CA Performance Management Data Aggregator

Administration Using REST Web Services Guide

2.4.1

CA Technologies Product References

This document references the following CA Technologies products:

- CA Performance Management Data Aggregator (Data Aggregator)
- CA Performance Management Data Collector (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 REST Web Services

Data Aggregator uses REST web services to manage administrative operations, such as retrieve data or manage relationships between profiles and tenants or groups. These REST web services support the CA Performance Center user interface. Alternatively, you can perform operations using the API. As an administrator, you can use any REST client tool to perform requests or an HTTP tool that can send requests and can receive responses.

There are three types of REST web services:

OpenAPI

Designed as a public API for customers who want to access Data Aggregator configuration information and metric data for their own purposes. The OpenAPI is designed to stay constant over time, even if the internal data storage structure changes.

Important: The OpenAPI uses a QueryBuilder interface. This release of the interface is in BETA, and we do *NOT* recommend using the interface in production environments. There are limitations in scalability and authorization options (single user and password).

Data-Driven

Used by the CA Performance Center UI to read and modify Data Aggregator configuration information, such as Monitoring Profiles and Groups.

Generic

Used by the CA Performance Center UI to manage metric families and limited SNMP vendor certification support. Generic REST web services are self-filtering and do not use an argument within the URL to manage relationships.

Basic REST Operations

Endpoints are types of items, such as groups, monitoring profiles, tenants, and device certifications. Use specific endpoints with the following request methods or basic operations to return a list of results, or create, update, or delete an item.

Note: Individual metric family items and SNMP vendor certification items are specified using name instead of ID.

Important! Set the Context-type files to **application/xml** when you perform operations using Data Aggregator REST web services.

Basic operations include:

GET http://.../*endpoint*

Returns a list of all items of the specified type. The getlist.xsd schema defines the format for the return data.

GET http://.../*endpoint*/[*id* | *name*]

Returns the details for a single item with the specified ID or certification name. The XSD schema defines the format for the return data.

POST http://.../*endpoint*

Creates an object of the specified type with specified facets. The XSD schema defines the format for the return data.

Note: When creating (using POST), it is possible to create objects with the same name but different IDs. This convention is allowed because it is often valid when the objects are scoped to different tenants. When creating or updating objects, name the objects according to what the user interface can represent.

PUT http://.../*endpoint*/[*id* | *name*]

Updates the attributes of the specified item. The update.xsd schema defines the format and expected fields.

Note: When updating (using PUT), it is possible to create objects with the same name but different IDs. This convention is allowed because it is often valid when the objects are scoped to different tenants. When creating or updating objects, name the objects according to what the user interface can represent.

DELETE http://.../*endpoint*/[*id* | *name*]

Deletes the item that is specified using the ID or certification name.

Chapter 2: About OpenAPI

Manage OpenAPI Credentials

By default, access to the OpenAPI is disabled. To enable access, set the username and password for the OpenAPI. The username and password are stored in an external file, and cannot be changed using the OpenAPI interface.

Follow these steps:

1. Open the following file:
   `/opt/IMDataAggregator/apache-karaf-2.3.0/etc/com.ca.im.odata.auth.OpenAPIAuthenticationFilter.cfg`
2. Change the username and password.

   Delete any spaces following the last character. If there is a space, enter it for the password to work.
3. Save the file.

   Start using the new username and password without restarting the Data Aggregator. The Data Aggregator recognizes the new credentials immediately.

Access the OpenAPI

Access to the OpenAPI is restricted until the system administrator sets a username and password. Once you log in with the provided username and password, you do not need to provide the credentials again in the same session. The session does not time out until you clear browser cookies or restart your computer.

The base URL for the OpenAPI Query Builder is the following:
`http://`*`hostname`*`:`*`port`*`/odataquery`

hostname*:*port

Specifies the Data Aggregator host name and the port number.

Default port: 8581

About OpenAPI Query

The OpenAPI is used to export configuration and polled data for review. The OpenAPI lets you run queries that create custom Query URLs, which you can visit at any time to view your customized data. The OpenAPI stays constant when the internal data storage structure changes.

Unlike the data-driven and generic web services, the OpenAPI is a public API, and is meant for customer consumption. The OpenAPI uses the OData 2.0 industry standard. The documentation for this standard is found at the OData 2.0 Standard http://www.odata.org/documentation/odata-version-2-0/ web page.

Note: Refer to the metadata to review the schema XML description. This description provides detailed information about the OpenAPI. To view the metadata navigate to the following URL:

```
http://<yourDA>:8581/odata/api/$metadata
```

If you add new metric families to the Data Aggregator, restart the Data Aggregator to update the OData schema in the OpenAPI. To restart the Data Aggregator, run the following command:

```
/etc/init.d/dadaemon restart
```

Important: This release of the QueryBuilder interface is in BETA, and we do *NOT* recommend using the interface in production environments. There are limitations in scalability and authorization options (single user and password).

OpenAPI Controls

The OpenAPI Query Builder uses several types of controls. Use these controls to create and export Query URLs:

Token

Tokens are logical elements in the OpenAPI query syntax, which compose a complete query. Tokens appear when you click the Specify the Query field. A token lets you select the type of entity, column, metric family, metrics, or time range to include in the query. When each token is selected, the Query URL is updated accordingly.

Examples:

Column, filter, time range, metric family, component

Filter Control

Lets you create custom filters that are based on logical expressions. The Filter Control lets you create expressions using the AND and OR operations. You can manage single rules or groups of rules, and can select the operation (AND or OR) to apply to each.

Example:

Column filter, metrics filter

Format Control

Lets you set the formatting style of the data, including the columns that appear and the format of the output. When each format control is selected, the Query URL is updated accordingly. Select Optionally Specify the Formatting.

Examples:

Sorting, result limits, results format

The following two types of data tables are available:

HTML table with Extra Metrics

Allows you to view the aggregated minimum, maximum, and average metrics. Click the magnifying glass icon to view detailed metric data for the corresponding item.

HTML Table with Export

The metric columns appear empty, but you can export configuration information.

Button

Lets you execute an action in the UI, such as to copy the Query URL to the clipboard.

Examples:

Copy (📋), Add rule, Add group, Reset

Predefined Query Filters (Optional)

Lets you use out-of-the-box selections as templates that you can customize to create an OpenAPI query. These templates provide a starting point for building your own queries. Select a predefined query filter, and then select more tokens and filter controls to create a custom query. After you add a template, modify the filter controls to suit your needs.

Examples:

Routers within a specific group

Create an OpenAPI Query

The OpenAPI Query Builder lets you create a Query URL. For every parameter in the query, the OpenAPI adds the corresponding parameter to the Query URL. The parameter availability depends on the component, device, router, or other item that you want to query.

Follow these steps:

1. Click the Specify the Query field to start a query.

   A list of tokens appears.

2. Select a token or recipe.

   The token or recipe appears in the Specify the Query field.

   The address bar of the web browser updates to show the changes that you make in the Specify the Query field. Copy and save this URL for your records if you or another user wants to continue editing the query later.

3. Complete the query using the available tokens.

4. Click Format and Payload Options to customize the formatting of the data.

   Select the sorting method, the number of items to return, and the format of the data.

5. (Optional) Click the Run button.

   The OpenAPI runs the query and the results appear in the format that you selected.

6. Click the Copy (📋) button.

   The Query URL is copied to the clipboard, and can be pasted into a web browser or REST tool.

   The base URL for all OpenAPI queries is the following:
   `http://hostname:port/odata/api/`

Important: Queries that return large sets of results can negatively impact your system. We recommend refining your queries to return only the results that are relevant to your needs.

OpenAPI Query Troubleshooting

Metric Values Do Not Appear in Table

Symptom:

I ran a query to generate a table with metric values. However, when I open the table, no metric values appear in the results.

Solution:

The OpenAPI does not currently support the aggregation of multiple data samples into a single value for a particular time range.

To preview the results of your query, select the appropriate output format, such as:

- HTML table with extra metrics
- JSON
- XML

Query Results in Empty Table

Symptom:

After you run a query, the resulting Query URL shows an empty table.

Solution:

When selecting to show metric columns in a table format, select at least one configuration column, such as item name. Another solution is to select a format other than 'HTML table with export'.

Chapter 3: About Generic REST Web Services

This section contains the following topics:

Access Generic REST Web Services

Use generic REST web services for managing metric families and limited SNMP vendor certification support. Generic REST web services are self-filtering and do not use an argument within the URL to manage relationships.

The base URL for generic web services is:

```
http://hostname:port/genericWS
```

hostname*:*port

Specifies the Data Aggregator host name and the port number.

Default port: 8581

Access the following URLs in a browser to get detailed information about generic web services, including XSDs, URIs, supported HTTP methods, attributes, and relationships.

- To displays details about every user-facing, generic web service in the system:

  ```
  http://hostname:port/genericWS/
  ```

- To display details about the specified endpoint:

  ```
  http://hostname:port/genericWS/endpoint/documentation
  ```

Manage Relationships With Generic REST Web Services

Generic REST web services do not use an argument within the URL to manage relationships. Instead, generic REST web services rely on the basic operations alone to manage relationships. The endpoints filter on themselves to expose the information. These methods are used for managing relationships between the metric families and SNMP vendor certifications.

Relationships for generic REST web services are viewed, created, and deleted using the following methods:

```
GET http://hostname:port/genericWS/endpoint/name/endpoint
```

```
PUT http://hostname:port/genericWS/endpoint/name/endpoint
```

```
DELETE http://hostname:port/genericWS/endpoint/name/endpoint
```

Parameters

hostname*:*port

Specifies the Data Aggregator host name and the port number.

Default port: 8581

endpoint*/*name

Specifies the name of a metric family or SNMP vendor certification endpoint.

Example: List Metric Families Related to an SNMP Vendor Certification

To return a list of metric families that are related to a specified SNMP vendor certification, use the following URL syntax:

```
GET http://hostname:port/genericWS/certifications/snmp/name/metricfamilies
```

Chapter 4: About Data-Driven REST Web Services

This section contains the following topics:

Access Data-Driven REST Web Services

As an administrator, use data-driven REST web services for most Data Aggregator web services, such as monitoring profiles and groups. You can also scope to tenant domains (to a limited extent).

The base web service URL for data-driven web services is:

`http://`*`hostname`*`:`*`port`*`/rest`

hostname*:*port

Specifies the Data Aggregator host name and the port number.

Default port: 8581

Access the following URLs in a browser to get detailed information about endpoints, and data-driven web services, including XSDs, URIs, supported HTTP methods, attributes, and relationships.

- To display details about every user-facing, data-driven web service in the system:

  `http://`*`hostname:port`*`/rest/`

- Displays details about the specified endpoint:

  `http://`*`hostname:port`*`/rest/`*`endpoint`*`/documentation`

View XSD Schemas for Data-Driven REST Web Services

Do the following steps before performing an HTTP request:

- Verify the XML schema definition (XSD) for the endpoint.
- Review the format of the return or upload XML that the service provides.

Each item of content that is placed in an XML document must adhere to the description of the endpoint.

To obtain the XSD for an endpoint, use the following paths with the data-driven web services URL:

`http://`*`hostname`*`:`*`port`*`/rest/`*`endpoint`*`/XSD/`*`operation`*`.xsd`

operation

Specifies the type of operation to execute.

The following operations are supported overall, but some are not supported for each endpoint:

get

Obtains the XSD for a single item get.

getlist

Obtains the XSD for a list of the endpoint items.

filterselect

Allows advanced filter criteria and return XML format to be specified using GET Tunneling.

create

Obtains the XSD that any input XML must match when trying to create.

update

Obtains the XSD that any input XML must match when trying to update.

Note: If an operation is not supported, the web service fails and returns a '403 Forbidden' message.

The automatically-generated XSD files for create, update, get, getlist, and filterselect contain tags that describe the attributes and the purpose of the metric families.

Filter on Attributes in the XSD Schema for Data-Driven REST Web Services

You can filter on attributes such as the item name, description and other such attributes. For example, filter monitoring profiles by the metric families they contain. You can use this information to determine whether to add or remove metric families to or from a monitoring profile.

Follow these steps:

1. Enter the following URL in a web browser:

   `http://`*`hostname:port`*`/rest/`

   A list displays the available data-driven web services.

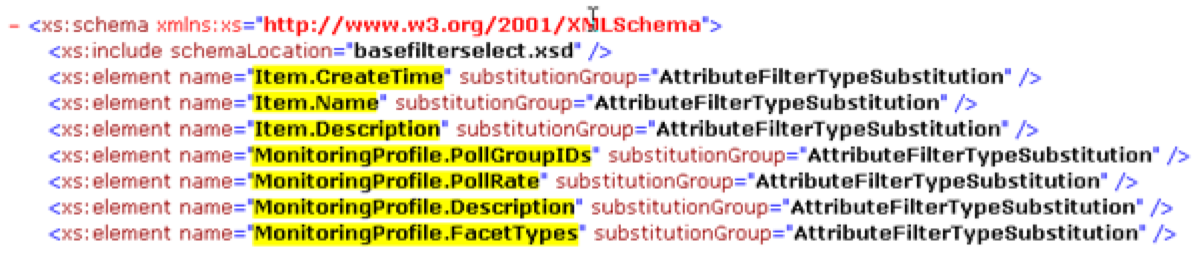
2. Click a web service.

