList>
       <PrimaryTag>PVCIndexTag</PrimaryTag>
       <IndexTag>
         <Name>ifXIndexTag</Name>

<PrimaryKeyExpression>snmpOIDParser(INDEX,1,1)</PrimaryKeyExpressi
on>

<ThisTagKeyExpression>ifXTableIndex</ThisTagKeyExpression>
       </IndexTag>
     </IndexTagList>
     <AttributeGroup name="AttributeGroup" external="true"
list="true">
       <Documentation />
       <UseIndex>PVCIndexTag</UseIndex>
       <Attribute name="INDEX" type="ObjectID">
```

```
  <Documentation />
  <Source>1.3.6.1.2.1.10.32.2.1.4</Source>
  <IsIndex>true</IsIndex>
  <IsKey>false</IsKey>
  <NeedsDelta>false</NeedsDelta>
</Attribute>
<Attribute name="frCircuitReceivedFECNs" type="Long">
  <Documentation />
  <Source>1.3.6.1.2.1.10.32.2.1.4</Source>
  <IsIndex>false</IsIndex>
  <IsKey>true</IsKey>
  <NeedsDelta>true</NeedsDelta>
</Attribute>
<Attribute name="frCircuitReceivedBECNs" type="Long">
  <Documentation />
  <Source>1.3.6.1.2.1.10.32.2.1.5</Source>
  <IsIndex>false</IsIndex>
  <IsKey>true</IsKey>
  <NeedsDelta>true</NeedsDelta>
</Attribute>
<Attribute name="frCircuitSentFrames" type="Long">
  <Documentation />
  <Source>1.3.6.1.2.1.10.32.2.1.6</Source>
  <IsIndex>false</IsIndex>
  <IsKey>true</IsKey>
  <NeedsDelta>true</NeedsDelta>
</Attribute>
<Attribute name="frCircuitSentOctets" type="Long">
  <Documentation />
  <Source>1.3.6.1.2.1.10.32.2.1.6</Source>
  <IsIndex>false</IsIndex>
  <IsKey>true</IsKey>
  <NeedsDelta>true</NeedsDelta>
</Attribute>
<Attribute name="frCircuitReceivedFrames" type="Long">
  <Documentation />
  <Source>1.3.6.1.2.1.10.32.2.1.8</Source>
  <IsIndex>false</IsIndex>
  <IsKey>true</IsKey>
  <NeedsDelta>true</NeedsDelta>
</Attribute>
<Attribute name="frCircuitReceivedOctets" type="Long">
  <Documentation />
  <Source>1.3.6.1.2.1.10.32.2.1.9</Source>
  <IsIndex>false</IsIndex>
  <IsKey>true</IsKey>
  <NeedsDelta>true</NeedsDelta>
 </Attribute>
 <Attribute name="frCircuitState" type="int">
```

