CA Performance Management Data Aggregator

REST Web サービスを使用した管理ガイド

2.4

ca technologies

CA Technologies 製品リファレンス

このマニュアルが参照している CA Technologies の製品は以下のとおりです。

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

CA への連絡先

テクニカル サポートの詳細については、弊社テクニカル サポートの Web サイト（http://www.ca.com/jp/support/）をご覧ください。

目次

第 1 章: 概要と基本操作 7

REST Web サービスについて 7
基本操作の使用 8

第 2 章: OpenAPI の概要 8

OpenAPI 認証情報の管理 9
OpenAPI へのアクセス 9
OpenAPI クエリについて 10
OpenAPI コントロール 11
OpenAPI クエリの作成 13
OpenAPI クエリのトラブルシューティング 14
メトリック値がテーブルに表示されません 14
クエリによって空のテーブルが作成されました 14

第 3 章: 一般的な REST Web サービス 15

一般的な REST Web サービス 15
一般的な REST Web サービス用の関係の管理 16

第 4 章: Data-Driven REST Web サービス 17

データ駆動型 REST Web サービス 17
データ駆動型 REST Web サービス用の XSD スキーマの表示 18
フィルタ属性 19
データ駆動型 REST Web サービス用の基本的関係の管理 22

第 5 章: Web サービスの使用 25

データ駆動型範囲のテナント ドメインへの制限 25
機密性が高く重要なデバイスをパフォーマンスに影響を与えずポーリングする 26
データ保持期間の変更 31
データ パージのスケジュール 33
同じ曜日の同一時間のベースライン平均が計算されるタイミングの変更 35
ロールアップ処理とベースライン計算の実行タイミングの変更 37
監視対象デバイスのプライマリ IP アドレスの変更 39

カスタム メトリック ファミリを作成する方法 41
メトリック ファミリの表示 42
メトリック ファミリ スキーマおよびサンプル ファイルのダウンロード 43
カスタム メトリック ファミリの XML ファイルの作成 44
テスト環境でのカスタム メトリック ファミリの結果の確認 76
カスタム メトリック ファミリの XML ファイルのインポート 77
廃止されたコンポーネントの削除の自動化 78

第 6 章: トラブルシューティング 83

エラー メッセージに関する詳細の表示 83
トラブルシューティング： メトリック ファミリが不完全 83
トラブルシューティング： メトリック ファミリがサポートされていません 85

第 1 章: 概要と基本操作

このセクションには、以下のトピックが含まれています。

REST Web サービスについて (P. 7)
基本操作の使用 (P. 8)

REST Web サービスについて

Data Aggregator は REST Web サービスを使用して、データの取得や、プロファイルとテナントまたはグループ間の関係の管理といった管理上の操作を扱います。これらの REST Web サービスでは、CA Performance Center のユーザ インターフェースがサポートされます。 または、API を使用して操作を実行できます。 管理者は、リクエストを実行するための REST クライアント ツール、またはリクエストの送信およびレスポンスの受信が可能な HTTP ツールを使用できます。

REST の Web サービスには、以下の 3 種類があります。

OpenAPI

独自の目的で Data Aggregator 設定情報およびメトリック データにアクセスする顧客向けにパブリック API として設計されました。 OpenAPI は、内部データ ストレージ構造が変わっても、将来的に変わらないように設計されています。

重要: OpenAPI は QueryBuilder インターフェースを使用します。 インターフェースのこのリリースは BETA 版であるので、実稼働環境での使用はお勧めできません。 拡張性および認可のオプションには制限があります（単一ユーザおよびパスワード）。

データ駆動型

CA Performance Center UI によって使用され、監視プロファイルやグループなどの Data Aggregator 設定情報の読み取りおよび変更を行います。

汎用

CA Performance Center UI によって使用され、メトリック ファミリおよび制限付きの SNMP ベンダー認定サポートを管理します。 汎用的な REST Web サービスは自己フィルタリングで、関係を管理するために URL 内の引数を使用しません。

基本操作の使用

*エンドポイント*は、グループ、監視プロファイル、テナントおよびデバイス認証などといったアイテムのタイプです。 特定のエンドポイントを以下のリクエスト方法または基本操作で使用して、結果のリストを返したり、アイテムを作成、更新、削除したりします。

注: 個別のメトリック ファミリおよび SNMP ベンダー認定アイテムは、ID ではなく名前を使用して指定されます。

重要: Data Aggregator REST Web サービスを使用して操作を実行するとき、コンテキスト タイプ ファイルを **application/xml** に設定してください。

基本的な操作には以下が含まれます。

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

指定されたタイプのアイテムすべてのリストを返します。getlist.xsd スキーマで戻り値データの形式を定義します。

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

指定された ID または認証名を持つ単一アイテムの詳細を返します。 XSD スキーマで戻り値データの形式を定義します。

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

指定された要素を持つ、指定されたタイプのオブジェクトを作成します。 XSD スキーマで戻り値データの形式を定義します。

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

指定されたアイテムの属性を更新します。 update.xsd スキーマで形式と所定のフィールドを定義します。

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

ID または認証名を使用して指定されるアイテムを削除します。

第 2 章: OpenAPI の概要

OpenAPI 認証情報の管理

デフォルトでは、OpenAPI へのアクセスは無効です。アクセスを有効にするには、OpenAPI に対してユーザ名とパスワードを設定します。ユーザ名およびパスワードは外部ファイルに格納されており、OpenAPI インターフェースを使用して変更することはできません。

以下の手順に従います。

1. 以下のファイルを開きます。
   `/opt/IMDataAggregator/apache-karaf-2.3.0/etc/com.ca.im.odata.auth.OpenAPIAuthenticationFilter.cfg`
2. ユーザ名およびパスワードを変更します。

   最後の文字の後にスペースがあれば、すべて削除します。 スペースが必要な場合は、パスワードが機能するようにそれを入力します。
3. ファイルを保存します。

   Data Aggregator を再起動せずに、新しいユーザ名およびパスワードの使用を開始します。 Data Aggregator は新しい認証情報をすぐに認識します。

OpenAPI へのアクセス

システム管理者がユーザ名とパスワードを設定するまで、OpenAPI へのアクセスは制限されます。 提供されたユーザ名とパスワードを使用してログインすると、同じセッションで認証情報を再度指定する必要はありません。 ブラウザ Cookie をクリアするか、またはコンピュータを再起動するまで、セッションはタイムアウトしません。