   The documentation page for that web service displays.

3. Click the URL under the "filtered get list" method.

   The XSD schema opens.

4. Look for the elements that have substitutionGroup="AttributeFilterTypeSubstitution", as shown in the following example. Use this information to determine which attribute you want to filter on.



```
- <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:include schemaLocation="basefilterselect.xsd" />
    <xs:element name="Item.CreateTime" substitutionGroup="AttributeFilterTypeSubstitution" />
    <xs:element name="Item.Name" substitutionGroup="AttributeFilterTypeSubstitution" />
    <xs:element name="Item.Description" substitutionGroup="AttributeFilterTypeSubstitution" />
    <xs:element name="MonitoringProfile.PollGroupIDs" substitutionGroup="AttributeFilterTypeSubstitution" />
    <xs:element name="MonitoringProfile.PollRate" substitutionGroup="AttributeFilterTypeSubstitution" />
    <xs:element name="MonitoringProfile.Description" substitutionGroup="AttributeFilterTypeSubstitution" />
    <xs:element name="MonitoringProfile.FacetTypes" substitutionGroup="AttributeFilterTypeSubstitution" />
```

5. Open a REST client editor or HTTP tool that sends requests and gets responses, and set the Content-type to **application/xml**.

6. Enter the following filter criteria:

   ■ URL: http://*hostname*:*port*/rest/*endpoint*/filtered/

   ■ HTTP method = POST

     This method must define the filter criteria.

- Basic filter select criteria on the Body tab in the HTTP Request pane:

  ```
  <FilterSelect xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="filter.xsd">
         <Filter>
                     <elementName
  type="CONTAINS">filter-criteria</elementName>
         </Filter>
  </FilterSelect>
  ```

  filter-criteria

  Specifies the actual value of the attribute.

  elementName

  Specifies the element name (attribute) to filter on.

  Note: You can specify selection criteria also, such as poll rates only. This method is also known as Get Tunneling. For more information, see the following example.

  Results are returned in the Body tab of the HTTP Response pane.

Example: Return a List of Monitoring Profiles that Contain a Metric Family Using Filter and Selection Criteria (Get Tunneling).

Enter the following URL to return the monitoring profiles that contain a metric family using poll rate as the selection criteria:

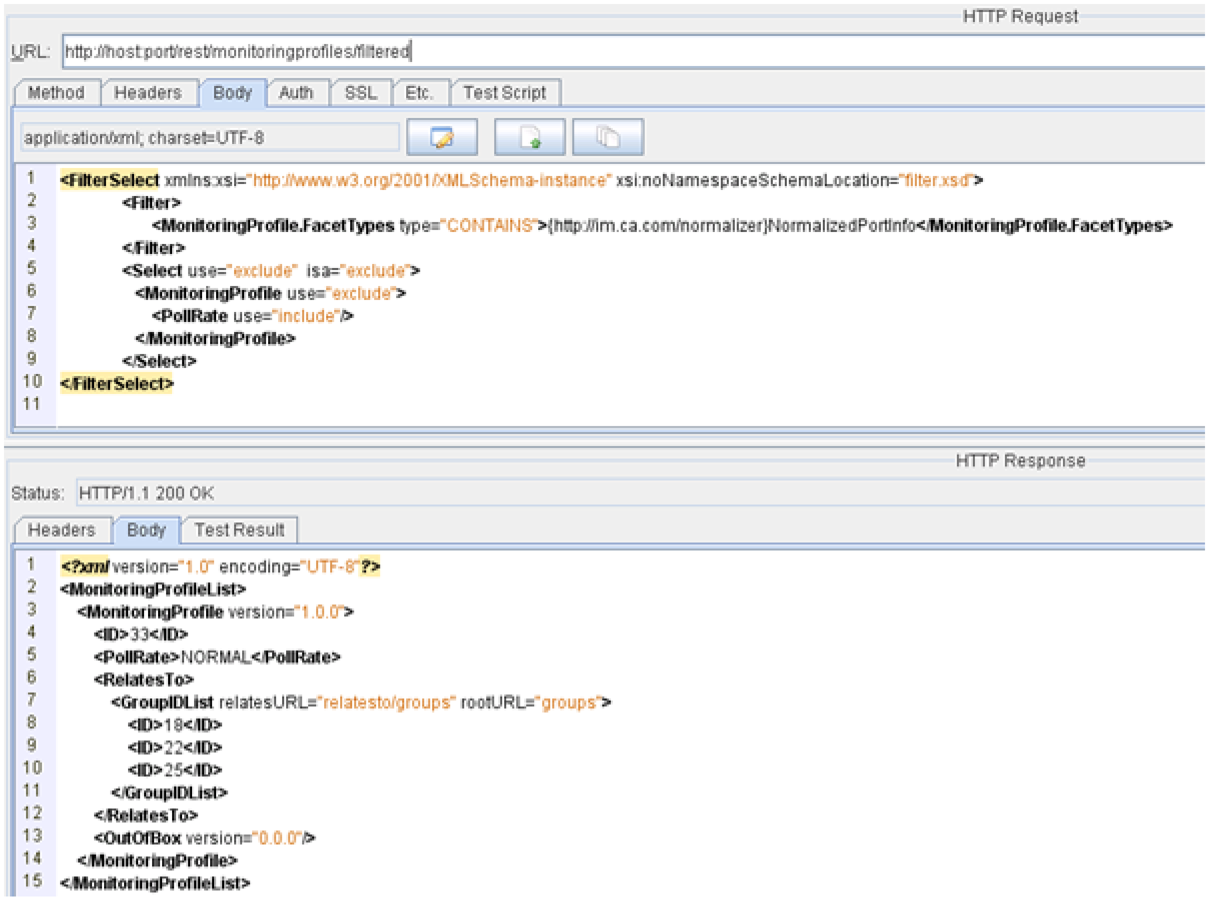
- Method and URL:

  ```
  POST http://hostname:port/rest/monitoringprofiles/filtered/
  ```

- Body:

  ```
  <FilterSelect xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="filter.xsd">
                 <Filter>
                        <MonitoringProfile.FacetTypes
  type="CONTAINS">{http://im.ca.com/normalizer}NormalizedPortInfo</MonitoringPr
  ofile.FacetTypes>
                 </Filter>
                 <Select use="exclude"  isa="exclude">
                    <MonitoringProfile use="exclude">
                        <PollRate use="include"/>
                    </MonitoringProfile>
                 </Select>
  </FilterSelect>
  ```

The following graphic reflects the preceding example. The response shows the monitoring profiles that contain the NormalizedPortInfo metric (filter criteria), and these profiles only contain the specified PollRate attribute (selection criteria).


HTTP Request

URL: http://host:port/rest/monitoringprofiles/filtered

Body: application/xml; charset=UTF-8

```
<FilterSelect xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="filter.xsd">
          <Filter>
               <MonitoringProfile.FacetTypes type="CONTAINS">{http://im.ca.com/normalizer}NormalizedPortInfo</MonitoringProfile.FacetTypes>
          </Filter>
          <Select use="exclude"  isa="exclude">
            <MonitoringProfile use="exclude">
               <PollRate use="include"/>
            </MonitoringProfile>
          </Select>
</FilterSelect>
```

HTTP Response

Status: HTTP/1.1 200 OK

```
<?xml version="1.0" encoding="UTF-8"?>
<MonitoringProfileList>
  <MonitoringProfile version="1.0.0">
    <ID>33</ID>
    <PollRate>NORMAL</PollRate>
    <RelatesTo>
      <GroupIDList relatesURL="relatesto/groups" rootURL="groups">
        <ID>18</ID>
        <ID>22</ID>
        <ID>25</ID>
      </GroupIDList>
    </RelatesTo>
    <OutOfBox version="0.0.0"/>
  </MonitoringProfile>
</MonitoringProfileList>
```

Manage Basic Relationships With Data-Driven REST Web Services

Relationships are created and deleted using the following methods:

PUT http://*hostname*:*port*/rest/*endpoint*/*id*/relatesto/*endpoint*/*id*
DELETE http://*hostname*:*port*/rest/*endpoint*/*id*/relatesto/*endpoint*/*id*

You can also view nested relationships using the following method:

http://*hostname*:*port*/rest/*endpoint*/*id*/relatesto/*endpoint*/*endpoint*

Example

Use the following URL to see all groups and devices that are associated with a specific monitoring profile ID 781:

```
GET http://hostname:port/rest/monitoringprofile/781/relatesto/groups/devices
```

The response URL lists all the groups and devices that are related to the specified monitoring profile ID with the attributed listed in the current getlist.xsd file.

Chapter 5: Using Data-Driven Web Services

This section contains the following topics:

Limit Data-Driven Scope to Tenant Domains

You can limit the data-driven scope of your operations to a specific tenant domain for some of the Data Aggregator features, rather than accessing information in the entire global repository.

Use the following basic method to access information for an endpoint with a specific tenant domain:

```
GET http://hostname:port/rest/tenant/id/endpoint
```

You can use some, but not all, web services with tenant domains.

Poll Sensitive and Critical Devices Without a Performance Impact

As the Administrator, you know which of your critical devices are sensitive to too many polls, which can lead to performance problems. However, to ensure the performance of your critical devices, you must somehow monitor these sensitive devices. By configuring the SNMP polling controls, you can throttle the SNMP poll requests and avoid overwhelming your sensitive devices.

By default, SNMP polling is controlled in two ways:

- SNMP traffic threshold — No more than 15 SNMP requests can be sent to a device at a time. Poll and discovery SNMP requests over 15 are queued and sent to a device when possible during the polling cycle. Up to 600 requests can be queued.
- SNMP timeouts threshold — When 15 or more SNMP requests timeout, polling is suspended for the remainder of the current polling cycle. An event is generated, informing you of the situation.

  Note: Polling resumes at the beginning of each poll cycle. When the timeouts do not exceed the 15 threshold, a "clear" event is generated.

Both thresholds are designed to prevent overwhelming a device with poll requests. However, you can configure these SNMP polling thresholds, when needed.

For example, your older router is exceptionally sensitive to polling. But, this router is critical and must be polled as frequently as possible. You already adjusted your monitoring profile to remove unnecessary metric families from polling. You also applied a filter in your monitoring profile to reduce the number of polled interfaces. However, polling still causes this router to crash. Therefore, your only option is to adjust the default SNMP polling parameters for your sensitive router.

Certain parameters, such as the following, may be added to the policy for individual IPs or IP ranges in an IPRange section within the IPRangeList:

MaxOutstandingRequests

The maximum number of outstanding requests sent to the devices within the indicated IP range.

MaxRequestSize

Limits the number of OIDs in an outgoing SNMP request. If the number of OIDs in the SNMP request exceeds the value of *MaxRequestSize*, the outgoing request is split into two or more smaller requests.

Some IP ranges are not covered in the IPRange sections. For global settings, use the *MaxRequestSizeDefault* parameter to set the OID limit.

Follow these steps:

1. Find the ID for your IP Domain (that contains your sensitive router) by opening:

   `http://`*`hostname:port`*`/rest/ipdomains`

   hostname*:*port

   Specifies the Data Aggregator hostname and the port number where you are accessing the REST web services from.

2. Locate your IP Domain ID in the following SNMP throttle policy list, and note the corresponding policy ID:

   `http://`*`hostname:port`*`/rest/snmpthrottlepolicies`

3. Determine the number of OIDs that you want to include in a single outgoing SNMP request. Some devices ignore requests that are too large without sending an error. As a result, the SNMP poller cannot reach the device. Use the *MaxRequestSize* value to allow the Data Collector to monitor these devices.

   Example

   If the interface SNMP request has 27 OIDs and *MaxRequestSizeDefault* is set to 15, the outgoing request is split into two smaller requests. One request contains 14 OIDs, and the other contains 13 OIDs.

   Example: The following example from an SNMP throttle policy shows that the policy ID is "601" for IP Domain "2" with no limit on the number of OIDs:

```
<SnmpThrottlePolicy version="1.0.0">
    <ID>601</ID>
    <MaxOutstandingRequestsDefault>15</MaxOutstandingRequestsDefault>
    <QueueLength>600</QueueLength>
    <TimeoutFailSafeThrottleDefault>15</TimeoutFailSafeThrottleDefault>
    <MaxRequestSizeDefault>0</MaxRequestSizeDefault>
     <IPDomainID>2</IPDomainID>
    </SnmpThrottlePolicy>
```

4. Open a REST client editor or HTTP tool that sends requests and gets responses, and set the Content-type to application/xml.

5. Open and edit the SNMP throttle policy for your IP Domain by entering the following criteria:

   ■ URL: http://*hostname*:*port*/rest/snmpthrottlepolicies/*policyID*

     policyID

     Specifies a unique identification number that is assigned to the SNMP throttle policy for the IP Domain that contains your sensitive device.

     Example: http://*hostname:port*/rest/snmpthrottlepolicies/601

   ■ HTTP method = PUT

- Adjust the following values for your IP Range on the Body tab in the HTTP Request pane:
  - <MaxOutstandingRequests> – SNMP traffic threshold
  - <TimeoutFailSafeThrottle> – SNMP timeouts threshold

  Note: Both values are required for every IP Range entry. You can disable either parameter by setting the value to "0."
- Remove the following lines:
  - <ID>
  - <IPDomainID>

Results are returned in the Body tab of the HTTP Response pane.