```
          <Documentation />
          <Source>1.3.6.1.2.1.10.32.2.1.3</Source>
          <IsIndex>false</IsIndex>
          <IsKey>false</IsKey>
          <NeedsDelta>false</NeedsDelta>
        </Attribute>
        <Attribute name="frCircuitDlci" type="int">
          <Documentation />
          <Source>1.3.6.1.2.1.10.32.2.1.2</Source>
          <IsIndex>false</IsIndex>
          <IsKey>false</IsKey>
          <NeedsDelta>false</NeedsDelta>
        </Attribute>
      </AttributeGroup>
      <Protocol>SNMP</Protocol>
      <DisplayName>Frame Relay PVC Certification</DisplayName>
      <Expressions>
       <ExpressionGroup
destCert="{http://im.ca.com/normalizer}frPVCInfo"
name="frPVCInfoDS">
       <Filter>(frCircuitState==2)</Filter>
       <Expression destAttr="Indexes">INDEX</Expression>
       <Expression destAttr="Names">isdef(ifName)?
(isdef(frCircuitDlci) ? ifName + " DCLI:" + frCircuitDlci : "Frame
Relay " + INDEX) : "Frame Relay " + INDEX</Expression>
       <Expression
destAttr="FECNIn">frCircuitReceivedFECNs</Expression>
       <Expression
destAttr="BECNIn">frCircuitReceivedBECNs</Expression>
       <Expression
destAttr="FramesIn">frCircuitReceivedFrames</Expression>
       <Expression
destAttr="FramesOut">frCircuitSentFrames</Expression>
       <Expression
destAttr="BytesIn">frCircuitReceivedOctets</Expression>
       <Expression
destAttr="BytesOut">frCircuitSentOctets</Expression>
       <Expression
destAttr="BitsIn">frCircuitReceivedOctets*8</Expression>
       <Expression
destAttr="BitsOut">frCircuitSentOctets*8</Expression>
      </ExpressionGroup>
     </Expressions>
    <MIB>RFC1315-MIB</MIB>
   </FacetType>
</DataModel>
```

We are assuming that we made these changes after we imported the custom vendor certification. Therefore, after you make the changes, update the vendor certification (see page 90).

AttributeGroup (Multiple MIB Tables)

Each table must go into its own AttributeGroup section. Each Attribute on that table is added as a child of that AttributeGroup.

Refer to these sections for the following information:

- The AttributeGroup information – Details about the XML elements that are used to define primary and secondary table attributes.
- The UseIndex and IndexTagList information – Details about the XML elements for joining the primary and secondary attribute groups.

In this scenario, the primary attribute group represents the table that you want to “extend” with more information. The secondary attribute group contains the “extension” information for the primary one.

The primary AttributeGroup contains an Attribute identifying the MIB table variable serving as the common “key” into the secondary AttributeGroup.

The secondary AttributeGroup includes the Attribute definitions for all MIB table variables carrying the “extension” information for the primary table. In addition, there is an Attribute identifying the variable matching the common “key” from the primary AttributeGroup.

More information:

AttributeGroup (Vendor Certification) (see page 66)
UseIndex (see page 89)
IndexTagList (Multiple MIB Tables) (see page 89)

UseIndex

Each AttributeGroup is given a UseIndex tag. The UseIndex tag lets you group OIDs under a common name. This common name is then associated with a given variable serving as the common key (index) into the respective MIB table.

The following information summarizes the XML elements to customize:

AttributeGroup/UseIndex

Uniquely identifies the primary and secondary tag name (respectively) that is used in the IndexTagList section.

Recommendation: Set to the value of the AttributeGroup/name property.

More Information:

AttributeGroup (Multiple MIB Tables) (see page 88)

IndexTagList (Multiple MIB Tables)

The IndexTagList section is a mechanism to relate two attribute groups (or MIB tables) with different indexes. When the groups are related, one item has multiple index IDs from multiple tables.

The IndexTagList section contains all of the join information, including an IndexTag section for every secondary attribute group.

IndexTagList/PrimaryTag

Defines the primary attribute group (or MIB table). Set to the value of the UseIndex property of the primary AttributeGroup.

IndexTag/Name

Defines the secondary attribute group. Set to the value of the UseIndex property of the secondary AttributeGroup.

IndexTag/PrimaryKeyExpression

Specifies the expression to generate the common key in the primary table. Consider using the MVEL functions to derive a common key from the designated primary table index Attribute.

IndexTag/ThisTagKeyExpression

Specifies the expression to generate the common key in the secondary table. Consider using the MVEL functions to derive a common key from the designated secondary table index Attribute.

The multitable approach supports the chaining of more than two tables. Two types of relationships exist in multiple table joins:

- **Primary –> Secondary #1, Primary –> Secondary #2**
  Ordering of the secondary tables does not matter in an index tag list.
- **Primary –> Secondary #1 –> Secondary #2**
  List secondary table #1 before secondary table #2 because of the way tables are merged.

One or more rows in the primary table can legally map to the same row in the secondary table. Keys on the secondary table are searched in order, and the first match wins.

More information:

Multi-MIB Table Support (see page 84)
AttributeGroup (Multiple MIB Tables) (see page 88)

Update a Custom Vendor Certification

You can update an existing custom vendor certification. Continuing with our example, you previously made a few changes to your custom metric family and to your custom vendor certification.

To your metric family, you added metrics. To your vendor certification, you added a MIB table, and you included filtering to your custom vendor certification. Now, update the vendor certification for the changes to take effect.