OpenAPI Query Builder のベース URL を以下に示します。
`http://`*`hostname`*`:`*`port`*`/odataquery`

hostname:*port*

Data Aggregator のホスト名およびポート番号を指定します。

デフォルト ポート： 8581

OpenAPI クエリについて

設定およびポーリングされたデータを確認用としてエクスポートするには、OpenAPI を使用します。OpenAPI を使用すると、カスタム Query URL を作成するクエリを実行できます。カスタマイズされたデータは、いつでもこの URL にアクセスして表示できます。 内部データ ストレージ構造が変わっても OpenAPI は変わりません。

データ駆動型または一般的な Web サービスとは異なり、OpenAPI はパブリック API であり、カスタマによる利用を目的としています。 OpenAPI には OData 2.0 業界標準が使用されています。 この標準に関するドキュメントは OData 2.0 Standard http://www.odata.org/documentation/odata-version-2-0/ の Web ページを参照してください。

注: スキーマ XML の記述を確認するには、メタデータを参照してください。この記述は OpenAPI に関する詳細情報を提供します。 メタデータを表示するには、以下の URL に移動します。
`http://<yourDA>:8581/odata/api/$metadata`

新しいメトリック ファミリを Data Aggregator に追加した場合は、Data Aggregator を再起動して OpenAPI 内の OData スキーマを更新します。Data Aggregator を再起動するには、以下のコマンドを実行します。
`/etc/init.d/dadaemon restart`

重要: QueryBuilder インターフェースのこのリリースは BETA 版であるので、実稼働環境での使用はお勧めできません。 拡張性および認可のオプションには制限があります（単一ユーザおよびパスワード）。

OpenAPI コントロール

OpenAPI Query Builder では、いくつかのタイプのコントロールを使用します。 これらのコントロールを使用して、以下の Query URL を作成およびエクスポートします。

トークン

トークンは OpenAPI クエリ構文の論理要素であり、これによって完全なクエリが構成されます。［Specify the Query］フィールドをクリックすると、トークンが表示されます。 トークンでは、クエリに含めるエンティティ、列、メトリック ファミリ、メトリック、または時間範囲のタイプを選択できます。 各トークンが選択されると、Query URL がそれに応じて更新されます。

例:

列、フィルタ、時間範囲、メトリック ファミリ、コンポーネント

フィルタ コントロール

論理式に基づくカスタム フィルタを作成できます。 フィルタ コントロールでは、AND および OR の演算子を使用して式を作成できます。単一のルールまたはルールのグループを管理することができ、それぞれに適用する演算子（AND または OR）を選択できます。

例:

列フィルタ、メトリック フィルタ

フォーマット コントロール

表示される列や出力の形式など、データのフォーマット スタイルを設定できます。各フォーマット コントロールが選択されると、それに応じて Query URL が更新されます。 ［Optionally Specify the Formatting］を選択します。

例:

ソート、結果制限、結果形式

以下の 2 種類のデータ テーブルが使用できます。

追加のメトリックを使用した HTML テーブル

集計された最小、最大、および平均のメトリックを表示できます。対応するアイテムの詳細なメトリック データを表示するには、虫めがねアイコンをクリックします。

エクスポートによる HTML テーブル

メトリック列は空で表示されますが、設定情報をエクスポートできます。

ボタン

Query URL をクリップボードにコピーするなどのアクションを UI で実行できます。

例:

コピー（  ）、ルールの追加、グループの追加、リセット

事前定義済みのクエリ フィルタ（オプション）

OpenAPI クエリを作成するには、カスタマイズ可能なテンプレートとして標準の選択内容を使用できます。 これらのテンプレートは、独自のクエリを構築する上で出発点となります。事前定義済みクエリ フィルタを選択し、追加のトークンとフィルタ コントロールを選択して、カスタム クエリを作成します。テンプレートを追加した後で、ユーザのニーズに合うようにフィルタ コントロールを変更します。

例:

特定グループ内のルータ

OpenAPI クエリの作成

OpenAPI Query Builder では、Query URL を作成できます。 クエリ内のすべてのパラメータには、対応するパラメータが OpenAPI によって Query URL に追加されます。使用可能なパラメータは、コンポーネント、デバイス、ルータ、またはクエリを実行するその他のアイテムによって異なります。

以下の手順に従います。

1. ［Specify the Query］フィールドをクリックして、クエリを開始します。

   トークンのリストが表示されます。

2. トークンまたはレシピを選択します。

   ［Specify the Query］フィールドにトークンまたはレシピが表示されます。

   Web ブラウザのアドレス バーが更新され、［Specify the Query］フィールドで加えた変更内容が表示されます。 自分または別のユーザが引き続き後でクエリの編集を行う場合は、記録用にこの URL をコピーし保存します。

3. 使用可能なトークンを使用して、クエリを完成させます。

4. ［Format］および［Payload］オプションをクリックして、データのフォーマットをカスタマイズします。

   ソート方法、返すアイテムの数、およびデータの形式を選択します。

5. （オプション） ［Run］ボタンをクリックします。

   OpenAPI によってクエリが実行され、選択した形式で結果が表示されます。

6. コピー（ ）ボタンをクリックします。

   Query URL がクリップボードにコピーされ、Web ブラウザまたは REST ツールに貼り付けできます。

   すべての OpenAPI クエリのベース URL は以下のとおりです。
   `http://hostname： port/odata/api/`

重要: 大きな結果セットを返すクエリは、システムに悪影響を及ぼす可能性があります。 要求に関連する結果のみを返すようにクエリを改良することをお勧めします。

OpenAPI クエリのトラブルシューティング

メトリック値がテーブルに表示されません

症状:

クエリを実行し、メトリック値を持つテーブルを生成しました。しかし、テーブルを開いても、結果にメトリック値が表示されません。

解決方法:

OpenAPI では現在、特定の時間範囲について、複数のデータ サンプルを単一の値に集約する機能をサポートしていません。

クエリの結果をプレビューするには、以下に示すような、適切な出力形式を選択します。

- 追加のメトリックを使用した HTML テーブル
- JSON
- XML

クエリによって空のテーブルが作成されました

症状:

クエリを実行した後、結果として得られる Query URL に空のテーブルが表示されます。

解決方法:

テーブル形式でメトリック列を表示するように選択した場合は、アイテム名など少なくとも 1 つの設定列を選択します。もう 1 つのソリューションは、「エクスポートによる HTML テーブル」以外の形式を選択するというものです。

第 3 章: 一般的な REST Web サービス

このセクションには、以下のトピックが含まれています。

一般的な REST Web サービス (P. 15)
一般的な REST Web サービス用の関係の管理 (P. 16)

一般的な REST Web サービス

メトリック ファミリおよび一部の SNMP ベンダー認定サポートの管理には、一般的な REST Web サービスを利用できます。一般的な REST Web サービスは自己フィルタリングで、関係を管理するために URL 内の引数を使用しません。

一般的な Web サービス用のベース URL は以下のとおりです。

`http://hostname:port/genericWS`

hostname:*port*

Data Aggregator のホスト名およびポート番号を指定します。

デフォルト ポート： 8581

ブラウザで以下の URL にアクセスし、XSD、URI、サポートされた HTTP メソッド、属性および関係などの一般的な Web サービスについての詳細を取得します。

- システムでユーザが対面する一般的な Web サービスすべての詳細を表示するには、以下を参照します。

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

- 指定されたエンドポイントの詳細を表示するには、以下を参照します。

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

一般的な REST Web サービス用の関係の管理

一般的な REST Web サービスは、関係を管理するために URL 内の引数を使用しません。 代わりに、一般的な REST Web サービスは、関係の管理を基本的な操作に頼っています。 エンドポイントは、情報を公開するために自身をフィルタにかけます。 これらの方法は、メトリック ファミリと SNMP ベンダー認定の関係を管理するのに使用されます。

一般的な REST Web サービス用の関係は以下のメソッドを使用して表示、作成、削除されます。

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

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

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

パラメータ

hostname:*port*

Data Aggregator のホスト名およびポート番号を指定します。

デフォルト ポート： 8581

endpoint/*name*

メトリック ファミリまたは SNMP ベンダー認定エンドポイントの名前を指定します。

例: SNMP ベンダー認定に関連するメトリック ファミリのリスト

指定された SNMP ベンダー認定に関連するメトリック ファミリのリストを返すには、以下の URL 構文を使用します。

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

第 4 章: Data-Driven REST Web サービス

このセクションには、以下のトピックが含まれています。

データ駆動型 REST Web サービス (P. 17)
データ駆動型 REST Web サービス用の XSD スキーマの表示 (P. 18)
フィルタ属性 (P. 19)
データ駆動型 REST Web サービス用の基本的関係の管理 (P. 22)

データ駆動型 REST Web サービス

データ駆動型 REST Web サービスは、ほとんどの Data Aggregator Web サービス（プロファイルおよびグループの監視など）で使用されます。

データ駆動型 Web サービス用の基準 Web サービス URL は次のとおりです。

`http://hostname:port/rest`

hostname:*port*

Data Aggregator のホスト名およびポート番号を指定します。

デフォルト ポート： 8581

ブラウザの以下の URL にアクセスし、XSD、URI、サポートされた HTTP メソッド、属性、および関係などのデータ駆動型 Web サービスおよびエンドポイントの詳細情報を取得します。

- システムでユーザが対面するデータ駆動型 Web サービスすべての詳細を表示するには、以下を参照します。

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

- 指定したエンドポイントの詳細を表示します。

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

データ駆動型 REST Web サービス用の XSD スキーマの表示

HTTP リクエストを実行する前に以下の手順を行います。

- エンドポイント用の XML スキーマ定義（XSD）を確認します。
- 戻り値の形式を確認するか、またはサービスが提供する XML をアップロードします。

XML ドキュメント内に配置されるコンテンツのアイテムは、それぞれエンドポイントの説明に忠実である必要があります。

エンドポイント用の XSD を取得するには、以下のパスのデータ駆動型 Web サービス URL を使用します。

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

operation

実行する操作のタイプを指定します。

全体として以下の操作がサポートされていますが、エンドポイントによってはサポートされていない操作もあります。

get

単一アイテム get 用の XSD を取得します。

getlist

エンドポイント アイテムのリスト用の XSD を取得します。

filterselect

GET Tunneling を使用して高度なフィルタ条件とリターン XML 形式を指定することを可能にします。

create

作成する際、すべての入力 XML が一致する必要がある XSD を取得します。

update

更新する際、すべての入力 XML が一致する必要がある XSD を取得します。

注: 操作がサポートされていない場合、Web サービスは失敗し、「403 Forbidden」メッセージを返します。

create、update、get、getlist、および filterselect で自動生成される XSD ファイルには、属性とメトリック ファミリの目的を説明するタグが含まれています。

フィルタ属性

データ駆動型 Web サービス用に XSD スキーマでリスト表示される属性（アイテム名、説明、他の類似属性など）をフィルタできます。 たとえば、監視プロファイルを、それらに含まれるメトリック ファミリでフィルタリングします。 この情報を使用して、メトリック ファミリを監視プロファイルに追加または削除するかどうかを決定できます。

以下の手順に従います。

1. Web ブラウザに以下の URL を入力します。

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

   利用可能なデータ駆動型 Web サービスがリスト表示されます。

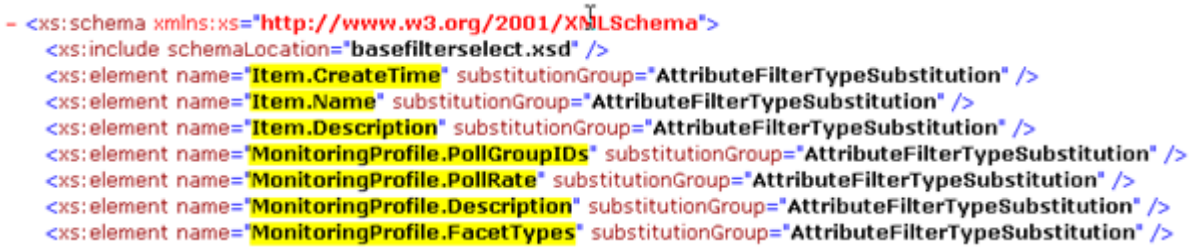
2. Web サービスをクリックします。

   その Web サービス用のドキュメント ページが表示されます。

3. 「フィルタリングされた GETList」メソッド下の URL をクリックします。

   XSD スキーマが表示されます。

4. 以下の例に示すように substitutionGroup="AttributeFilterTypeSubstitution" という条件を満たすエレメントを探します。 この情報を使用してどの属性をフィルタリングするかを決定します。