Example: In this example, the thresholds are lowered to "10" for device 10.231.41.7 only. For this device, the number of OIDs is limited to 50. The default thresholds and other IP Range thresholds continue using the default value of "15." For devices 10.231.41.1-10.231.41.255, SNMP requests are limited to 30 OIDs.

```
<SnmpThrottlePolicy version="1.0.0">
    <IPRangeList>
      <IPRange>
            <IPRangeText>10.231.41.7</IPRangeText>
            <MaxOutstandingRequests>10</MaxOutstandingRequests>
            <TimeoutFailSafeThrottle>10</TimeoutFailSafeThrottle>
                    <MaxRequestSize>50</MaxRequestSize>
      </IPRange>
      <IPRange>
            <IPRangeText>10.231.41.1-10.231.41.255</IPRangeText>
            <MaxOutstandingRequests>15</MaxOutstandingRequests>
            <TimeoutFailSafeThrottle>15</TimeoutFailSafeThrottle>
                    <MaxRequestSize>30</MaxRequestSize>
      </IPRange>
    </IPRangeList>
        <MaxOutstandingRequestsDefault>15</MaxOutstandingRequestsDefault>
        <QueueLength>600</QueueLength>
        <TimeoutFailSafeThrottleDefault>15</TimeoutFailSafeThrottleDefault>
</SnmpThrottlePolicy>
```

Note: You can adjust the thresholds for a single device or a range of devices. The IP Range definition and the IP Range order determine which threshold applies. The IP Ranges are listed in priority order. That is, the first IP Range that applies to a device determines the threshold value to apply.

6. Always include the *MaxOutstandingRequestsDefault, TimeoutFailSafeThrottleDefault,* and *QueueLength* parameters in the update/POST XML at the root level. Include the parameters even if the values do not differ from the default.

   Example:

   This PUT command generates the policy that follows.

```
Update XML: PUT on URL DA-HOST:8581/rest/snmpthrottlepolicies/21
   <SnmpThrottlePolicy version="1.0.0">
   <IPRangeList>

     <IPRange>

       <IPRangeText>130.119.103.8</IPRangeText>
       <MaxOutstandingRequests>10</MaxOutstandingRequests>
       <TimeoutFailSafeThrottle>10</TimeoutFailSafeThrottle>
          <MaxRequestSize>20</MaxRequestSize>
     </IPRange>
   </IPRangeList>
   <MaxRequestSizeDefault>50</MaxRequestSizeDefault>
   <MaxOutstandingRequestsDefault>15</MaxOutstandingRequestsDefault>
   <TimeoutFailSafeThrottleDefault>15</TimeoutFailSafeThrottleDefault>
   <QueueLength>600</QueueLength>
   </SnmpThrottlePolicy>
```

   This command generates the following policy:

```
<SnmpThrottlePolicy version="1.0.0">
<ID>21</ID>
<QueueLength>600</QueueLength>
<TimeoutFailSafeThrottleDefault>15</TimeoutFailSafeThrottleDefault>
<IPDomainID>2</IPDomainID>
<IPRangeList>
<IPRange>
<IPRangeText>130.119.103.8</IPRangeText>
<MaxOutstandingRequests>10</MaxOutstandingRequests>
<TimeoutFailSafeThrottle>10</TimeoutFailSafeThrottle>
<MaxRequestSize>20</MaxRequestSize>
</IPRange>
</IPRangeList>
<MaxRequestSize>50</MaxRequestSize>
<MaxOutstandingRequestsDefault>15</MaxOutstandingRequestsDefault>
</SnmpThrottlePolicy>
```

Change Data Retention Periods

You can change the rate at which Data Repository retains the polled data, hourly rollup data, daily rollup data, and weekly rollup data. For example, you can change the polled data retention value to 30 days to conserve disk space. Find the balance that best suits your needs and environment.

By default, data is retained in Data Repository for the following number of days:

- Polled data: 45 days

  Note: If you upgraded to this release from a previous release of Data Aggregator, the polled data retention will not change from the prior default of ten days.

- Hourly rollup data: 90 days
- Daily rollup data: 365 days
- Weekly rollup data: 730 days

The minimum number of days that Data Repository can retain data for is as follows:

- Polled data: two days
- Hourly rollup data: eight days
- Daily rollup data: 31 days
- Weekly rollup data: 366 days

Follow these steps:

1. Enter the following information in a web browser:
2. http://*hostname:port*/rest/globalretentiondefinition

   hostname*:*port

   Specifies the Data Aggregator host name and the port number.

   Default port: 8581

   The globalretentiondefinition webservice endpoint URL opens.

3. Take note of the ID that is assigned to the globalretentiondefinition.
4. Look for the elements that have GtdRollupDataRetentionPeriod, DailyRollupDataRetentionPeriod, PolledDataRetentionPeriod, and HourlyRollupDataRetentionPeriod. This information helps you determine which types of data you want to modify the retention period for.
5. Open a REST client editor or HTTP tool that sends requests and gets responses and set the Content-type to application/xml.

6. Enter the following criteria:

   - URL: http://*hostname*:*port*/rest/globalretentiondefinition/*ID*

     ID

       Is a unique identification number that is assigned to the globalretentiondefinition.

   - HTTP method = PUT

   - Enter the retention periods that you want to change in the Body tab of the HTTP Request pane.

     For example:

     ```
     <GlobalRetentionDefinition version="1.0.0">
     <PolledDataRetentionPeriod>4</PolledDataRetentionPeriod>
     </GlobalRetentionDefinition>
     ```

     Important! Verify that there is no white space at the beginning of each of these lines, otherwise the PUT fails.

     In this example, the polled data retention period has been changed to four days.

   Results are returned in the Body tab of the HTTP Response pane.

   For example:

   ```
   <GlobalRetentionDefinitionList>
   <GlobalRetentionDefinition version="1.0.0">
     <ID>4</ID>
     <GtdRollupDataRetentionPeriod>730</GtdRollupDataRetentionPeriod>
     <DailyRollupDataRetentionPeriod>365</DailyRollupDataRetentionPeriod>
     <PolledDataRetentionPeriod>4</PolledDataRetentionPeriod>
     <HourlyRollupDataRetentionPeriod>90</HourlyRollupDataRetentionPeriod>
   <Item version="1.0.0">
     <CreateTime>Thu Dec 08 16:03:05 CST 2011</CreateTime>
     <Name>Global Retention Definition</Name>
     </Item>
     </GlobalRetentionDefinition>
     </GlobalRetentionDefinitionList>
   ```

   In this example, the polled data retention period has been changed to four days. The default retention periods for weekly rollup data, daily rollup data, and hourly rollup data remain.

Schedule Data Purges

You schedule how often Data Repository purges all data that is older than the specified retention periods. You can modify the start hour, start minute, and you can modify the start second. By default, the Data Aggregator purges data every day at 2:00:00 AM.

Follow these steps:

1. Enter the following information in a web browser:

   `http://`*`hostname:port`*`/rest/globalretentionscheduledefinition`

   hostname*:*port

   Specifies the Data Aggregator hostname and the port number where you are accessing the REST web services from.

   The globalretentionscheduledefinition webservice endpoint URL.

2. Take note of the ID that is assigned to the globalretentionscheduledefinition.
3. Look for the elements that have StartMinute, StartHour, and StartSecond. Use this information to determine whether you want to modify the start hour, start minute, or start second when old data is purged.
4. Open a REST client editor or HTTP tool that sends requests and gets responses and set the Content-type to application/xml.
5. Enter the following criteria:
   - URL: http://*hostname*:*port*/rest/globalretentionscheduledefinition/*ID*

     ID

     Is a unique identification number that is assigned to the globalretentionscheduledefinition.
   - HTTP method = PUT
   - Enter the time values that you want to change in the Body tab of the HTTP Request pane.

For example:

```
<GlobalRetentionScheduleDefinition version="1.0.0">
  <StartMinute>28</StartMinute>
  <StartHour>17</StartHour>
  <Enabled>true</Enabled>
  <Status>Scheduled to run everyday at 17:28:00</Status>
 </GlobalRetentionScheduleDefinition>
```

Important! Be sure that there is no white space at the beginning of each of these lines, otherwise the PUT operation fails.

In this example, the start hour has been changed to 17 and the start minute has been changed to 28.

Note: To disable the purge job, set <Enabled> to false. To reenable the purge job, set <Enabled> to true.

Results are returned in the Body tab of the HTTP Response pane.

For example:

```
<GlobalRetentionScheduleDefinitionList>
<GlobalRetentionScheduleDefinition version="1.0.0">
  <ID>9</ID>
  <StartMinute>28</StartMinute>
  <StartHour>17</StartHour>
  <Enabled>true</Enabled>
  <JobStatus>Has never run</JobStatus>
  <Status>Scheduled to run everyday at 17:28:00</Status>
  <StartSecond>0</StartSecond>
<Item version="1.0.0">
  <CreateTime>Thu Dec 15 15:52:20 EST 2011</CreateTime>
  <Name>Global Retention Schedule Definition</Name>
  </Item>
  </GlobalRetentionScheduleDefinition>
  </GlobalRetentionScheduleDefinitionList>
```

In this example, data that is older than the specified retention periods will be purged every day at 17:28:00.

Change When Same Day, Same Hour Baseline Averages Are Calculated

Initially, when a limited amount of data has been collected, the baseline average is calculated for the same hour for every preceding day of the week.

When more data is available, a switchover in the calculation method occurs automatically and Data Aggregator establishes "normal" by averaging hourly samples across available preceding same days of the week.

By default, this automatic switchover occurs when at least 3 same day of the week, same hour data samples are available for the past 12 weeks. You can change when this automatic switchover occurs.

Consider the following information about changing when same day of the week, same hour baseline averages are calculated:

- You have the option of changing both attributes, or just one of the attributes.
- Both attribute values must be a numeric value that is greater than or equal to 1.
- No upper limit is enforced. However, the retention policy of the hourly roll ups defines the upper limit. By default, the hourly retention rate is 90 days (which is approximately 12 weeks of data). If you increase the maximum number of preceding weeks, increase the hourly roll-up retention rate also.
- The MinimumNumberOfRequiredDataPoints attribute value must be less than or equal to the MaximumNumberOfWeeks value.

Follow these steps:

1. Enter the following information in a web browser:

   http://*hostname:port*/rest/sdshbaselineconfig

   hostname*:*port

   Specifies the Data Aggregator host name and the port number.

   Default port: 8581

   The sdshbaselineconfig webservice endpoint URL opens.

2. Review the current values for the minimum required number of data points and the maximum number of preceding weeks.

3. Open a REST client editor or HTTP tool that sends requests and gets responses and set the Content-type to application/xml.

4. Enter the following criteria:
   - HTTP method = PUT
   - Enter the minimum required number of data points within the maximum number of preceding weeks (to trigger the switchover in the baseline calculation method) that you want to change on the Body tab in the HTTP Request pane.

     For example:

     ```
     <SdshBaselineConfiguration version="1.0.0">
      <SDSHSettings>

     <MinimumNumberOfRequiredDataPoints>5</MinimumNumberOfRequiredDataPoints>
       <MaximumNumberOfWeeks>10</MaximumNumberOfWeeks>
       </SDSHSettings>
     </SdshBaselineConfiguration>
     ```

   In this example, the minimum number of data points to be available for baseline average calculation has been changed to 5. The number of preceding weeks to look for these data points has been changed to 10.

Modify When Rollup Processing and Baseline Calculations Are Performed

Administrators can modify when rollup processing and baseline calculations are performed. Changing the time when these operations run allows administrators to schedule these Vertica-intensive operations to occur during off-hours. When these operations occur during off-hours, users who generate reports will not be impacted during business hours.

By default, rollup processing and baseline calculations are run at the bottom of the hour, every hour of the day.

Follow these steps:

1. Enter the following URL in a web browser:

   http://*hostname*:*port*/rest/rollups/config

   hostname*:*port

   Specifies the Data Aggregator host name and the port number.

   Default port: **8581**

2. Take note of the ID that is assigned to the configuration item.

3. Open a REST client editor or HTTP tool that sends requests and gets responses. Enter the following criteria:

   ■ URL: http://*hostname*:*port*/rest/rollups/config/*ID*

     hostname*:*port

       Specifies the Data Aggregator host name and the port number.

       Default port: 8581

     ID

       Is a unique identification number that is assigned to the configuration item. You noted this number in the previous step.

   ■ HTTP method: PUT

   ■ Enter the hour of the day when you want rollup processing to begin and end in the Body tab of the HTTP Request pane.

   By default, the following results are returned:

```
<RollupsConfigurationList>
    <RollupsConfiguration version="1.0.0">
           <ID>8</ID>
           <StartHour>0</StartHour>
           <EndHour>23</EndHour>
    </RollupsConfiguration>

</RollupsConfigurationList>
```

   StartHour

     Defines the hour of the day (in your local timezone) in 24-hour time format when rollup processing will begin.

   EndHour

     Defines the hour of the day (in your local timezone) in 24-hour time format when rollup processing should end. No new rollups will be kicked off after the end-hour, but any rollups that are in-progress will be allowed to complete.

   Note: For more details on these attributes, see http://*hostname*:*port*/rest/rollups/config/documentation.