Note: For information about the effects of updating a tag or attribute, review the custom vendor certification XML details. In particular, review the specific attribute descriptions.

Follow these steps:

1. Enter the following addresss In the URL field:

   http://*da_hostname*:8581/typecatalog/certifications/snmp/*CustomVendorCertName*

   CustomVendorCertName

   Is the name of the custom vendor certification to update. For our example, the vendor certification is named frPVCInfoCustom.

2. To verify that the vendor certification exists at that URL, select GET in the Method tab.
3. After you verify that the vendor certification exists at the URL, select PUT in the Method tab.

4. Copy and paste your updated custom metric family XML into the Body tab (Edit field) and set the Content-type to application/xml.

   Important! Failing to set the Content-type results in a 404 error.

5. Click the Go button next to the URL field.

   Your custom vendor certification is updated. If no errors occur, the Status field in the HTTP Response section displays the following message:

   ```
   HTTP/1.1 200 OK
   ```

   Note: Any other return code indicates that an error occurred while updating your custom vendor certification. Fix the error and retry updating the vendor certification by doing another PUT.

   Important! To avoid possible data loss, always back up your deploy directory each time you create or update a vendor certification, metric family, or component.

More information:

Understanding the Vendor Certification XML Structure (see page 62)
Update a Custom Metric Family (see page 58)

Appendix A: Vendor Certification Expressions: Expression Operators, Functions, and Global Variables

This section describes some of the expression operators, functions, and global variables you can use in your vendor certification expressions.

You can use MVEL syntax in vendor certification expressions. MVEL is a publicly available embeddable Expression Language for Java environments that has a syntax close to Java. MVEL supports expressions similar to Java expressions.

You can build expressions using operators, you can use braces to control precedence, and you can terminate statements by semi-colons. For a detailed reference of the MVEL language, see http://mvel.codehaus.org.

The MVEL language has vendor certification utility functions and vendor certification global variables that you can also use in vendor certification expressions.

To study the use of functions, operators, and global variables, you can use the Vendor Certification tab in CA Performance Center.

This section contains the following topics:

Expression Operators

This section describes operators that you can use in vendor certification expressions.

You can use MVEL syntax in vendor certification expressions. MVEL is a publicly available embeddable Expression Language for Java environments that has a syntax close to Java. MVEL supports expressions similar to Java expressions.

You can build expressions using operators, use braces to control precedence, and terminate statements by semi-colons. For a detailed reference of the MVEL language, see http://mvel.codehaus.org.

The following table summarizes the available operators:

Note: In XML documents, use the XML Named Entities (XNE) presentation.

Operator	XNE	Description	Example
=		Assign	a = 1
==		Equals	"fred" == "fred"
!=		Not Equals	"fred" != "tom"
>	&gt;	Greater Than	1 > 0 is true
<	&lt;	Less Than	0 < 1 is true
>=		Greater Than or Equal	1 >= 0 is true
<=		Less Than or Equal	1 <= 1 is true
contains		Verify if the value on the left side contains the value on the right	"tomcat" contains "cat"
isdef		Tests whether a variable is defined	isdef a
+		Add	1 + 1
+		Concatenate	"one " + "two"
-		Minus	2 - 1
*		Multiply	2 * 2
/		Divide	4 / 2
%		Modulus	5 % 2
&&	&amp;&amp;	Logical AND	(x>-1) && (x<1)
||		Logical OR	(x<-1) || (x>1)
&	&amp;	AND bit operation	17 & 0xF
|		OR bit operation	4 | 1
^		Exclusive OR bit operation	5 ^ 1
!		Negate	! True
?		Ternary operator	age > 17 ? "allow" : "deny"

Functions and Global Variables

This section describes some of the expression operators, functions, and global variables you can use in your vendor certification expressions.

You can use MVEL syntax in vendor certification expressions. MVEL is a publicly available embeddable Expression Language for Java environments that has a syntax close to Java. MVEL supports expressions similar to Java expressions.

You can build expressions using operators, you can use braces to control precedence, and you can terminate statements by semi-colons. For a detailed reference of the MVEL language, see http://mvel.codehaus.org.

The MVEL language has vendor certification utility functions and vendor certification global variables that you can also use in vendor certification expressions.

To study the use of functions and global variables, you can use the Vendor Certification tab in CA Performance Center.

availabilityWithSysUptime Function

This function calculates availability as a percentage using sysUptime and the poll duration, granting a grace period.

Syntax

This function has the following format:

`Object availabilityWithSysUptime (Long` *`sysUpTime`*`, Long` *`duration`*`)`

Parameters

sysUpTime

The time (in centiseconds) since the network management portion of the system was last reinitialized.

duration

The poll duration time in seconds. Use the global variable _rspDuration. See the Advanced example for more information.

Return Values

Returns the availability as a percentage (0–100), or returns "null" when invalid data is passed.

Examples

The following expression produces the following result for a sysUpTime of 30000 and a poll duration of 300:

Expression:

```
availabilityWithSysUptime (sysUpTime, duration)
```

Result:

```
100
```

The same expression produces the following result for a sysUpTime of 6000 and a poll duration of 300:

Result:

```
20
```

The same expression produces the following result for a sysUpTime of 30005 and a poll duration of 300:

Result:

```
100
```

Advanced Example

The following expression is taken from “System Statistics” Vendor Certification:

```
Availability=availabilityWithSysUptime(sysUpTime,_rspDuration)
```

mapModel Function

This function uses the value of an objectID (sysObjectID) and maps the system OID to a model name string. Use this function to certify devices.

Syntax

This function has the following format:

```
String  mapModel ( ObjectID sysObjectID )
```

Parameters

sysObjectID

The object ID value to parse.

Return Values

Returns the string containing the mapped model name.

Examples

The following expression produces the following result for an OID value of 1.3.6.1.4.1.9.1.223:

Expression:

```
mapModel (oid )
```

Result:

```
Cisco7204VXR
```

The same expression produces the following result for an OID value of 1.3.6.1.4.1:

Result:

```
Unknown 1.3.6.1.4.1
```

Advanced Example

The following expression is taken from “System Statistics” Vendor Certification:

```
Model=mapModel (sysObjectID)
```

mapVendor Function

This function uses the value of an objectID (sysObjectID) and maps the system OID to a vendor name string. Use this function to find the vendor of a device.

Syntax

This function has the following format:

```
String mapVendor( ObjectID sysObjectID )
```

Parameters

sysObjectID

The object ID value to parse.

Return Values

Returns the string containing the mapped vendor name. If a vendor is not found, returns "".

Examples

The following expression produces the following result for an OID value of 1.3.6.2.1.2.2636.0:

Expression:

```
mapVendor (oid )
```

Result:

```
Juniper
```

The same expression produces the following result for an OID value of 1.3.6.2.1.2.1234567.0:

Result:

```
Unknown
```

Advanced Example

The following expression is taken from “System Statistics” Vendor Certification:

```
Model=mapVendor (sysObjectID)
```

snmpConstArrayMap Function

This function maps a value (index) to a set of constant values (array). If necessary, this function rounds the input value to nearest integer value. Then, it uses the integer value as an index to the set of constant values (array) that are shown as c0, c1, up to cn-1. The c values must be integers. This function checks these values when the expression is parsed and returns c*x*. If the value is not in the domain from 0 to n-1 (inclusive), the result is 0 (without an error message). Use this function to certify devices.

Syntax

This function has the following format:

```
Integer  snmpConstArrayMap(Double index, Integer[] array)
```

Parameters

index

A Double value, which is used as an index into the array.

array

Any range of integer values.

Return Values

Returns an integer value from the array. An index value of null will return null.

Examples

The following expression produces the following result for an index of 2 and an array of {5, 6, 7, 8, 9, 4}:

Expression:

`snmpConstArrayMap (`*`index`*`, `*`array`*`)`

Result:

7

The following expression produces the following result for an index of 4.88 and an array of {5, 6, 7, 8, 9, 4}:

Expression:

`snmpConstArrayMap (`*`value`*`, `*`array`*`)`

Result:

4

Advanced Example

The following expression is taken from “Generic Modem” Vendor Certification:

```
SpeedOut=snmpConstArrayMap(mdmCsFinalTxLinkRate,{0,110,300,600,1200,2400,4800,720
0,9600,12000,14000,16000,19000,38000,75,450,0,57000,21000,24000,26000,28000,0,310
00,33000,25333,26666,28000,29333,30666,32000,33333,34666,36000,37333,38666,40000,
41333,42666,44000,45333,46666,48000,49333,50666,52000,53333,54666,56000,57333,586
66,60000,61333,62666,64000})
```

mvelInfo Function

This function populates the INFO level of the karaf.log file with the input parameters. Use this function to log the polled values from a report that returns a result that you believe is incorrect.

The poll log of the Data Collector only shows the values of the polled attributes, while the report only shows the result of the calculation. The mvelInfo function allows you to view the input poll values to determine where the calculation went wrong.

Syntax

This function has the following format:

```
String mvelInfo (Array objects)
```

Parameters

objects

The object array is logged under the INFO level in the karaf.log file of the Data Collector.

Return Values

Null

Examples

The following expression logs cpmCPUTotal5minRev in the karaf.log file.

Expression:

```
mvelInfo([“cpmCPUTotal5minRev=”, cpmCPUTotal5minRev])
```

Result:

Null

Result (karaf.log):

MVEL info: cpmCPUTotal5minRev=15

Advanced Example

```
mvelInfo([“cpmCPUTotal5minRev=”, cpmCPUTotal5minRev, “ cpmCPUTotal10minRev=”,
cpmCPUTotal10minRev]); cpmCPUTotal10minRev;
```

Result:

12

Result (karaf.log):

MVEL info: cpmCPUTotal5minRev=15 cpmCPUTotal10minRev=12

mvelWarn Function

This function populates the WARN level of the karaf.log file with the input parameters. Use this function to log the polled values from a report that returns a result that you believe is incorrect.

The poll log of the Data Collector only shows the values of the polled attributes, while the report only shows the result of the calculation. The mvelWarn function allows you to view the input poll values to determine where the calculation went wrong.

Syntax

This function has the following format:

`String mvelWarn (Array` *`objects`*`)`

Parameters

objects

The object array is logged under the WARN level in the karaf.log file of the Data Collector.

Return Values

Null

Examples

The following expression logs cpmCPUTotal5minRev in the karaf.log file.

Expression:

```
mvelWarn([“cpmCPUTotal5minRev=”, cpmCPUTotal5minRev])
```

Result:

Null

Result (karaf.log):

MVEL warn: cpmCPUTotal5minRev=15

Advanced Example