5. リクエストを送信しレスポンスを取得する REST クライアント エディタまたは HTTP ツールを開き、コンテンツ タイプを **application/xml** に設定します。
6. 以下のフィルタ条件を入力します。
   - URL: http://*hostname*:*port*/rest/*endpoint*/filtered/
   - HTTP method = POST

     このメソッドはフィルタ条件を定義する必要があります。
   - ［HTTP 要求］ペイン内の［Body］タブ上の基本的なフィルタ選択条件は以下のとおりです。

     ```
     <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

     属性の実際の値を指定します。

     elementName

     フィルタするエレメント名（属性）を指定します。

     注: またポーリング レートのみのように、選択条件も指定できます。 詳細については、以下の例を参照してください。

   結果は［HTTP レスポンス］ペインの［Body］タブに返されます。

例: フィルタおよび選択条件を使用してメトリック ファミリを含む監視プロファイルのリストを返します。

選択条件としてポーリング レートを使用して、メトリック ファミリが含まれる監視プロファイルを返すために以下の URL を入力します。

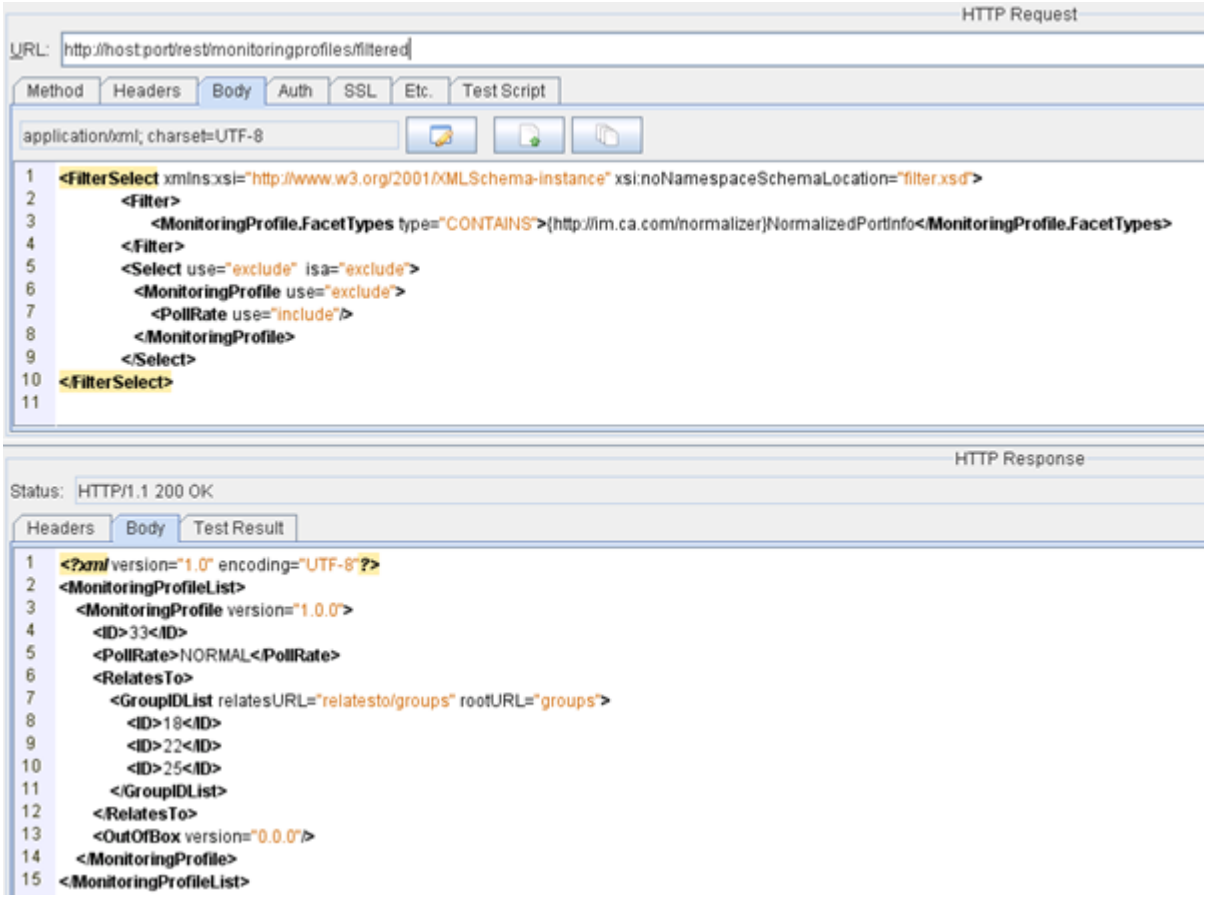
- メソッドと URL：

  ```
  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>
  ```

以下は、この例に対応するスクリーンショットです。 レスポンスには、NormalizedPortInfo メトリック（フィルタ条件）が含まれる監視プロファイルが表示されます。また、これらのプロファイルには指定された PollRate 属性（選択条件）だけが含まれます。



データ駆動型 REST Web サービス用の基本的関係の管理

データ駆動型 REST Web サービス用の関係は以下のメソッドを使用して作成し、削除します。

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

また、以下のメソッドを使用して、入れ子になった関係を表示させることもできます。

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

例

以下の URL を使用して、特定の監視プロファイル ID 781 と関連付けられたグループおよびデバイスをすべて参照します。

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

レスポンス URL は、指定された監視プロファイル ID に関連するデバイスとグループすべてをリスト表示します。この監視プロファイル ID には、現在の getlist.xsd ファイルにリスト表示されている属性が備わっています。

第 5 章: Web サービスの使用

このセクションには、以下のトピックが含まれています。

データ駆動型範囲のテナント ドメインへの制限 (P. 25)
機密性が高く重要なデバイスをパフォーマンスに影響を与えずポーリングする (P. 26)
データ保持期間の変更 (P. 31)
データ パージのスケジュール (P. 33)
同じ曜日の同一時間のベースライン平均が計算されるタイミングの変更 (P. 35)
ロールアップ処理とベースライン計算の実行タイミングの変更 (P. 37)
監視対象デバイスのプライマリ IP アドレスの変更 (P. 39)
カスタム メトリック ファミリを作成する方法 (P. 41)
廃止されたコンポーネントの削除の自動化 (P. 78)