Example: In this example, you change the schedule so that rollup processing and baseline calculations run only from 20:00 to 7:00.

```
<RollupsConfigurationList>
    <RollupsConfiguration version="1.0.0">
        <ID>8</ID>
        <StartHour>20</StartHour>
        <EndHour>6</EndHour>
    </RollupsConfiguration>
</RollupsConfigurationList>
```

The <EndHour> is inclusive. In this example, that means if you specify 6 as the EndHour, rollup processing and baseline calculations will be initated in at the bottom of the hour during the 06:00 hour, but they will not be initiated at the 07:00 hour. Any calculations that are in progress will be allowed to complete.

Important! Any modifications to the default schedule can result in a larger delay in data showing up in reports pertaining to the corresponding resolution. For example, if hourly rollups are delayed, then a reporting showing hourly resolution data will not be current until the hourly rollup has been performed.

Change the Primary IP Address On a Monitored Device

As an administrator, you want your monitored devices to maintain consistent data, even if the IP address on a device changes. If the IP address on the device changes and you do not update it, a subsequent discovery can create a new monitored device.

Consider the following information before you change the primary IP address:

- If a device has a DNS hostname, when the IP address changes and the old IP address is no longer reachable, the Data Collector instance that monitors the device performs reverse hostname lookup to find and set the device item with the new IP address.
- If you change the primary IP address of a device to an IP address that another devices uses, an error message displays.

Note: If the primary IP address on a monitored device changes, an event is generated on the device.

To change the primary IP address on a monitored device, open a REST client editor or HTTP tool and enter the following criteria:

- URL: http://*hostname*:*port*/rest/devices/*deviceitemID*

  hostname*:*port

  Specifies the Data Aggregator host name and the port number.

  Default port: 8581

  deviceitemID

  Is the device item identification number for the monitored device for which you are changing the primary IP address.

  Note: An error is returned if you change the IP address to the IP address of an existing monitored device.

- HTTP method: PUT
- Enter the changed primary IP address in the Body tab of the HTTP Request pane.

For example:

```
<Device version="1.0.0">
      <PrimaryIPAddress>IP</PrimaryIPAddress>
</Device>
```

IP

Is the changed primary IP address.

Example: In this example, you change the primary IP address on a monitored device to 1.2.3.4:

```
<Device version="1.0.0">

      <PrimaryIPAddress>1.2.3.4</PrimaryIPAddress>

</Device>
```

How to Create a Custom Metric Family

Factory metric families define the most common metric attributes to monitor. Several of these are included upon installation, and these metric families satisfy the needs of most users. 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.

Your custom metric family defines the following details:

- Which metrics to gather
- How to calculate the values
- (Optional) How to display the values in your views

Optionally, you can then create a vendor certification, if one does not already exist, to monitor the metric attributes for process, using your new metric family.

Important! Always create and verify your custom metric family in a *test environment first*. Creating a custom metric family requires you to edit the metric family XML file manually. Semantic errors in this XML file can cause unpredictable results.

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 basic certification method. For information about the advanced certification method, see the *Data Aggregator Power User Certification Guide*.

Creating a custom metric family is a complex process. Follow these steps carefully to create your custom metric family:

1. View existing metric families to determine if you require a custom metric family (see page 37).
2. Download the schema and sample metric family files (see page 38).
3. Create a custom metric family XML file (see page 38).
4. Verify your custom metric family results in a test environment (see page 66).
5. Import the custom metric family XML file (see page 67).

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.

View Metric Families

Review the list of metric families to see which metrics are supported in your Data Aggregator installation. Viewing a metric family shows its associations with device collections, vendor certifications, and monitoring profiles. Understanding the relationships between metric families, device collections, and device types helps you control how to monitor your devices. Also, you can determine whether you need additional metric families to monitor your environment sufficiently.

Follow these steps:

1. Click Metric Families from the Monitoring Configuration menu for a Data Aggregator data source.

   A list of metric families displays, including factory and custom metric families. Predefined certifications display a lock symbol.

2. Select a metric family from the list.
3. Click a tab to get more information:

   Metrics tab

   Shows the metrics that are included in the selected metric family and various properties for each metric.

   Vendor Certification Priorities tab

   Displays a list of device collections that are associated with the selected metric family. Typically, a metric family is associated with a single device collection. When you select a device collection, a prioritized list of MIB sources (vendor certifications) appears. This information tells you the priority order in which the vendor certifications are applied to that device collection for the selected metric family.

   Monitoring Profiles tab

   Displays a list of associated monitoring profiles and their poll rates.

More Information:

How to Create a Custom Metric Family (see page 36)

Download the Metric Family Schema and Example Files

Before you create a custom metric family XML file, download and review the metric family schema and example metric family XML files. You need the schema to create your own XML file. The example demonstrates how to code a successful metric family. Reviewing these files before creating your own helps ensure the accuracy of your XML content.

Locate and download the following files into the same folder:

- http://*hostname*:*port*/resource/xsd/MetricFamily.xsd
- http://*hostname*:*port*/resource/xsd/Component.xsd
- http://*hostname*:*port*/resource/examples/metricFamily/ProcessInfoMFWithComponent.xml

Note: For more information about each XML tag, see the inline documentation provided in the MetricFamily.xsd and Component.xsd files. For code examples, see the ProcessInfoMFWithComponent.xml file. This example file defines a metric family for gathering process metrics.

More information:

How to Create a Custom Metric Family (see page 36)

Create a Custom Metric Family XML File

After gathering the required files, you are ready to create your custom metric family XML file.

Note: For more information about each XML tag, see the inline documentation provided in the MetricFamily.xsd and Component.xsd files. For code examples, see the ProcessInfoMFWithComponent.xml file. This example file defines a metric family for gathering process metrics.

Follow these steps:

1. Create an XML file from the MetricFamily.xsd using any XML editor. Any custom metric family starts with <TechnologyCertification> as the root element.

We recommend that you create your XML file by copying the example metric family XML file or by exporting a factory XML file using web services (http://*hostname*:*port*/genericWS/metricfamilies).

Important! Do not copy a metric family XML file from the deploy directory. These files use an internal-only format that is not compatible with custom metric families.

Note: You cannot modify factory metric families. However, you can copy a factory metric family and then modify it.

2. Edit the content in your XML file, as follows:

   a. Review data type usage in metric family XML files (see page 40).

   b. Add the basic information that defines your metric family (see page 43).

   c. Add the <AttributeList> section (see page 45).

   d. Add the <BaselineList> section (see page 50).

   e. Add the <ComponentList> section (see page 52).

   f. Add the <ComponentDefinitionList> section (see page 52).

   g. (Optional) Add the <ComponentReconciliation> section (see page 58).

   h. (Optional) Add the <ReconfigDetectionAttr> section (see page 63).

   i. Add the <ExpressionGroupList> section (see page 64).

3. Save your file.

   Your custom metric family is created and ready to import into Data Aggregator.

   Important! Always create and verify your custom metric family in a *test environment first*. Creating a custom metric family requires you to edit the metric family XML file manually. Semantic errors in this XML file can cause unpredictable results.

More information:

How to Create a Custom Metric Family (see page 36)

Data Type Usage in Metric Family XML Files

This section provides guidance on data type usage in metric family and vendor certification XML files.

ObjectID

To access the polled OID rather than the return value of the poll, use the type *ObjectID*.

Integer/Long/Double

Numeric values are typically stored in the types *Integer* or *Long* in vendor certifications. Metric Families typically use the type *Double*.

BigInteger/Double

Data Aggregator supports the polling of 64-bit counters, which in turn supports the collection of data for high-speed interfaces. As the speed of network media increases, the minimum time in which a 32-bit counter wraps decreases. Using 64-bit counters lengthens the time that it takes for a counter to wrap and enables polling at a normal rate. The ifxTable in MIB2 provides 64-bit counters, which for vendor certifications are typically stored in the *BigInteger* type. Metric families typically use the type *Double*.

String/OctetString

String values are stored in the type *String* in vendor certifications. Metric families use the type *OctetString*.

QName

A special type used in metric families for the *FacetTypes* attribute.

Note: The schema files that are provided with CA Performance Management have detailed information about types. Schema-aware XML editors such as XML Notepad use this information and can assist you during creation of XML files.

Example of a part of the ProcessInfoVCForEmpireMIB.xml file with data types that define the MIB object attributes in this vendor certification

```
      <Attribute name="INDEX" type="ObjectID">
        <!--This variable serves as the index for the other variables in the same MIB
table.-->
        <IsKey>true</IsKey>
        <IsIndex>true</IsIndex>
        <Source>1.3.6.1.4.1.546.1.1.4.1.1</Source>
      </Attribute>
      <Attribute name="processID" type="Long">
        <!--The unique process ID (e.g. 0).-->
        <IsKey>true</IsKey>
        <NeedsDelta>false</NeedsDelta>
        <Source>1.3.6.1.4.1.546.1.1.4.1.1</Source>
      </Attribute>
      <Attribute name="processName" type="String">
        <!--The name of the running process (e.g. syslogd).-->
        <IsKey>true</IsKey>
        <NeedsDelta>false</NeedsDelta>
        <Source>1.3.6.1.4.1.546.1.1.4.1.2</Source>
      </Attribute>
      <Attribute name="processRSS" type="Long">
        <!--Real memory (resident set) size of the process in kilobytes. This value
indicates how many bytes are held by a process.-->
        <IsKey>false</IsKey>
        <NeedsDelta>false</NeedsDelta>
        <Source>1.3.6.1.4.1.546.1.1.4.1.11</Source>
      </Attribute>
```

Example of a part of the ProcessInfoMFWithComponent.xml file with data types that define the metric attributes in this metric family

```
    <Attribute>
      <Name>{http://im.ca.com/normalizer}ProcessInfo.Indexes</Name>
      <AttributeDisplayName>Indexes</AttributeDisplayName>
      <Description></Description>
      <Type>ObjectID</Type>
      <WriteOnPoll>false</WriteOnPoll>
      <Polled>false</Polled>
      <IsList>true</IsList>
      <IsDbColumn>false</IsDbColumn>
    </Attribute>
    <Attribute>
      <Name>{http://im.ca.com/normalizer}ProcessInfo.PID</Name>
      <AttributeDisplayName>PID</AttributeDisplayName>
      <Description>The process ID for the process in the OS.</Description>
      <Type>Integer</Type>
      <WriteOnPoll>false</WriteOnPoll>
      <Polled>false</Polled>
      <IsList>true</IsList>
      <IsDbColumn>false</IsDbColumn>
    </Attribute>
    <Attribute>
      <Name>{http://im.ca.com/normalizer}ProcessInfo.Names</Name>
      <AttributeDisplayName>Names</AttributeDisplayName>
      <Description></Description>
      <Type>String</Type>
      <WriteOnPoll>false</WriteOnPoll>
      <Polled>false</Polled>
      <IsList>true</IsList>
      <IsDbColumn>false</IsDbColumn>
    </Attribute>
    <Attribute>
      <Name>{http://im.ca.com/normalizer}ProcessInfo.Memory</Name>
      <AttributeDisplayName>Memory</AttributeDisplayName>
      <Description>The total amount of real system memory allocated to this process,
in kilobytes.</Description>
      <Type>Double</Type>
      <WriteOnPoll>false</WriteOnPoll>
      <Polled>true</Polled>
      <IsList>true</IsList>
      <IsDbColumn>true</IsDbColumn>
      <Baseline>true</Baseline>
      <Maximum>true</Maximum>
      <Minimum>true</Minimum>
      <Variance>true</Variance>
      <StandardDeviation>true</StandardDeviation>
      <Percentile>95</Percentile>
      <RollupStrategy>Avg</RollupStrategy>
```

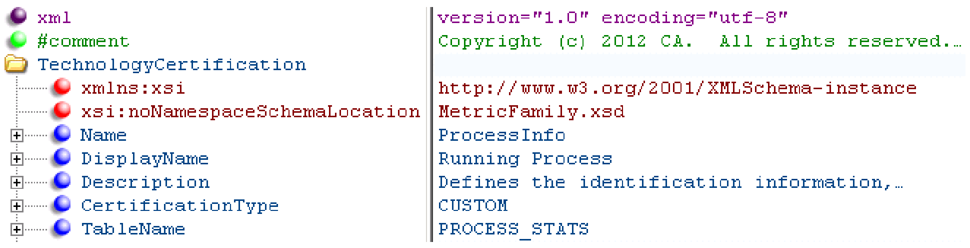
```
</Attribute>
```

More information:

Create a Custom Metric Family XML File (see page 38)

Add the Basic Information That Defines Your Custom Metric Family

The basic properties of your custom metric family help to distinguish it from other custom metric families you create. Also, these properties indicate where to store your collected metric data.



```
xml                                  version="1.0" encoding="utf-8"
#comment                             Copyright (c) 2012 CA.  All rights reserved.…
TechnologyCertification
   xmlns:xsi                         http://www.w3.org/2001/XMLSchema-instance
   xsi:noNamespaceSchemaLocation     MetricFamily.xsd
   Name                              ProcessInfo
   DisplayName                       Running Process
   Description                       Defines the identification information,…
   CertificationType                 CUSTOM
   TableName                         PROCESS_STATS
```

The tags included in this section of the XML file are as follows:

Name

Specifies the metric family name. Each metric family must have a unique name that identifies it internally within the system.