```
mvelWarn([“cpmCPUTotal5minRev=”, cpmCPUTotal5minRev, “ cpmCPUTotal10minRev=”,
cpmCPUTotal10minRev]); cpmCPUTotal10minRev;
```

Result:

12

Result (karaf.log):

MVEL warn: cpmCPUTotal5minRev=15 cpmCPUTotal10minRev=12

mvelError Function

This function populates the ERROR level of the karaf.log file with the input parameters. Use this function to log the polled values from a report that returns a result that you believe is incorrect.

The poll log of the Data Collector only shows the values of the polled attributes, while the report only shows the result of the calculation. The mvelError function allows you to view the input poll values to determine where the calculation went wrong.

Syntax

This function has the following format:

`String mvelError (Array` *`objects`*`)`

Parameters

objects

The object array is logged under the ERROR level in the karaf.log file of the Data Collector.

Return Values

Null

Examples

The following expression logs cpmCPUTotal5minRev in the karaf.log file.

Expression:

```
mvelError([“cpmCPUTotal5minRev=”, cpmCPUTotal5minRev])
```

Result:

Null

Result (karaf.log):

MVEL error: cpmCPUTotal5minRev=15

Advanced Example

```
mvelError([“cpmCPUTotal5minRev=”, cpmCPUTotal5minRev, “ cpmCPUTotal10minRev=”,
cpmCPUTotal10minRev]); cpmCPUTotal10minRev;
```

Result:

12

Result (karaf.log):

MVEL error: cpmCPUTotal5minRev=15 cpmCPUTotal10minRev=12

mvelDebug Function

This function populates the DEBUG level of the karaf.log file with the input parameters. Use this function to log the polled values from a report that returns a result that you believe is incorrect.

The poll log of the Data Collector only shows the values of the polled attributes, while the report only shows the result of the calculation. The mvelDebug function allows you to view the input poll values to determine where the calculation went wrong.

Syntax

This function has the following format:

```
String mvelDebug (Array objects)
```

Parameters

objects

The object array is logged under the DEBUG level in the karaf.log file of the Data Collector.

Return Values

Null

Examples

The following expression logs cpmCPUTotal5minRev in the karaf.log file.

Expression:

```
mvelDebug([“cpmCPUTotal5minRev=”, cpmCPUTotal5minRev])
```

Result:

Null

Result (karaf.log):

MVEL debug: cpmCPUTotal5minRev=15

Advanced Example

```
mvelDebug([“cpmCPUTotal5minRev=”, cpmCPUTotal5minRev, “ cpmCPUTotal10minRev=”,
cpmCPUTotal10minRev]); cpmCPUTotal10minRev;
```

Result:

12

Result (karaf.log):

MVEL debug: cpmCPUTotal5minRev=15 cpmCPUTotal10minRev=12

mvelDebug Function

This function populates the TRACE level of the karaf.log file with the input parameters. Use this function to log the polled values from a report that returns a result that you believe is incorrect.

The poll log of the Data Collector only shows the values of the polled attributes, while the report only shows the result of the calculation. The mvelTrace function allows you to view the input poll values to determine where the calculation went wrong.

Syntax

This function has the following format:

```
String mvelTrace (Array objects)
```

Parameters

objects

> The object array is logged under the TRACE level in the karaf.log file of the Data Collector.

Return Values

Null

Examples

The following expression logs cpmCPUTotal5minRev in the karaf.log file.

Expression:

```
mvelTrace([“cpmCPUTotal5minRev=”, cpmCPUTotal5minRev])
```

Result:

Null

Result (karaf.log):

MVEL trace: cpmCPUTotal5minRev=15

Advanced Example

```
mvelTrace([“cpmCPUTotal5minRev=”, cpmCPUTotal5minRev, “ cpmCPUTotal10minRev=”,
cpmCPUTotal10minRev]); cpmCPUTotal10minRev;
```

Result:

12

Result (karaf.log):

MVEL trace: cpmCPUTotal5minRev=15 cpmCPUTotal10minRev=12

snmpCounter64 Function

This function evaluates two 32-bit numeric values and returns a value containing the 64-bit representation. Use this function to certify devices. The hiVal is shifted 32 bits left and the lowVal is added and the result is placed in a 64-bit return variable.

Syntax

This function has the following format:

```
Object snmpCounter64 (Long hiVal, Long lowVal)
```

Parameters

hiVal

The 32-bit numeric value representing the high-order bits.

lowVal

The 32-bit numeric value representing the low-order bits.

Return Values

Returns the 64-bit representation of the two 32-bit numeric values, or returns "null" when either 32-bit value input is null.

Examples

The following expression produces the following result for a hiVal of 88 and a lowVal of 558.

Expression:

```
snmpCounter64 (hiVal, lowVal)
```

Result:

```
377957122606
```

Advanced Example

The following expression is taken from “Cisco CBQos ClassMap” Vendor Certification. This certification contains many snmpMax examples:

```
PrePolicyPackets=snmpMax(0,snmpCounter64(cbQosCMPrePolicyPktOverflow,cbQosCMPrePo
licyPkt))
```

snmpGetUpSinceTime Function

This function returns the time that the system was turned on based on the number of seconds since the current epoch.

Syntax

This function has the following format:

```
snmpGetUpSinceTime(Long upTime)
```

Parameters

upTime

The number of seconds since the beginning of the current epoch. You can get the system uptime from the following OID: 1.3.6.1.2.1.1.3.0. Convert it into centiseconds before you pass it in.

Return Values

Returns the time that the device was powered on in the form of total seconds since the current epoch.

snmpMax Function

This function returns the larger of two 64-bit values. Use this function to certify devices.

Syntax

This function has the following format:

```
Object snmpMax(BigInteger val1, BigInteger val2)
```

Parameters

val1

The first 64-bit BigInteger value.

val2

The second 64-bit BigInteger value.

Return Values

Returns the maximum of the two BigInteger values passed in, or returns "null" when either BigInteger value input is null.

Examples

The following expression produces the following result for a val1 of 2^32 and a val2 of 10:

Expression:

```
snmpMax (val1, val2)
```

Result:

2^32

The same expression produces the following result for a val1 of 5864 and a val2 of 134556890:

Result:

```
134556890
```

Advanced Example

The following expression is taken from "Cisco CBQos ClassMap" Vendor Certification. This certification contains many snmpMax examples:

```
PrePolicyPackets=snmpMax(0,snmpCounter64(cbQosCMPrePolicyPktOverflow,cbQosCMPrePo
licyPkt))
```

snmpObjectIDToASCIIString Function

This function converts an SNMP OID value to its string representation. Any leading or trailing spaces are removed.

Syntax

This function has the following format:

```
snmpObjectIDToASCIIString( Object Id oid )
```

Parameters

oid

The object ID to convert to a string.

snmpOIDParser Function

This function uses the value of an objectID (OID) and parses out a subset of the OID based on the startIndex and endIndex values. The indexes are 1 based. If the endIndex is -1, then we go to the end of the OID. Use this function to certify devices.

Syntax

This function has the following format:

```
ObjectID snmpOIDParser( ObjectID OID, Integer startIndex, Integer endIndex )
```

Parameters

OID

The object ID (OID) value to parse.

startIndex

An integer value of the index at which to begin parsing.

endIndex

An integer value of the index at which to stop parsing.

Return Values

Returns the parsed subset ObjectID (OID).

Examples

The following expression produces the following result for an OID value of 1.2.3.4.5.6.7.8.9.10, a startIndex value of 1, and an endIndex value of 5:

Expression:

```
snmpOIDParser(oid, startIndex, endIndex )
```

Result:

```
1.2.3.4.5
```

The same expression produces the following result for an OID value of 1.2.3.4.5.6.7.8.9.10, a startIndex value of 6, and an endIndex value of -1:

Result:

```
6.7.8.9.10
```

Advanced Example

The following expression is taken from “Cisco CBQos ClassMap” Vendor Certification:

```
ItemUniqueIDs=snmpOIDParser(cbQosConfigIndex, 2, 2)
```

snmpOctetStringFloat Function

This function converts an SNMP octet string to a floating point value. Use this function to certify devices. An SNMP octet string is a seven-bit ASCII string.

Syntax

This function has the following format:

```
Object snmpOctetStringFloat(byte[] octetString)
```

Parameters

octetString

The SNMP octet string.

Return Values

Returns the converted string value, or returns "null" when the function cannot convert the string.

Examples

The following expression produces the following result for an octetString of {0x33, 0x33, 0x2E, 0x33, 0x33}:

Expression:

```
snmpOctetStringFloat (octetString)
```

Result:

```
33.33
```

The same expression produces the following result for an octetString of {0x36, 0x36, 0x36}:

Result:

```
666.0
```

snmpProtectedDiv Function

This function divides two Double values and returns the result of the division as a Double. If the dividend or divisor is null or 0.0 the return value is 0.0. Use this function to protect the expression from dividing with null or 0. Data Repository can contain null or zero values, such as when a poll fails. In this case you avoid a divide-by-zero exception by using this function.

Syntax

This function has the following format:

```
Double snmpProtectedDiv(Double val1, Double val2)
```

Parameters

val1

Is the dividend. This is a Double value (floating number) to be divided by val2. (*Double* is a Java data type.)

val2

Is the divisor. This is a Double value (floating number). (*Double* is a Java data type.)

Return Values

Returns the result of the division as a Double or 0.0 if the dividend or divisor is null or 0.0 (*Double* is a Java data type).

Examples

The following expression produces the following result for a val1 of 7.2 and val2 of 2:

Expression:

```
snmpProtectedDiv(val1, val2)
```

Result:

```
3.6
```

The following expression produces the following result for a val1 of 7.2 and val2 of null or 0.0:

Result:

```
0.0
```

Advanced Example

The following expression is taken from Vendor Certification:

```
Utilization=snmpProtectedDiv((cpuStatsUser + cpuStatsSys),(cpuStatsUser +
cpuStatsSys + (isdef(cpuStatsWait)?cpuStatsWait:0) + cpuStatsIdle))*100
```

snmpRound Function

This function rounds a numeric value to the nearest integer value.

Syntax

This function has the following format:

`Long snmpRound(Double dNumber)`

Parameters

dNumber

A Double value (floating number) to be rounded (*Double* is a Java data type).

Return Values

Returns a Long value, which is the nearest integer value to the value provided in dNumber (*Long* is a Java data type).

Examples

The following expression produces the following result for a dNumber of 3.5:

Expression:

`snmpRound(dNumber)`

Result:

`4`

The same expression produces the following result for a dNumber of 3.4:

Result:

3

Advanced Example

The following expression is taken from “Cisco IPSLA Jitter Precision Statistics” Vendor Certification:

```
PathAvailability=snmpRound(rttMonJitterStatsNumOfRTT / (rttMonJitterStatsNumOfRTT
+ rttMonJitterStatsPacketLossSD + rttMonJitterStatsPacketLossDS +
rttMonJitterStatsPacketOutOfSequence + rttMonJitterStatsPacketMIA +
rttMonJitterStatsPacketLateArrival + rttMonJitterStatsError +
rttMonJitterStatsBusies + 1/100) * 100)
```

snmpStringParser Function

This function was written for internal use only. It parses IP addresses received from a CA Application Insight Module (AIM). CA has only tested this function with an internal class; therefore, another type of class may not be supported.

Syntax

This function has the following format:

snmpStringParser(*Delimiter, Type to convert to, String to parse 1, String to parse 2*)

Parameters

Type to convert to

The type of class to which to parse the supplied strings.

Strings to parse 1 and 2

The IP address to parse. Two strings enable you to provide addresses in IPv4 and IPv6 format.

Return Values

Returns the converted value of the string, or returns "null" when the function cannot parse the string.

snmpSvcs Function

This function takes the values from sysObjectOID, sysService, and ipForwarding MIB variables of an agent and determines what services the SNMP agent supports. For example, Router/Switch/Repeater/Host could be a supported service, as defined in the SNMP MIB RFC 1213.

The return from the function is evaluated as follows because custom device types have high precedence over system ones:

- If sysObjectID OID is mapped in the DeviceTypes file, then the return services is from the file.
- If sysObjectID OID is not mapped in the DeviceTypes file, then the sysServices and ipForwarding is used to return the supported services.

Syntax

This function has the following format:

DeviceService[] snmpSvcs(ObjectID *sysObjectID*, Integer *sysServices*, Integer *ipForwarding*)

Parameters

sysObjectID

The object ID value to parse.

sysServices

An integer where each bit represents a different service, such as switch/repeater/host.

ipForwarding

An integer indicating whether this entity is acting as an IP gateway or IP host regarding the forwarding of datagrams. This entity receives the forwarded datagrams, but the forwarded datagrams are not addressed to this entity. Possible values are 1 (Forwarding) and 2 (notForwarding).

Return Values

Returns a list of one or more of the following device services:

- ROUTER
- REPEATER
- SWITCH
- HOST
- UNKNOWN_TYPE

Example

The following expression produces the following result for a sysServices value of 8, an ipForwarding value of 0, and sysObjectID not found in the DeviceTypes file:

Expression:

```
snmpSvcs(sysObjectOID,sysServices,ipForwarding)
```

Result:

```
DeviceService[HOST]
```

Advanced Example

The following expression is taken from the "System Statistics" Vendor Certification:

```
Services=snmpSvcs(sysObjectID,isdef(sysServices)?sysServices:0,isdef(ipForwarding
)?ipForwarding:0)
```

storePortReconfig Function

This function returns a string containing XML representing the values of ifNumber, ifTableLastChange, ifStackLastChange. You can use XML to track device changes and to rediscover interfaces on a device, when needed.

Syntax

This function has the following format:

```
String storePortReconfig ( Integer ifNumber, Long ifTableLastChange, Long
ifStackLastChange )
```

Parameters

ifNumber

The number of ports on the device.

ifTableLastChange

The date and time of the latest port table change in milliseconds, calculated from a start date and time of January 1, 1970 GMT.

ifStackLastChange

The date and time of the latest port stack change in milliseconds, calculated from a start date and time of January 1, 1970 GMT.

Return Values

Returns a string containing XML in the form that is shown in the following example.

Example

The following expression produces the following result for an ifNumber of 5, ifTableLastChange of 123456, and ifStackLastChange of 234567:

Expression:

```
storePortReconfig ( ifNumber, ifTableLastChange, ifStackLastChange )
```

Result:

```
<ReconfigData>
     <ReconfigValue name="ifNumber" value="5"/>
     <ReconfigValue name="ifTableLastChange" value="123456"/>
     <ReconfigValue name="ifStackLastChange" value="234567"/>
</ReconfigData>
```

Global Variables

Data Aggregator supports the following global variables:

_ rspDuration

A Long (Java data type) containing the duration of the current poll cycle in seconds.

Note: The “System Statistics” Vendor Certification contains an example of the use of _rspDuration.

_rspTimestamp

A Long (Java data type) containing the timestamp at which the current poll cycle started in milliseconds since January 1, 1970 GMT.

Appendix B: Troubleshooting

This section contains the following topics:

Troubleshooting: Failure to Create Vendor Certification

Symptom:

I tried to create a vendor certification and received an error message that it failed.

Solution:

Open the karaf log files in the Data Aggregator installation directory, and follow these steps:

1. Look for the MIB name string or the name of the metric family you selected.
2. Review the stack trace of the exception and look for the CertManagerException and the reason for the error. The reason for the error follows the exception.

   Example: The expression parser did not expect the token after ++, as shown:

   ```
   Caused by: com.ca.im.dm.certmgr.interfaces.CertManagerException: Tech Cert:
   {http://im.ca.com/normalizer}NormalizedCPUInfo, Unable to compile expression:
   [Error: expected end of statement but encountered: e]
   [Near : {... stemID ++ extremeSystemBoardID ....}]
   ```

3. Fix the error based on the reason provided. Verify that the following requirements are met:
   - The expression group must not contain a mix of scalar and table entries.
   - Expressions must contain valid syntax.
   - At least one expression is defined for a metric family variable.
   - At least two metric family variables are defined; specifically Names and Indexes are required (except for scalar-only metrics).
   - The vendor certification variable used in the expression must be from the chosen MIB table (valid in the user interface).

Troubleshooting: A Metric Family is Not Supported

Symptom:

I created a monitoring profile to poll metric families on a collection of devices. However, in the Polled Metric Families table, one of those metric families has a status of 'Unsupported.'

Solution:

To correct the problem, follow these steps:

1. Verify that the polled device responds to SNMP queries.
2. Navigate to the unsupported metric family by clicking it.
3. Verify that a vendor certification supports the metric family. If no vendor certification is defined, create a custom vendor certification.
4. Verify that all key vendor certification attributes are supported on the device. If all key vendor certificate attributes are supported, navigate back to the device, select the metric family for which you added a custom vendor certification, and click Update Metric Family.

   Your device configuration is updated.

Troubleshooting: A Metric Family is Incomplete

Symptom:

I successfully imported a custom metric family, but found a defective metric definition afterwards. For example, the <Name> property has a maximum length of 32 characters. If this limit is exceeded, it can cause synchronization problems.

Solution:

Carefully delete the custom metric family, as follows:

1. Locate the IMDataAggregator/apache-karaf-2.3.0/deploy directory.
2. Delete the XML files that were created and deployed for the metric family. They are named as follows:
   - im.ca.com-normalizer-<technology>.xml
   - im.ca.com-inventory-<technology>.xml

   If applicable, also delete the file that was created for the vendor certification:

   - im.ca.com-certifications-snmp-<vendor>.xml

3. Restart Data Aggregator by running the following command:

   ```
   service dadaemon restart
   ```

   After Data Aggregator restarts, verify that the previously imported metric family or vendor certification does not appear in CA Performance Center. Also, all previously discovered components for this custom certification are deleted.

4. Click Admin, Data Sources in CA Performance Center.
5. Select Data Aggregator and click the Resync button.

   The components for remaining metric families synchronize between Data Aggregator and CA Performance Center.

6. Edit and correct your custom metric family XML file.
7. Import your corrected metric family XML file.

Troubleshooting: A Vendor Certification Expression is Erroneous

Symptom:

The MVEL compiler may not give an evaluation exception (error) for bad expressions. This situation can happen for somesyntax errors, including but not limited to missing or open parentheses, and multiple asterisks.

The incorrect expression is compiled and no error condition is visible until an expression evaluation is performed with the appropriate variables. Database columns that are the target for the intended expression are not populated.

Solution:

Turn on debug logging for the ExpressionEvaluator using the following steps:

1. Locate the IMDataAggregator/apache-karaf-2.3.0/etc directory.
2. Open the org.ops4j.pax.logging.cfg file and create the following entry:

   ```
   log4j.logger.com.ca.im.core.expressionevaluator=DEBUG
   ```

3. Restart Data Aggregator by running the following command:

   ```
   service dadaemon restart
   ```

4. Search for evaluation exceptions in the karaf.log file in the IMDataAggregator/apache-karaf-2.3.0/data/log directory.