データ駆動型範囲のテナントドメインへの制限

Data Aggregator の一部の機能について、操作のデータ駆動範囲を特定のテナント ドメインに制限し、グローバル リポジトリ全体の情報にアクセスできないようにすることができます。

以下の基本的方法を使用して、特定のテナント ドメインを持つエンドポイントの情報にアクセスします。

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

テナント ドメインを持つ Web サービスのすべてではなく一部を使用できます。

機密性が高く重要なデバイスをパフォーマンスに影響を与えずポーリングする

管理者は、重要なデバイスのどれが過剰なポーリングに敏感で、パフォーマンスの問題を引き起こす可能性があるかを認識しています。 しかし、重要なデバイスのパフォーマンスを確実に維持するには、これらの敏感なデバイスを何らかの方法で監視する必要があります。SNMP ポーリング制御を設定することによって、SNMP ポーリング リクエストをスロットル制御し、敏感なデバイスに対する影響を緩和できます。

デフォルトでは、SNMP ポーリングは次の 2 つの方法で制御されます。

- SNMP トラフィックしきい値 -- 1 度にデバイスに送信できるリクエストの数は最大 15 件です。 16 件以上のポーリングおよびディスカバリの SNMP リクエストはキューに登録され、ポーリング サイクルで可能なときにデバイスへ送信されます。 キューには最大 600 件のリクエストを登録できます。
- SNMP タイムアウトしきい値 -- 15 件以上の SNMP リクエストがタイムアウトすると、現在のポーリング サイクルが終了するまで、ポーリングが中断されます。 イベントが生成され、その状況がユーザに通知されます。

  注: 各ポーリング サイクルの最初の時点でポーリングが再開されます。タイムアウトがしきい値の 15 を超えていない場合、「クリア」イベントが生成されます。

これらのしきい値は両方とも、デバイスがポーリング リクエストで制圧されないように設計されています。 ただし必要に応じて、これらの SNMP ポーリングしきい値は設定できます。

たとえば、ポーリングに非常に敏感な古いルータを使っているとします。しかしこのルータは重要で、できるだけ頻繁にポーリングする必要があります。 既に監視プロファイルの調節を行い、不要なメトリック ファミリはポーリングから除外しています。 また、監視プロファイルにフィルタを適用し、ポーリングされるインターフェースの数も減らしました。 それでも、このルータはポーリングが原因でクラッシュします。この場合、この敏感なルータに合わせて、デフォルト SNMP ポーリング パラメータを調整するほかに解決策はありません。

以下のような特定のパラメータは、IPRangeList 内の IPRange セクションの個々の IP または IP 範囲のポリシーに追加できます。

MaxOutstandingRequests

指定された IP 範囲内のデバイスに送信される処理待ちリクエストの最大数。

MaxRequestSize

送信 SNMP リクエストに含まれる OID の数を制限します。 SNMP リクエストに含まれる OID の数が *MaxRequestSize* の値を超えている場合、送信リクエストは 2 つ以上のより小さなリクエストに分割されます。

一部の IP 範囲は IPRange セクションに含まれません。 グローバル設定では、*MaxRequestSizeDefault* パラメータを使用して OID 制限を設定します。

以下の手順に従います。

1. 以下を開いて、敏感なルータが属する IP ドメインの ID を検索します。

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

   hostname:*port*

   REST Web サービスにアクセスしている Data Aggregator のホスト名およびポート番号を指定します。

2. 以下の SNMP スロットル ポリシー リストからこの IP ドメイン ID を見つけ、対応するポリシー ID をメモします。

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

3. 単一の送信 SNMP リクエストに含める OID の数を決定します。 デバイスの中には、リクエストが大きすぎる場合にエラーを送信せずに無視するものもあります。 そのため、SNMP ポーラーがデバイスにアクセスできない結果となります。 Data Collector でこれらのデバイスを監視できるようにするには、*MaxRequestSize* 値を使用します。

   例

   インターフェース SNMP リクエストに 27 の OID があり、*MaxRequestSizeDefault* が 15 に設定されている場合、送信リクエストは 2 つのより小さなリクエストに分割されます。 一方のリクエストには 14 の OID が含まれ、他方には 13 の OID が含まれます。

例： 以下の SNMP スロットル ポリシーの例では、IP ドメイン「2」のポリシー ID が「601」であり、OID の数に制限がないことがわかります。

```
<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. リクエストを送信しレスポンスを取得する REST クライアント エディタまたは HTTP ツールを開き、コンテンツ タイプを application/xml に設定します。
5. IP ドメインの SNMP スロットル ポリシーを開き、次の条件を入力して編集します。
   - URL: http://*hostname*:*port*/rest/snmpthrottlepolicies/*policyID*

     policyID

     機密性の高いデバイスが属する IP ドメインの SNMP スロットル ポリシーに割り当てる一意の ID 番号を指定します。

     例： http://*hostname:port*/rest/snmpthrottlepolicies/601
   - HTTP method = PUT
   - ［HTTP 要求］ペインの［Body］タブにある［IP 範囲］で以下の値を調節します。
     - <MaxOutstandingRequests> – SNMP トラフィックしきい値
     - <TimeoutFailSafeThrottle> – SNMP タイムアウトしきい値

     注: これらの値は両方とも、すべての［IP 範囲］エントリに必要です。 どちらのパラメータも、値を 0 に設定することで無効にできます。
   - 以下の行を削除します。
     - <ID>
     - <IPDomainID>

   結果は［HTTP Response］ペインの［Body］タブに返されます。

例： この例では、デバイス 10.231.41.7 のみで、しきい値を "10" に下げます。 このデバイスの場合は、OID の数が 50 に制限されています。デフォルトしきい値およびその他の IP 範囲しきい値には、デフォルト値の「15」が引き続き使用されます。デバイス 10.231.41.1 から 10.231.41.255 では、SNMP リクエストの OID が 30 に制限されています。

```
<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>
```

注: 単一のデバイスまたは複数のデバイスで使用されるしきい値を調節できます。 適用されるしきい値は、［IP 範囲］の定義および［IP 範囲］の順序により決定されます。 ［IP 範囲］は優先度順にリスト表示されます。つまり、デバイスに適用される最初の［IP 範囲］によって、適用されるしきい値が決まります。

6. *MaxOutstandingRequestsDefault*、*TimeoutFailSafeThrottleDefault* および *QueueLength* の各パラメータは、常に update/POST XML のルート レベルに含めます。 値がデフォルトと異ならない場合でも、これらのパラメータを含めます。

   例：

   この PUT コマンドは、以下のポリシーを生成します。