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

DisplayName

Specifies the metric family name to display in the user interface.

Description

Specifies the external description for the attribute.

CertificationType

Defines the type of metric family.

Note: Only CUSTOM is supported for user-defined metric families.

TableName

Specifies the database table name that stores the metrics that the metric family collects.

Note: The database table must be unique (not a value that another metric family uses) and composed of uppercase letters and the underscore character, for example, PROCESS_STATS.

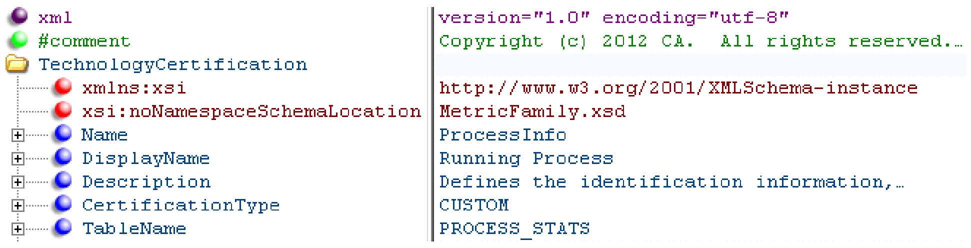
Note: For more information about each XML tag, see the inline documentation provided in the MetricFamily.xsd and Component.xsd files. For code examples, see the ProcessInfoMFWithComponent.xml file. This example file defines a metric family for gathering process metrics.

More information:

Create a Custom Metric Family XML File (see page 38)

Basic Properties

Provide Namespace and Schema information as shown in the TechnologyCertification section in the following illustration:



```
xml                                   version="1.0" encoding="utf-8"
#comment                              Copyright (c) 2012 CA.  All rights reserved.…
TechnologyCertification
    xmlns:xsi                         http://www.w3.org/2001/XMLSchema-instance
    xsi:noNamespaceSchemaLocation     MetricFamily.xsd
  ⊞ Name                              ProcessInfo
  ⊞ DisplayName                       Running Process
  ⊞ Description                       Defines the identification information,…
  ⊞ CertificationType                 CUSTOM
  ⊞ TableName                         PROCESS_STATS
```

The following information describes the elements in the TechnologyCertification section:

Name

Specifies the metric family name. Each metric family must have a unique name that identifies it internally within the system.

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

DisplayName

Specifies the metric family name to display in the user interface.

Description

Specifies the external description for the attribute.

CertificationType

Defines the type of metric family.

Note: Only CUSTOM is supported for user-defined metric families.

TableName

Specifies the database table name that stores the metrics that the metric family collects.

Note: The database table must be unique (not a value that another metric family uses) and composed of uppercase letters and the underscore character, for example, PROCESS_STATS.

Add the <AttributeList> Section

The <AttributeList> section defines the configuration or performance metrics you want to collect for your metric family. Even if many attributes are available for a device type, plan carefully which attributes are important for monitoring your infrastructure. The three common uses for attribute data are as follows:

- Reporting—These attribute values can change frequently. Typically, these values are polled at every cycle, providing volatile values that are used in reports to demonstrate changes over time. Examples include bytes in/out, bandwidth, and CPU or memory utilization.
- Configuration—These attributes provide data for identifying items. The data rarely changes. Therefore, this data is typically gathered during discovery only (that is, the data is not polled at every cycle). Examples include names, descriptions, and IDs.
- Reconciliation—The purpose of these attributes is to detect configuration changes in your environment. A detected change can trigger a discovery to ensure that your Data Aggregator data is accurate.

Each attribute can include the following tags:

<Name>

A unique internal name for the attribute (in qualified name format).

<Description>

This text appears as a tooltip in your CA Performance Center views.

<AttributeDisplayName>

This tag determines how your attributes are identified in reports and CA Performance Center dashboards.

<Type>

The data type of the attribute. The most-used types are *Integer*, *Long*, *Double*, *String*, or *ObjectID*.

<Polled>

When set to True, Data Aggregator collects this metric data during each polling cycle. When False, the data is collected only upon discovery of an item.

<IsList>

The attribute provides either scalar or table data. Consider the following definitions:

- Scalar attributes—These attributes return a single value at a time, which is stored with the device data. For example, an attribute for "Number of current processes" returns a single value.
- Table attributes—These attributes return a value for each row in a table. The example Process metric family uses table attributes. Each row corresponds to a process item.

Note: If this tag is True for an attribute, it must be set to True for every attribute in the same attribute list. In this case, it is required to define an Indexes (*ObjectID* type) and a Names (*String* type) attribute in the list.

<IsDbColumn>

This tag indicates that the attribute value is stored in the database table. For configuration metrics, set this tag value to False.

<RollupStrategy>

This tag specifies to calculate either a summation or an average of the individual poll value. This tag is required when <Polled> and <IsDbColumn> are set to True.

<Baseline>

When set to True, a baseline is calculated for this attribute and you add a corresponding baseline definition in the <BaselineList> section.

The following tags are also required when you include <RollupStrategy> or when <Baseline> is set to True:

Baseline

Indicates whether to calculate a baseline for this attribute. If it is set to True, a corresponding BaselineList definition must exist.

Maximum

Indicates whether to calculate the maximum of this attribute during rollup. Creates a 'max_' column in the database table.

Minimum

Indicates whether to calculate the minimum of this attribute during rollup. Creates a 'min_' column in the database table.

Variance

Indicates whether to calculate the variance of this attribute during rollup. Creates a 'var_' column in the database table.

StandardDeviation

Indicates whether to calculate the standard deviation of this attribute during rollup. Creates a 'std_' column in the database table.

RollupStrategy

Specifies the operation to perform every cycle during rollup of the individually polled values. The value can be 'Sum' or 'Avg' for calculating either a summation or an average.

<Filterable>

This tag indicates that the attribute can be used for monitoring profile filtering. When applying a filter, an attribute with this tag becomes available for the filter expression.

Note: For more information about each XML tag, see the inline documentation provided in the MetricFamily.xsd and Component.xsd files. For code examples, see the ProcessInfoMFWithComponent.xml file. This example file defines a metric family for gathering process metrics.

More information:

Create a Custom Metric Family XML File (see page 38)

AttributeList

The Attribute List lists the attributes that are collected and stored in the database table. The following information describes the elements that you use in the AttributeList section:

Name

Specifies the unique internal name that is used for naming the DB column. Name is specified in QName format:

```
{URI}FamilyName.AttrName
```

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

AttributeDisplayName

Specifies the value that appears in operator and administrator interfaces. AttributeDisplayName can be localized into multiple languages.

Description

Displays the attribute description in the user interface. The description also displays in tool tips when hovering over the attribute name.

Type

Indicates the data type of this attribute. The most frequently used data types are *Integer*, *Long*, *Double*, *String* or *ObjectID*.

Polled

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

IsList

If it is set to True, retrieves the attribute from a table. Component items are also created for each row in the table.

Note: Set the property to True for every attribute in the metric family that is stored in the database when you set this property to True for any attribute.

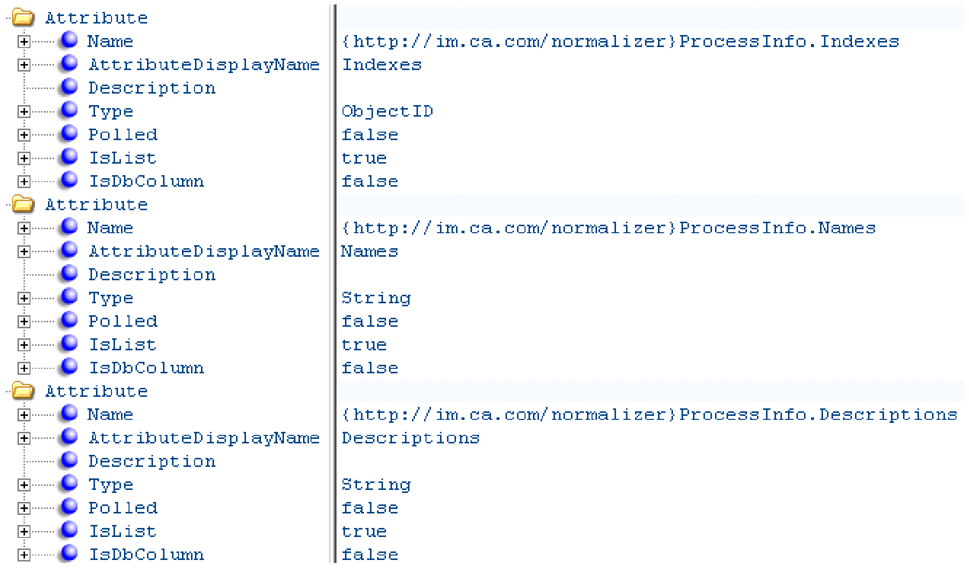
IsDbColumn

Stores its value in the database table. IsDbColumn is typically used for attributes that are polled every cycle and are included in reports.

Note: If you did not use ProcessInfoMFWithComponent.xml but downloaded an XML file from http://host:port/genericWS/metricfamilies/<name> to get started with your metric family, delete RollupExpression and Units nodes because they are deprecated.

General Attributes

In the following illustration, the Indexes, Names, and Descriptions attributes exist for any metric family:



```
Attribute
  Name                   {http://im.ca.com/normalizer}ProcessInfo.Indexes
  AttributeDisplayName   Indexes
  Description
  Type                   ObjectID
  Polled                 false
  IsList                 true
  IsDbColumn             false
Attribute
  Name                   {http://im.ca.com/normalizer}ProcessInfo.Names
  AttributeDisplayName   Names
  Description
  Type                   String
  Polled                 false
  IsList                 true
  IsDbColumn             false
Attribute
  Name                   {http://im.ca.com/normalizer}ProcessInfo.Descriptions
  AttributeDisplayName   Descriptions
  Description
  Type                   String
  Polled                 false
  IsList                 true
  IsDbColumn             false
```

Any supporting vendor certification can provide URIs exposed by the metric family, such as:

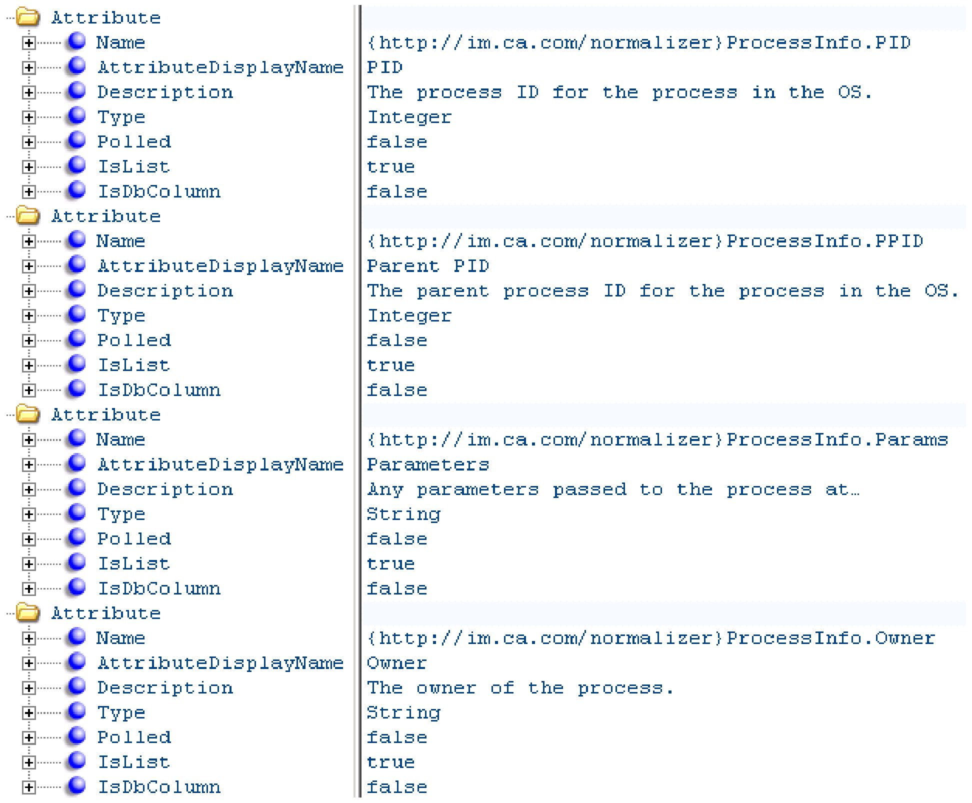
```
{http://im.ca.com/normalizer}Name.Indexes
```

```
{http://im.ca.com/normalizer}Name.Names
```

```
{http://im.ca.com/normalizer}Name.Descriptions
```

Simple Attributes

In the following illustration, for most attributes only the value is stored in the database. No further processing, such as evaluation of a baseline, is performed.