```
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>
```

   このコマンドから以下のポリシーが生成されます。

```
   <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>
```

データ保持期間の変更

ポーリング済みデータ、時間単位/日単位/週単位のロールアップ データを Data Repository に保持するレートを変更することができます。 たとえば、ポーリング済みデータの保存期間を 30 日間に変更して、ディスク容量を節約することができます。 ニーズと環境に対して最適なバランスを見つけてください。

デフォルトでは、Data Repository にデータが保持される日数は以下のとおりです。

- ポーリング済みデータ： 45 日

  注: Data Aggregator の以前のリリースからこのリリースにアップグレードした場合、ポーリング済みデータの保存期間は、以前のデフォルトである 10 日から変更されません。
- 時間単位のロールアップ データ： 90 日
- 日単位のロールアップ データ： 365 日
- 週単位のロールアップ データ： 730 日

Data Repository がデータを保持できる最小の日数は以下のとおりです。

- ポーリング済みデータ： 2 日
- 時間単位のロールアップ データ： 8 日
- 日単位のロールアップ データ： 31 日
- 週単位のロールアップ データ： 366 日

以下の手順に従います。

1. Web ブラウザに以下の情報を入力します。
2. http://*hostname:port*/rest/globalretentiondefinition

   hostname:*port*

   Data Aggregator のホスト名およびポート番号を指定します。

   デフォルト ポート： 8581

   globalretentiondefinition Web サービス エンドポイント URL が開きます。
3. globalretentiondefinition に割り当てられてる ID をメモしておきます。

4. GtdRollupDataRetentionPeriod、DailyRollupDataRetentionPeriod、PolledDataRetentionPeriod および HourlyRollupDataRetentionPeriod があるエレメントを探します。 この情報は、どのタイプのデータに対して保存期間を変更するかを決定するのに役立ちます。
5. リクエストを送信しレスポンスを取得する REST クライアント エディタまたは HTTP ツールを開き、コンテンツ タイプを application/xml に設定します。
6. 以下の条件を入力します。
   - URL: http://*hostname*:*port*/rest/globalretentiondefinition/*ID*

     ID

     globalretentiondefinition に割り当てられる一意の ID 番号です。
   - HTTP method = PUT
   - ［HTTP Request］ペインの［Body］タブに変更する保持期間を入力します。

     以下に例を示します。

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

     重要： これらの行それぞれの先頭に空白がないことを確認します。もし空白があると PUT は失敗します。

     この例では、ポーリングされたデータの保持期間が 4 日間に変更されました。

   結果は［HTTP Response］ペインの［Body］タブに返されます。

   以下に例を示します。

   ```
   <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>
   ```

この例では、ポーリングされたデータの保持期間が 4 日間に変更されました。 週単位のロールアップ データ、日単位のロールアップ データ、および時間単位のロールアップ データ用のデフォルト保持期間は変更されていません。

データ パージのスケジュール

指定した保持期間を過ぎたデータをデータ リポジトリがすべてパージする頻度をスケジューリングします。開始する時、分、秒を変更できます。デフォルトでは、Data Aggregator は毎日午前 2:00:00 にデータをパージします。

以下の手順に従います。

1. Web ブラウザに以下の情報を入力します。
2. http://*hostname*:*port*/rest/globalretentionscheduledefinition

   hostname:*port*

   REST Web サービスにアクセスしている Data Aggregator のホスト名およびポート番号を指定します。

   globalretentionscheduledefinition Web サービス エンドポイント URL。
3. globalretentionscheduledefinition に割り当てられてる ID をメモしておきます。
4. StartMinute、StartHour および StartSecond があるエレメントを探します。この情報を使用して、古いデータのパージを開始する時、分、秒を変更するかどうかを決定します。

5. リクエストを送信しレスポンスを取得する REST クライアント エディタまたは HTTP ツールを開き、コンテンツ タイプを application/xml に設定します。
6. 以下の条件を入力します。
   - URL: http://*hostname*:*port*/rest/globalretentionscheduledefinition/*ID*

     ID

     globalretentionscheduledefinition に割り当てられる一意の ID 番号です。
   - HTTP method = PUT
   - ［HTTP 要求］ペイン内の［Body］タブ上で変更する時間の値を入力します。

     例：

     ```
     <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>
     ```

     重要: これらの各行の先頭に空白がないことを確認します。空白があると PUT 操作は失敗します。

     この例では、開始時間は 17 時に、開始分は 28 分に変更されました。

     注: パージ ジョブを無効にするには、<Enabled> を false に設定します。 再度パージ ジョブを有効にするには、<Enabled> を true に設定します。

   結果は［HTTP レスポンス］ペインの［Body］タブに返されます。

例：

```
<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>
```

この例では、指定された保持期間を過ぎたデータは毎日 17:28:00 にパージされます。

同じ曜日の同一時間のベースライン平均が計算されるタイミングの変更

はじめに、限られた量のデータが収集されると、過去のすべての曜日の同一時間に対するベースライン平均が計算されます。

より多くのデータが利用可能になると、計算方法が自動的に切り替わり、Data Aggregator は利用可能な過去の同じ曜日における 1 時間ごとのサンプルを平均して「標準」を確立します。

デフォルトでは、この自動切り替えが発生するのは、過去 12 週間に、同じ曜日の同一時間におけるデータ サンプルが少なくとも 3 つ利用できる場合です。

この自動切り替えが発生するタイミングを変更することができます。

以下の手順に従います。

1. Web ブラウザに以下の情報を入力します。

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

   hostname:*port*

   Data Aggregator のホスト名およびポート番号を指定します。

   デフォルト ポート： 8581

   sdshbaselineconfig Web サービスのエンドポイント URL が開きます。

2. 必要な最小データ ポイント数と過去の週の最大数について、現在の値を確認します。

3. リクエストを送信しレスポンスを取得する REST クライアント エディタまたは HTTP ツールを開き、コンテンツ タイプを application/xml に設定します。

4. 以下の条件を入力します。

   - HTTP method = PUT
   - ［HTTP Request］ペインの［Body］タブに、過去の週の最大数と、その期間に必要な最小データ ポイント数（ベースライン計算方法の切り替えをトリガする条件）を入力します。

     例：

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

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

     この例では、ベースライン平均の計算で利用可能な最小データポイント数が 5 に変更されました。 これらのデータポイントを検索する過去の週の数は 10 に変更されました。

同じ曜日の同一時間のベースライン平均が計算されるタイミングを変更する際は、以下の情報を考慮します。

- 両方の属性、または一方の属性のみを変更できます。
- 属性値は共に、1 以上の数値である必要があります。
- 上限は適用されません。 ただし、時間単位のロールアップの保持ポリシーでは上限が定義されます。 デフォルトでは、時間単位の保持レートは 90 日（およそ 12 週間のデータ）です。 過去の週の最大数を増加させる場合は、時間単位のロールアップの保持レートも増加させてください。
- MinimumNumberOfRequiredDataPoints 属性値は MaximumNumberOfWeeks 値以下である必要があります。

ロールアップ処理とベースライン計算の実行タイミングの変更

管理者はロールアップ処理とベースライン計算を実行するタイミングを変更できます。 これらの操作の実行時間の変更は、管理者が、Vertica を多用するこれらの操作が時間外に実行されるようにスケジュールできるようにします。 これらの操作が時間外に実行されると、レポートを生成するユーザは営業時間中に影響を受けずに済みます。

デフォルトでは、ロールアップ処理とベースライン計算は、毎時 30 分に実行されます。

以下の手順に従います。

1. Web ブラウザに以下の URL を入力します。

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

   hostname:*port*

   Data Aggregator のホスト名およびポート番号を指定します。

   デフォルト ポート: 8581

2. 構成アイテムに割り当てられてる ID をメモしておきます。

3. リクエストを送信し、レスポンスを取得する REST クライアント エディタまたは HTTP ツールを開きます。 以下の条件を入力します。

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

     hostname:*port*

     Data Aggregator のホスト名およびポート番号を指定します。

     デフォルト ポート： 8581

     ID

     構成アイテムに割り当てられる一意の ID 番号です。 この番号は、前の手順で確認しました。

   - HTTP メソッド： PUT
   - ロールアップ処理が開始および終了する希望時間を［HTTP Request］ペインの［Body］タブに入力します。

   デフォルトでは、以下の結果が得られます。

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

   StartHour

   ロールアップ処理が開始する時間（ローカル タイム ゾーンの時間）を 24 時間形式で定義します。

   EndHour

   ロールアップ処理が終了する時間（ローカル タイム ゾーンの時間）を 24 時間形式で定義します。 新しいロールアップを終了時間後に開始することはできませんが、進行中のロールアップを完了することはできます。

   注: これらの属性の詳細については、http://*hostname*:*port*/rest/rollups/config/documentation を参照してください。

例： この例では、ロールアップ処理をおよびベースライン計算が 20:00 ～ 7:00 のみに実行されるようにスケジュールを変更します。

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

<EndHour> では、指定の時間も含まれます。 この例では、EndHour として 6 を指定した場合、ロールアップ処理およびベースライン計算は、06:00 の時間帯に開始されますが、07:00 時には開始されません。 進行中の計算は完了させることができます。

重要: デフォルト スケジュールに変更を加えると、対応する間隔に関連してレポートにデータが表示されるまで大きな遅延が生じる場合があります。 たとえば、時間単位のロールアップに遅れが生じると、1 時間間隔で表示されるレポートは、時間単位のロールアップが実行されるまで最新の状態になりません。

監視対象デバイスのプライマリ IP アドレスの変更

管理者は、デバイスの IP アドレスが変わっても、監視対象デバイスが一貫性のあるデータを保持するようにしたいとします。 デバイスの IP アドレスが変更され、管理者がそれを更新しない場合、後続のディスカバリは新しい監視対象デバイスを作成できます。

プライマリ IP アドレスを変更する前に、以下の情報を考慮してください。

- IP アドレスを変更し、古い IP アドレスにアクセスできなくなった場合、デバイスに DNS ホスト名があれば、デバイスを監視する Data Collector インスタンスは、ホスト名の逆引きを実行して新しい IP アドレスのデバイス アイテムを検索および設定します。
- デバイスのプライマリ IP アドレスを、別のデバイスが使用する IP アドレスに変更した場合、エラー メッセージが表示されます。

注: 監視対象デバイスのプライマリ IP アドレスが変わる場合、イベントがデバイス上で生成されます。

監視対象デバイス上のプライマリ IP アドレスを変更するには、REST クライアント エディタまたは HTTP ツールを開き、以下の条件を入力します。

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

  hostname:*port*

  Data Aggregator のホスト名およびポート番号を指定します。

  デフォルト ポート： 8581

  deviceitemID

  プライマリ IP アドレスを変更している監視対象デバイスのデバイス アイテム ID 番号です。

  注: IP アドレスを既存の監視対象デバイスの IP アドレスに変更した場合、エラーが表示されます。

- HTTP メソッド： PUT
- 変更されたプライマリ IP アドレスを［HTTP Request］ペインの［Body］タブに入力します。

  以下に例を示します。

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

  IP

  変更されたプライマリ IP アドレスです。

例: この例では、監視対象デバイス上のプライマリ IP アドレスを 1.2.3.4 に変更します。