Attribute	
Name	{http://im.ca.com/normalizer}ProcessInfo.PID
AttributeDisplayName	PID
Description	The process ID for the process in the OS.
Type	Integer
Polled	false
IsList	true
IsDbColumn	false
Attribute	
Name	{http://im.ca.com/normalizer}ProcessInfo.PPID
AttributeDisplayName	Parent PID
Description	The parent process ID for the process in the OS.
Type	Integer
Polled	false
IsList	true
IsDbColumn	false
Attribute	
Name	{http://im.ca.com/normalizer}ProcessInfo.Params
AttributeDisplayName	Parameters
Description	Any parameters passed to the process at…
Type	String
Polled	false
IsList	true
IsDbColumn	false
Attribute	
Name	{http://im.ca.com/normalizer}ProcessInfo.Owner
AttributeDisplayName	Owner
Description	The owner of the process.
Type	String
Polled	false
IsList	true
IsDbColumn	false

A supported vendor certification must provide some metric family-exposed URIs in the following form:

```
{http://im.ca.com/normalizer}Name.AttributeName
```

Add the <BaselineList> Section

When analyzing your environment, knowing when an item is operating outside of a normal range is valuable information. To help you determine what is "normal," Data Aggregator can calculate baseline values.

Note: For more information about baselines in reports, see the *Data Aggregator Administrator Guide*.

For a custom metric family, you can instruct Data Aggregator to calculate baseline information for an attribute. The <BaselineList> section in your custom metric family indicates how to calculate the baseline information for a given attribute.

Important! You must define a baseline calculation in this section for each attribute in the <AttributeList> section with a <Baseline_> tag set to True.

In this section, your options for setting up baseline calculations are as follows:

Name

Specifies the baseline definition name.

ID

Specifies a unique identifier for the baseline definition. The value must be unique within the set of all baseline definitions for this metric family.

PerformanceMetric

Specifies the name of the metric for which the baseline is calculated. Specify Name in AttributeList; that attribute must have its Polled property set to True.

StartDate

Deprecated. If it is present, set it to 0.

EndDate

Deprecated. If it is present, set it to 0.

DaysOfWeek

Deprecated. If it is present, set it to 0.

Period

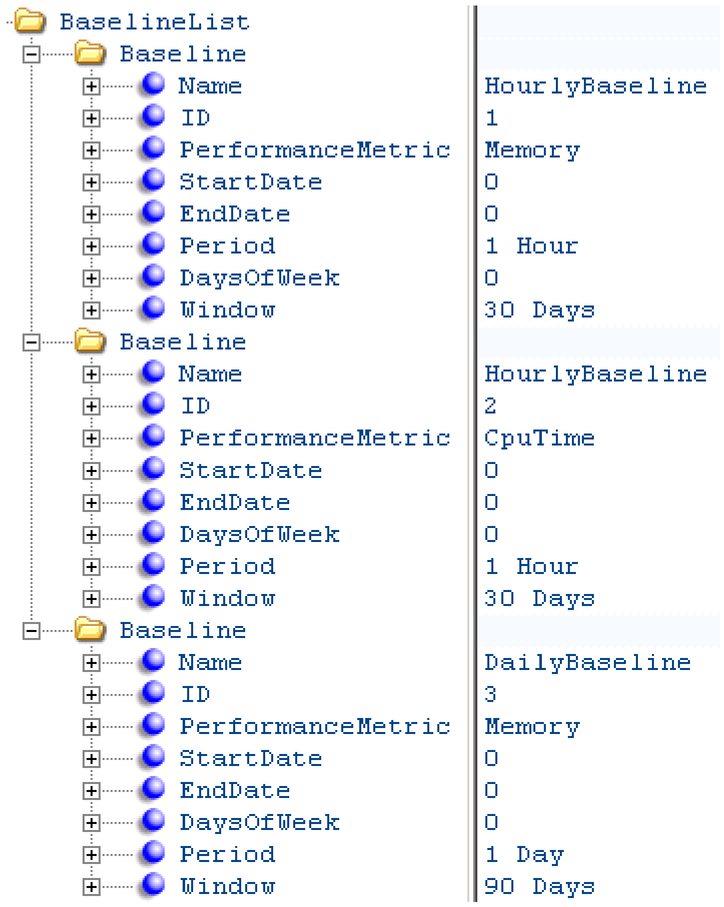
Specifies to calculate either hourly or daily baseline.

Window

Deprecated. If it is present, set it to 30 days.

Note: For more information about each XML tag, see the inline documentation provided in the MetricFamily.xsd and Component.xsd files. For code examples, see the ProcessInfoMFWithComponent.xml file. This example file defines a metric family for gathering process metrics.

The following illustration provides an example of using BaselineList elements:



```
BaselineList
  Baseline
    Name                 HourlyBaseline
    ID                   1
    PerformanceMetric    Memory
    StartDate            0
    EndDate              0
    Period               1 Hour
    DaysOfWeek           0
    Window               30 Days
  Baseline
    Name                 HourlyBaseline
    ID                   2
    PerformanceMetric    CpuTime
    StartDate            0
    EndDate              0
    DaysOfWeek           0
    Period               1 Hour
    Window               30 Days
  Baseline
    Name                 DailyBaseline
    ID                   3
    PerformanceMetric    Memory
    StartDate            0
    EndDate              0
    DaysOfWeek           0
    Period               1 Day
    Window               90 Days
```

More information:

Create a Custom Metric Family XML File (see page 38)

Add the <ComponentList> Section

A *component* is an item that is associated with a device (for example, a device can be associated with CPUs, interfaces, and process components). Using the component type helps you categorize the items that are associated with a device. In a metric family, table attributes require a list of the associated components. For components that do not currently exist in Data Aggregator, your metric family XML must also define these new components.

The component information in your metric family XML file serves the following two purposes:

- Discovery—When discovery creates items for a device, the component value identifies a category (facet) for the item.
- Synchronization—During synchronization between Data Aggregator and CA Performance Center, the component definition maps the items, and it determines how they appear in the interface.

In the <ComponentList> section, list the facets you want to create for any component items that are associated with your custom metric family. For example, a metric family that collects process metrics is associated with a device that is running processes. We can define a "process" facet in this section, as follows:

```
<Component>{namespace}Process</Component>
```

This example indicates that all items created for our processes metric family are categorized as a "process" component.

Important! Each component listed in this section must be defined in Data Aggregator. If a component is not defined by default in Data Aggregator, define it in the <ComponentDefinitionList> section.

Note: For more information about each XML tag, see the inline documentation provided in the MetricFamily.xsd and Component.xsd files. For code examples, see the ProcessInfoMFWithComponent.xml file. This example file defines a metric family for gathering process metrics.

More information:

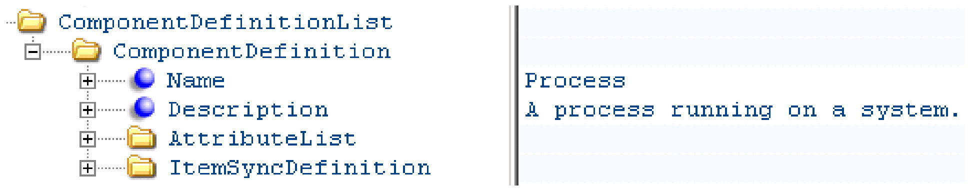
Create a Custom Metric Family XML File (see page 38)

Add the <ComponentDefinitionList> Section

The <ComponentDefinitionList> section defines details about your components. The details include definitions of more attributes to store in the database for a component, and information about synchronization of components and their attributes with CA Performance Center.

This section is needed only when the following conditions are met:

- Your custom metric family XML file specifies attributes with the <IsList_> tag set to True.
- Components in the <ComponentList> section are not predefined components.



```
ComponentDefinitionList
  ComponentDefinition
    Name                    Process
    Description             A process running on a system.
    AttributeList
    ItemSyncDefinition
```

Include the following tags in this section of your custom metric family:

<Name>

<Description>

<CertificationType>—This tag is always set to CUSTOM.

AttributeList

(Optional) Defines the list of attributes to calculate and store in the database for the component. Use component attributes to make more information of a component (such as arguments of a process component) available in CA Performance Center by synchronizing these attributes with CA Performance Center. This section includes the following tags for each attribute:

Name

Specifies a unique internal name in QName format.

Type

Specifies the data type of this attribute.

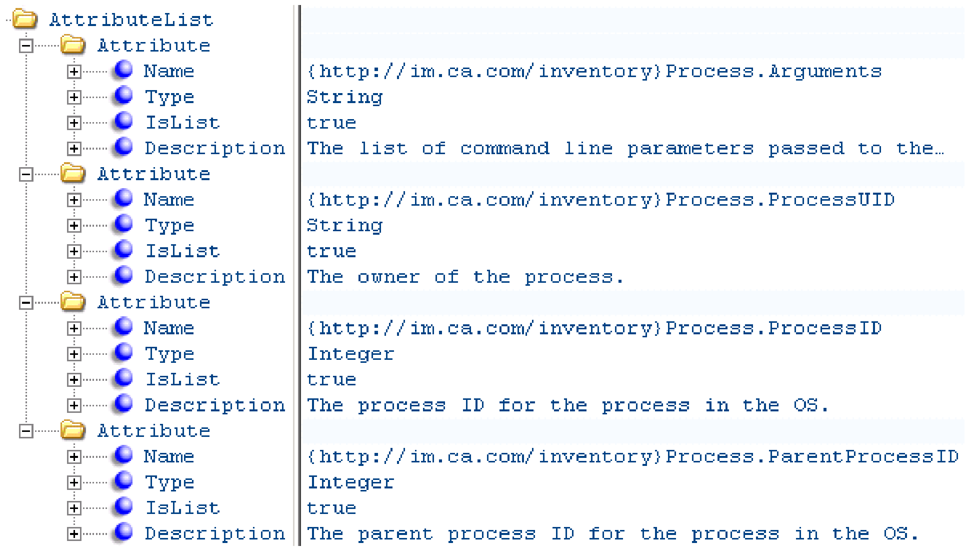
IsList

Indicates whether this attribute is scalar or table based. Typically this value equals the value of the corresponding attribute that is collected for the metric family.

Description

Specifies the description for this attribute that is visible in the user interface.

The following illustration provides an example of using AttributeList:



AttributeList	
Attribute	
Name	{http://im.ca.com/inventory}Process.Arguments
Type	String
IsList	true
Description	The list of command line parameters passed to the…
Attribute	
Name	{http://im.ca.com/inventory}Process.ProcessUID
Type	String
IsList	true
Description	The owner of the process.
Attribute	
Name	{http://im.ca.com/inventory}Process.ProcessID
Type	Integer
IsList	true
Description	The process ID for the process in the OS.
Attribute	
Name	{http://im.ca.com/inventory}Process.ParentProcessID
Type	Integer
IsList	true
Description	The parent process ID for the process in the OS.

ItemSyncDefinition

Provides the necessary details for synchronizing the component with CA Performance Center. ItemSyncDefinition also lists the additional attributes to send to CA Performance Center as component properties during the synchronization so that they are visible in the CA Performance Center user interface. Only the items with subtypes are synchronized, so the <ItemSyncDefinition> must define a base item type and subtypes. The base item type can be used to create a logical group and provide the default information and synchronization behavior available for all subtypes. In some use cases, a new base item type can improve performance. For most use cases, it is the “component” item type. The subtype items can provide more properties, specific to those subtypes, that must be synchronized. For example, the metric family can define a process arguments property for the process subtype. This section includes the following tags:

ItemTypeName

Defines the name of the item base type to use in CA Performance Center.

ItemSubTypeName

Defines the name of the item subtype to use in CA Performance Center.

ItemTypeLabel

Specifies the user interface label to use when displaying a single component of this type.

ItemTypeLabelPlural

Specifies the user interface label to use when displaying multiple components of this type.

IsDeviceComponent

Specifies whether this item is a component of a device (typically set to True). When set to True, this tag indicates that this component is listed in the 'Device Components' view.

GroupBy

Defines whether to group components of this type in CA Performance Center. WHen set to True, a new menu entry below the 'Inventory' menu lets you list these components in their own group view.

Context

Specifies whether to create a context type in CA Performance Center automatically for components of this type (typically set to False).

Categorize

Specifies whether to create a dynamic group in CA Performance Center for components of this type (typically set to False).

Mapped

Identifies the Data Aggregator instance to which the component belongs when multiple Data Aggregator instances are synchronized with CA Performance Center.

Note: This element is set to False because it is not currently supported.

ItemPropertyList

Lists all the attributes that are synchronized to CA Performance Center as a property of an item type. This section includes the following tags for each attribute:

Name

Defines the internal name for this property in CA Performance Center.

Note: The Name tag has a maximum length of 32 characters. Do not exceed this limit.

Label

Specifies the label to display for this property in the CA Performance Center user interface.

AttributeName

References the component attribute populating this property during CA Performance Center synchronization.

Justification

Specifies the text justification in the CA Performance Center user interface when displaying this property.

Default: Left

DisplayWidth

Defines the width of the column (in millimeters) to use in the CA Performance Center user interface when displaying this property.

OrderBySQL

Identifies the Data Aggregator instance to which the component belongs when multiple Data Aggregator instances are synchronized with CA Performance Center.

Note: This element is set to False because it is not currently supported.

DatabaseType/MaxLength

Specifies an override of the default maximum length of the inferred database type to use for this property.

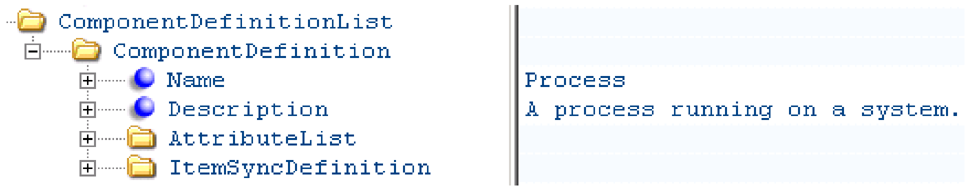
Note: For more information about each XML tag, see the inline documentation provided in the MetricFamily.xsd and Component.xsd files. For code examples, see the ProcessInfoMFWithComponent.xml file. This example file defines a metric family for gathering process metrics.

More information:

Create a Custom Metric Family XML File (see page 38)

ComponentDefinitionList

If your custom metric family introduces a new component, such as *Process* as shown in the following illustration, you define it in this section. ComponentDefinition specifies all properties for the creation of components for a metric family. The properties include definitions of more attributes to store in the database for a component, and information about synchronization of components and their attributes with CA Performance Center.



```
ComponentDefinitionList
  ComponentDefinition
    Name                  Process
    Description           A process running on a system.
    AttributeList
    ItemSyncDefinition
```

Synchronization with CA Performance Center

Synchronizing between Data Aggregator and CA Performance Center helps ensure that the component items appear in the Inventory menu. Further, synchronizing enables grouping in CA Performance Center for those items and enables context pages for their item types.

Data Aggregator synchronizes devices, interfaces, and various device component types to CA Performance Center. Only interfaces with data that can be used in reports are synchronized with CA Performance Center. These include interfaces that are being actively monitored, and retired interfaces, which have historical data.

Data Aggregator also synchronizes IP addresses for any interfaces that have one or more IP addresses. If it is available, Data Aggregator synchronizes subnet information for a given interface IP address.

If you want to synchronize other components of your custom metric family with CA Performance Center, you can do so using the <ItemSyncDefinition> section. This section defines the component's types and subtypes and how they appear in CA Performance Center.

Discovered devices and components usually take up to 5 minutes to begin to synchronize with CA Performance Center. Synchronization can take more than 5 minutes to complete when there are many items.

While synchronization is in progress, Data Aggregator waits to synchronize newly discovered devices and components until after the current synchronization finishes.

Synchronize an Item Property With a <Name_> Tag Value Greater Than 32 Characters

The <Name> tag value for the ItemPropertyList section should not exceed 32 characters. However, if it causes synchronization problems, you *can* synchronize an item property with a <Name> tag value that is greater than 32 characters.

Follow these steps:

1. Delete the metric family:
   a. Locate the IMDataAggregator/apache-karaf-2.3.0/deploy directory.
   b. Delete the XML files that you created for the metric family and the vendor certification you imported. They are named as follows:
      - im.ca.com-normalizer-<technology>.xml
      - im.ca.com-inventory-<technology>.xml
      - im.ca.com-certifications-snmp-<vendor>.xml
2. Restart Data Aggregator using the following command:

   ```
   /etc/init.d/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.
3. Click Admin, Data Sources in CA Performance Center.
4. Select a Data Aggregator and click the Resync button.

   The components for the remaining metric families synchronize between Data Aggregator and CA Performance Center.
5. Reduce the number of characters in the <Name> tag to fewer than 32 characters by editing your custom metric family XML file.
6. Import your corrected metric family XML file.

(Optional) Add the <ComponentReconciliation> Section

You can define reconciliation algorithms for your custom metric family in the <ComponentReconciliation> section. The reconciliation algorithm is used to support configuration changes for monitored devices. How a device supports a given metric family can be considered its configuration. For example, the number of interfaces and their configuration represent how a device supports the Interface metric family. As changes to the interfaces of a device occur, Data Aggregator must update its representation of those interfaces to help ensure that monitoring is up-to-date. The reconciliation algorithm is applied during this configuration update.

In support of certain metric families, individual components items are created within Data Aggregator. These component items represent the device configuration in support of a given metric family. Using the Interface metric family as an example, Data Aggregator creates port component items to represent each network interface of the device.

The initial discovery and creation of component items occurs when a monitoring profile is applied to the device through a device collection. Subsequent component discoveries occur as necessary to support changes on the monitored device. Upon subsequent component discovery, the reconciliation algorithm is applied to determine the changes that are required to update the set of component items.

When the configuration of a device changes, there are four scenarios to evaluate:

- New components – A device changed such that new component items are required in Data Aggregator.
- Unchanged components – A device configuration change did not alter some existing components, therefore, those component items are unchanged.
- Changed components – A device configuration altered some components, but they still exist. Component items in Data Aggregator must be updated to reflect the new configuration.
- Removed components – A device changes such that one or more components no longer exist on the device. These component items can be deleted or retired.

The reconciliation algorithm defines a set of attributes that can be compared between existing component items and new discovery results. The values of the attributes are compared to determine which new discovery results match existing component items and which represent new or changed components. This comparison produces the following results:

- If a discovery result does not match any existing component items, a new component item is created.
- When a discovery result matches all attributes of an existing component item, the component item is unchanged.
- If a discovery result finds a match, but some attribute values are different, the existing component item is updated with any new attribute values.
- Sometimes, discovery results determine that some existing component items do not match the new results in any way. In this case, those components are determined to no longer exist on the device and can be deleted or retired.

You can define two types of matches in a reconciliation algorithm; ExactMatch and BestofMatch.

ExactMatch

Indicates that a discovery result must match all of the specified attributes of a component item.

Example: ExactMatch Reconciliation Algorithm

The following example shows a reconciliation algorithm that defines an ExactMatch match type:

```
<ComponentReconciliation>
     <MatchAlgorithmList>
          <MatchAlgorithm>
               <AlgorithmType>Exact</AlgorithmType>
               <MatchAttributeList>
                    <MatchAttribute>
                         <Name>{http://im.ca.com/core}Item.Name</Name>
                    </MatchAttribute>
                    <MatchAttribute>
                         <Name>{http://im.ca.com/inventory}Process.Path</Name>
                    </MatchAttribute>
                    <MatchAttribute>

<Name>{http://im.ca.com/inventory}Process.Arguments</Name>
                    </MatchAttribute>
               </MatchAttributeList>
          </MatchAlgorithm>
     </MatchAlgorithmList>
</ComponentReconciliation>
```

This algorithm indicates that a discovery result must match all three attributes of an existing component item to be an exact match. If an exact match is not found, the following conditions produce these results:

- If any of the three attribute values are different, a new component item is created.
- If the three attribute values match but other attribute values are different, the existing component item is updated with the new attribute values.

BestofMatch

Specifies the least number of attributes that must match to that same number of attributes on an existing component item. Each attribute includes a "required" key. If the "required" key is set to "true," that attribute must be one of the matched attributes.

Example: BestofMatch Reconciliation Algorithm

The following example shows a reconciliation algorithm that defines a BestofMatch match type:

```
<ComponentReconciliation>
     <MatchAlgorithmList>
          <MatchAlgorithm>
               <AlgorithmType>BestOf</AlgorithmType>
               <LeastMatchCount>2</LeastMatchCount>
               <MatchAttributeList>
                    <MatchAttribute>
                         <Required>true</Required>
                         <Name>{http://im.ca.com/core}Item.Name</Name>
                    </MatchAttribute>
                    <MatchAttribute>
                         <Name>{http://demo/custom}Process.Path</Name>
                    </MatchAttribute>
                    <MatchAttribute>
                         <Name>{http://demo/custom}Process.Arguments</Name>
                    </MatchAttribute>
               </MatchAttributeList>
          </MatchAlgorithm>
     </MatchAlgorithmList>
</ComponentReconciliation>
```

This algorithm specifies the following requirements:

- A discovery result must match at least two of the listed attributes with an existing component item to be a match. If a BestofMatch match is not found, the following conditions produce these results:
  - If less than two attributes match, a new component item is created.
  - If at least two attribute values match but other attribute values are different, the existing component item is updated with the new attribute values.

    Note: One of the matching attribute values must be the required attribute.
- The "required" key is set to 'true' for the Name attribute. This means that the Name attribute must be one of the three attributes that match.

When a *BestOf* match algorithm is included in the reconciliation definition, the following results occur:

- If a discovery result matches only one existing component item, the result is a match.
- If a discovery result matches more than one existing component item, the number of matching attributes is considered. The existing component with the most matching attributes is considered the match for the discovery result.

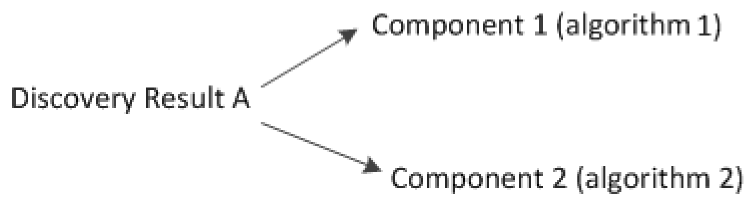
  Note: If the number of matching attributes is the same for multiple components, the matching component item is chosen randomly. As a result, the karaf log records a warning about this condition.

You can add multiple match algorithms to a reconciliation definition. When multiple match algorithms are added, they have match precedence. The match algorithm at the top of the MatchAlgorithmList has the highest precedence. The match type at the bottom has the lowest precedence.

When *more than one* match type is included in the reconciliation definition, the following results occur:

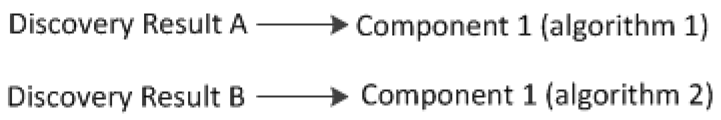
- If a discovery result matches only one existing component item, that is a match. The algorithm precedence does not matter.
- The algorithm with highest precedence prevails when a discovery result matches more than one existing component items. The match goes to the component item resulting from the highest precedence algorithm.

  For example:

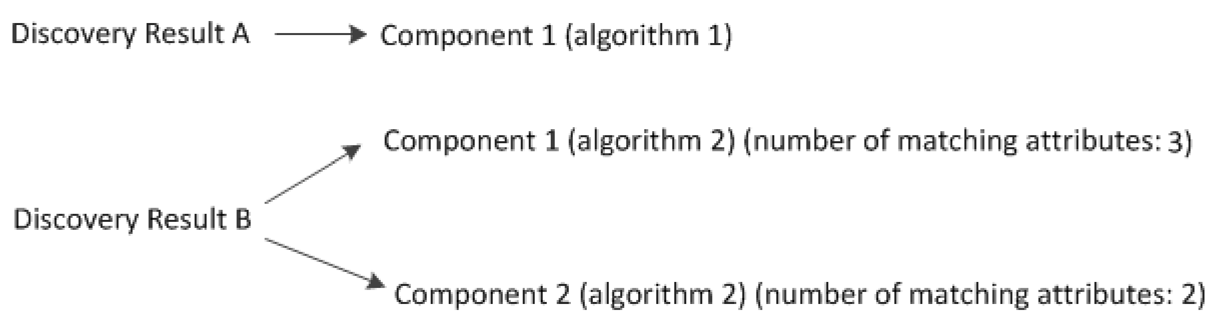
  Because algorithm1 has a higher precedence, Discovery Result A matches Component1.
- The algorithm with highest precedence prevails when more than one discovery result matches one existing component item. The match goes to the component item that matches as a result of the highest precedence algorithm.

  For example:

Because algorithm1 has a higher precedence, Discovery Result A matches Component 1.

For example:


Discovery Result A
Component 1 (algorithm 1)
Component 1 (algorithm 2) (number of matching attributes: 3)
Discovery Result B
Component 2 (algorithm 2) (number of matching attributes: 2)

In this example, Discovery Result B matches Component 1 with more attributes than it does with Component 2. Discovery Result A matches Component 1 because it matches with a higher precedence algorithm. Therefore, the match for Component 1 goes to Discovery Result A, and Discovery Result B matches Component 2.

Note: If you do not define a reconciliation algorithm, Data Aggregator uses the Item.Name attribute to match the components.

More information:

Create a Custom Metric Family XML File (see page 38)

(Optional) Add the <ReconfigDetectionAttr> Section

You can detect metric family changes using a dedicated scalar attribute in the metric family. This method is more efficient than add the <componentReconciliation> section; it performs better and generates much less network traffic. Only the scalar attribute is polled to determine if there is a change.

If a metric family has a scalar attribute to detect a change, you can specify that scalar attribute in the <ReconfigDetectionAttr> section.

Example: Configure Change Detection in the Interface Metric Family

1. Specify the change detection tag, <ReconfigDetectionAttr> to watch the PortReconfig attribute for change:

```
<ReconfigDetectionAttr>
 {http://im.ca.com/normalizer}NormalizedPortInfo.PortReconfig
</ReconfigDetectionAttr>
```

More information:

Create a Custom Metric Family XML File (see page 38)

Add the <ExpressionGroupList> Section

Expressions tell Data Aggregator how to calculate values for attributes of items that are either defined in or related to your metric family. The expressions included in your custom metric family determine how these values display in reports and dashboard views. Expressions are included in the <ExpressionGroupList> section.

If the attributes of your metric family are table attributes, your metric family XML must include the <ExpressionGroupList> section, and an expression group is required for at least the following two related items (destination certifications):

- {http://im.ca.com/core}Item
- {http://im.ca.com/inventory}DeviceComponent

You must also define an expression group for each component defined in your custom metric family. The tags in the <ExpressionGroupList> section are as follows:

- <ExpressionGroup>—Create one of these tags for Item, DeviceComponent, and each component type that your custom metric family monitors. This tag includes the following tags:
  - <DestCert>—The name of the related item to which the expression contributes.
  - <ExpressionList>—This tag contains the expressions that are defined for the attributes in the group. This tag includes the following tag:
    - <Expression>—Include at least one of these tags in each expression list. These tags define how to calculate the values for a given attribute. This tag includes the following tags:

      – <DestAttr>

      – <Expression>

Note: For more information about each XML tag, see the inline documentation provided in the MetricFamily.xsd and Component.xsd files. For code examples, see the ProcessInfoMFWithComponent.xml file. This example file defines a metric family for gathering process metrics.

More information:

Create a Custom Metric Family XML File (see page 38)

ExpressionGroupList

The following information describes performing calculations from elements in the attribute list. Often calculations are trivial assignments, such as the process ID.

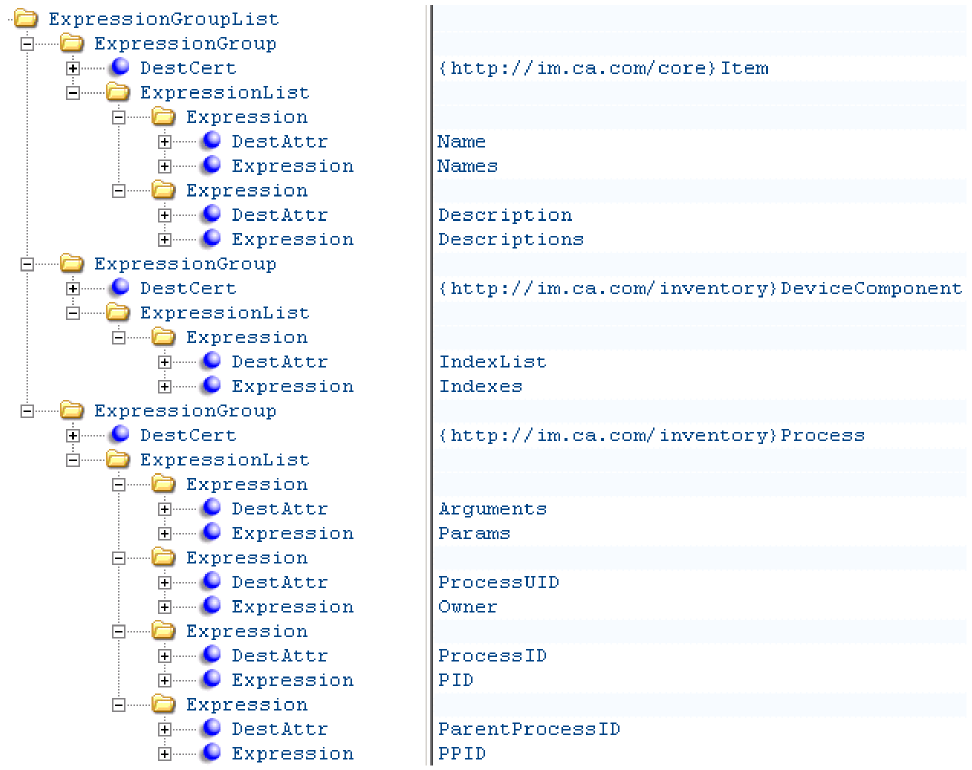
Note: Do not confuse the metric family and vendor certification ExpressionGroups. The metric family exposes its attributes with a URI of the following form:

`{http://im.ca.com/normalizer}`*`Name.AttributeName`*

The attributes are referred to in the vendor certification ExpressionGroup.

The metric family ExpressionGroup populates the database with different items that are addressed using a DestCert and a DestAttr URI. The value is computed in the Expression, which typically are trivial assignments of elements in the attribute list.

For example, the following illustration shows how {http://im.ca.com/inventory}Process.ProcessID is mapped to the PID attribute from the metric family AttributeList.



```
ExpressionGroupList
  ExpressionGroup
    DestCert                {http://im.ca.com/core}Item
    ExpressionList
      Expression
        DestAttr            Name
        Expression          Names
      Expression
        DestAttr            Description
        Expression          Descriptions
  ExpressionGroup
    DestCert                {http://im.ca.com/inventory}DeviceComponent
    ExpressionList
      Expression
        DestAttr            IndexList
        Expression          Indexes
  ExpressionGroup
    DestCert                {http://im.ca.com/inventory}Process
    ExpressionList
      Expression
        DestAttr            Arguments
        Expression          Params
      Expression
        DestAttr            ProcessUID
        Expression          Owner
      Expression
        DestAttr            ProcessID
        Expression          PID
      Expression
        DestAttr            ParentProcessID
        Expression          PPID
```

The following DestCert URIs must exist:

DestCert URI	DestAttr
{http://im.ca.com/core}Item	Name
{http://im.ca.com/core}Item	Description
{http://im.ca.com/inventory} DeviceComponent	IndexList
{http://im.ca.com/inventory} **component**	Attribute name as defined in ComponentDefinitionList (see page 57). In the example, the **component** Process provides the attributes Arguments, ProcessUID, ProcessID, and ParentProcessID.

Verify Your Custom Metric Family Results in a Test Environment

After you complete your custom metric family XML and import it into a test environment, verify your results. Verification is an important process to help ensure that your custom metric family produces the correct results in your production environment.

Important! Always create and verify your custom metric family in a *test environment first*. Creating a custom metric family requires you to edit the metric family XML file manually. Semantic errors in this XML file can cause unpredictable results.

Follow these steps:

1. Create a vendor certification on your test system using your custom metric family. Verify the following results as you create the vendor certification:
    a. The custom metric family XML file appears in the vendor certification wizard.
    b. The correct metric family attributes are visible when you select your custom metric family.
2. Verify that your vendor certification creates appropriate items during a discovery.
3. Verify that the items poll and collect the metric data you specified in your metric family XML file.
4. Verify that the collected data is correct.
5. Verify that the item data synchronizes correctly with CA Performance Center, if item synchronization is configured.
6. Run a discovery again and verify that the information updates correctly.

More information:

How to Create a Custom Metric Family (see page 36)

Import the Custom Metric Family XML File

After you create a custom metric family XML file and have verified it in a test environment, import it into your Data Aggregator installation.

Follow these steps:

1. Click Metric Families from the Monitoring Configuration menu for a Data Aggregator data source.

   A list of metric families displays, including factory and custom metric families. Factory metric families display a lock symbol.

2. Click Import.

3. Browse to select the custom metric family XML file, click Open, and then click Import.

   Your custom metric family is imported.

   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:

How to Create a Custom Metric Family (see page 36)

Automate the Removal of Retired Components

As an administrator, you can automate the removal of retired components from your network. Understand how to use the script that is included with Data Aggregator to delete retired components before you write a script to automate the process. For example, you can set up a weekly cron job to delete retired components that are a month old.

The remove_retired_items script that is included with Data Aggregator is comprised of two parts. The first part of the script identifies and returns data about retired components, which is based on the filter that you set. The second part of the script issues the delete of the retired component list. To automate the process, understand how this script was built.

Note: For information about using the remove_retired_items script, see the *Data Aggregator Administrator Guide*.

Example: Filter the List of Retired Components By a Device IP Address

In this example, you want to find all of the retired components for a device that has a primary IP address of 10.252.1.1. Filtering by IP address is a two-step process because no direct component filter by IP address is available. To filter retired components, first make note of the IP address for the device that the components are associated with. With the IP address information, you will determine the device item ID for the device. Then, using the device item ID, you will determine what the retired components are. Finally, you will delete the retired components.

Note: This example uses the curl command, but you can use any command that you are familiar with.

1. Create the filterDeviceIP.xml file. You will use this file to return the device item ID for the device that has a primary IP address of 10.252.1.1. The file must look like the following example:

```
<FilterSelect xsi:noNamespaceSchemaLocation="filter.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Filter>
    <And>
     <Device.PrimaryIPAddress
type="EQUAL">10.252.1.1</Device.PrimaryIPAddress>
    </And>
  </Filter>
</FilterSelect>
```

2. Run the following command:

   curl -X http://*hostname*:*port*/rest/devices/filtered -H "Content-Type: application/xml" -T "filterDeviceIP.xml" > returnedDeviceID.xml

   -X

   Creates the filter that you indicate.

   hostname*:*port

   Specifies the Data Aggregator host name and the port number.

   Default port: 8581

   -H

   Indicates the content type of the file that you are posting.

-T

Indicates the file that you are posting.

The following result is returned as an HTTP response:

```
<?xml version="1.0"?>
<DeviceList>
  <Device version="1.0.0">
    <ID>107881</ID>
    <PrimaryIPAddress>10.252.1.1</PrimaryIPAddress>
    <supportsOnDemandMFDiscovery>true</supportsOnDemandMFDiscovery>
    <SupportedProtocolsList>
      <SupportedProtocols>ICMP</SupportedProtocols>
    </SupportedProtocolsList>
    <DiscProfileID>107503</DiscProfileID>
    <HostName>rtp003723rts.ca.com</HostName>
    <RelatesTo>
      <MonitoredGroupIDList relatesURL="relatesto/monitoredgroups"
rootURL="monitoredgroups">
        <ID>509</ID>
      </MonitoredGroupIDList>
      <GroupIDList relatesURL="relatesto/groups" rootURL="groups">
        <ID>547</ID>
        <ID>530</ID>
        <ID>509</ID>
      </GroupIDList>
    </RelatesTo>
    <IsAlso>
      <IsA name="MetricFamilyDiscoveryHistory"
rootURL="devices/mfdiscoveryhistory"/>
      <IsA name="AccessibleDevice" rootURL="devices/accessible"/>
      <IsA name="Syncable" rootURL="syncable"/>
      <IsA name="IPDomainMember" rootURL="ipdomainmember"/>
    </IsAlso>
    <DataCoectionMgrId version="1.0.0">
      <DcmID>dcname.ca.com:8f53bc55-f442-42fc-9bd5-a907d0261421</DcmID>
    </DataCollectionMgrId>
    <Syncable version="1.0.0">
      <SyncID>-1</SyncID>
    </Syncable>
    <Item version="1.0.0">
      <DisplayName>router.ca.com</DisplayName>
      <CreateTime>Wed Feb 05 10:20:26 EST 2014</CreateTime>
      <Name>router.ca.com</Name>
    </Item>
    <IPDomainMember version="1.0.0">
      <IPDomainID>2</IPDomainID>
    </IPDomainMember>
    <DeviceMonitoringProfile version="1.0.0">
      <ConsolidatedMonitoringProfile>2509</ConsolidatedMonitoringProfile>
    </DeviceMonitoringProfile>
  </Device>
</DeviceList>
```

Device item ID 107881 is returned. The results also display detailed information about the device.

3. Create the filterRetired.xml file. You will use this file to return the retired components that are associated with the device whose device item ID is 107881. This file must look like the following example:

```
<FilterSelect xsi:noNamespaceSchemaLocation="filter.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Filter>
    <And>
        <DeviceComponent.DeviceItemID
type="EQUAL">107881</DeviceComponent.DeviceItemID>
    </And>
  </Filter>
  <Select use="exclude">
    <Item use="exclude">
      <DisplayName use="include"/>
    </Item>
  </Select>
</FilterSelect>
```

4. Run the following command:

   curl -X post http://*hostname*:*port*/rest/retired/filtered -H "Content-Type: application/xml" -T "filterRetired.xml" > returnedRetireItems.xml

   The following result is returned as an HTTP response:

```
<?xml version="1.0"?>
<RetiredList>
  <Retired version="1.0.0">
    <ID>128452</ID>
    <Item version="1.0.0">
      <DisplayName>GigabitEthernet0/239 - GigabitEthernet0/239</DisplayName>
    </Item>
  </Retired>
  <Retired version="1.0.0">
    <ID>128451</ID>
    <Item version="1.0.0">
      <DisplayName>GigabitEthernet0/238 - GigabitEthernet0/238</DisplayName>
    </Item>
  </Retired>
</RetiredList>
```

   Two retired components that fit the filter criteria are returned. The item IDs for the components are 128452 and 128451.

5. Create the deleteRetiredList.xml file. You will use this file to delete the returned list of retired components. The file must look like the following example:

   ```
   <DeleteList>
     <ID>128452</ID>
     <ID>128451</ID>
   </DeleteList>
   ```

6. Run the following command:

   curl -X post http://*hostname*:*port*/rest/retired/deletelist -H "Content-Type: application/xml" -T "deleteRetiredList.xml" > deletelistreponse.xml

   The following result is returned as an HTTP response:

   ```
   <?xml version="1.0"?>
   <DeleteListResult>
     <DeleteResult>
       <ID>128452</ID>
       <Error>SUCCESS</Error>
     </DeleteResult>
     <DeleteResult>
       <ID>128451</ID>
       <Error>SUCCESS</Error>
     </DeleteResult>
   </DeleteListResult>
   ```

   The retired components are successfully removed.

Chapter 6: Troubleshooting

This section contains the following topics:

Locate Details About Error Messages

Symptom:

I received a cryptic error message when the web service failed.

Solution:

To help debug the problem, locate the karaf.log file in your IMDataAggregator/apache-karaf-2.3.0/data/log directory for information. Anytime the web service fails to process a request, a stack trace and more detailed error message is printed in the log file. The log file often includes the specific tags that caused the problem.

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 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.