```
<Device version="1.0.0">

      <PrimaryIPAddress>1.2.3.4</PrimaryIPAddress>

</Device>
```

カスタム メトリック ファミリを作成する方法

ファクトリ メトリック ファミリは、監視する最も一般的なメトリック属性を定義します。 インストールにはいくつかのメトリック ファミリが含まれており、これらのメトリック ファミリは、大半のユーザのニーズを満たしています。 ただし、新しいメトリック属性のデータを収集する場合には、カスタム メトリック ファミリを作成できます。 たとえば、プロセス データの収集用のメトリック ファミリがない場合は、それを作成します。

カスタム メトリック ファミリは以下の詳細を定義します。

- 収集するメトリック
- 値の計算方法
- （オプション）値の表示方法

必要に応じて、新しいメトリック ファミリを使用することにより、ベンダー認定を作成して（それがまだ存在しない場合）プロセス用のメトリック属性を監視することができます。

重要: カスタム メトリック ファミリを作成し、*最初にテスト環境で*必ず確認してください。 カスタム メトリック ファミリを作成するには、メトリック ファミリの XML ファイルを手動で編集する必要があります。 この XML ファイルでのセマンティック エラーは、予測不能の結果を引き起こす場合があります。

注: Data Aggregator には、カスタムのベンダー認定とメトリック ファミリを作成するための基本的な方法と高度な方法があります。 基本的方法はより単純なプロセスであり、ユーザ インターフェースを使用して、サポートされた既存の技術（メトリック ファミリ）のベンダー サポートを追加することから構成されます。 この方法は、多くのユーザの要件を満たします。 一方、高度な方法は、ファクトリ認定の形式に基づいており、完全な機能セットを利用できます。 このガイドでは、基本的な認定方法について説明します。 高度な認定方法の詳細については、「Data Aggregator Power User Certification Guide」を参照してください。

カスタム メトリック ファミリの作成は複雑なプロセスです。カスタム メトリック ファミリを作成するには、以下の手順を注意して実行します。

1. 既存のメトリック ファミリを表示して、カスタム メトリック ファミリが必要かどうかを判断します (P. 42)。
2. スキーマおよびサンプル メトリック ファミリのファイルをダウンロードします。
3. カスタム メトリック ファミリの XML ファイルを作成します (P. 44)。
4. テスト環境でカスタム メトリック ファミリの結果を確認します (P. 76)。
5. カスタム メトリック ファミリの XML ファイルをインポートします (P. 77)。

重要: データの損失を回避するには、ベンダー認定、メトリック ファミリ、またはコンポーネントを作成または更新するたびに、展開ディレクトリを必ずバックアップします。

メトリック ファミリの表示

メトリック ファミリのリストを表示し、Data Aggregator インストールでどのメトリックがサポートされているかを確認します。 メトリック ファミリを表示すると、デバイス コレクション、ベンダー認定、および監視プロファイルとの関連を参照できます。 メトリック ファミリ、デバイス コレクション、およびデバイス タイプの関係を理解すると、デバイスを監視する方法を制御するうえで有用です。 また、環境を十分に監視するために、追加メトリック ファミリが必要かどうかを判断できます。

以下の手順に従います。

1. Data Aggregator データ ソースの［監視設定］メニューから［メトリック ファミリ］をクリックします。

   ファクトリおよびカスタムのメトリック ファミリも含めて、メトリック ファミリのリストが表示されます。事前定義済みの認定には鍵の記号が表示されます。

2. リストからメトリック ファミリを選択します。
3. 詳細情報を取得するにはタブをクリックします。

[メトリック]タブ

選択されたメトリック ファミリに存在するメトリック、および各メトリックのさまざまなプロパティを表示します。

[ベンダー認定優先度]タブ

選択されたメトリック ファミリに関連付けられているデバイス コレクションのリストを表示します。 通常、メトリック ファミリは単一のデバイス コレクションと関連付けられます。デバイス コレクションを選択すると、MIB ソース（ベンダー認定）の優先順位リストが表示されます。 この情報は、選択されたメトリック ファミリのデバイス コレクションに適用されるベンダー認定の優先順序を示します。

[監視プロファイル]タブ

関連する監視プロファイルとそのポーリング レートのリストを表示します。

詳細情報:

カスタム メトリック ファミリを作成する方法 (P. 41)

メトリック ファミリ スキーマおよびサンプル ファイルのダウンロード

カスタム メトリック ファミリ XML ファイルを作成する前に、メトリック ファミリ スキーマおよびサンプルのメトリック ファミリ XML ファイルをダウンロードして確認します。 自身の XML ファイルを作成するにはスキーマが必要です。 この例では、正常なメトリック ファミリをコード化する方法について実証します。 自身の XML ファイルを作成する前にこれらのファイルを確認しておくと、XML コンテンツの正確性を保証するために役立ちます。

以下のファイルを同じフォルダにダウンロードします。

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

注: それぞれの XML タグの詳細については、MetricFamily.xsd および Component.xsd ファイルで提供されるインライン ドキュメントを参照してください。 コード例については、ProcessInfoMFWithComponent.xml ファイルを参照してください。このサンプル ファイルでは、プロセス メトリックを収集するためのメトリック ファミリを定義します。

詳細:

カスタム メトリック ファミリを作成する方法 (P. 41)

カスタム メトリック ファミリの XML ファイルの作成

必要なファイルの収集が完了すると、カスタム メトリック ファミリ XML ファイルを作成する準備ができます。

注: それぞれの XML タグの詳細については、MetricFamily.xsd および Component.xsd ファイルで提供されるインライン ドキュメントを参照してください。 コード例については、ProcessInfoMFWithComponent.xml ファイルを参照してください。このサンプル ファイルでは、プロセス メトリックを収集するためのメトリック ファミリを定義します。

以下の手順に従います。

1. XML エディタを使用して MetricFamily.xsd から XML ファイルを作成します。カスタム メトリック ファミリはすべて <TechnologyCertification> をルート エレメントとして開始されます。

XML ファイルを作成する場合は、サンプルのメトリック ファミリ XML ファイルをコピーするか、または Web サービス（http://*ホスト名*: *ポート*/genericWS/metricfamilies）を使用してファクトリ XML ファイルをエクスポートして使用することをお勧めします。

重要: deploy ディレクトリからメトリック ファミリの XML ファイルをコピーしないでください。これらのファイルは、カスタム メトリック ファミリと互換性のない内部専用の形式を使用しています。

注: ファクトリ メトリック ファミリは変更できません。ただし、ファクトリ メトリック ファミリをコピーして、そのコピーを変更することは可能です。

2. XML ファイル内のコンテンツを以下のように編集します。
   a. メトリック ファミリ XML ファイルでのデータ型の使用状況を確認します (P. 46)。
   b. メトリック ファミリを定義する基本情報を追加します (P. 49)。
   c. <AttributeList> セクションを追加します。 (P. 51)
   d. <BaselineList> セクションを追加します。 (P. 57)
   e. <ComponentList> セクションを追加します。 (P. 59)
   f. <ComponentDefinitionList> セクションを追加します。 (P. 60)
   g. （オプション）<ComponentReconciliation> セクションを追加します (P. 66)。
   h. （オプション）<ReconfigDetectionAttr> セクションを追加します。(P. 73)
   i. <ExpressionGroupList> セクションを追加します。 (P. 73)
3. ファイルを保存します。

   カスタム メトリック ファミリが作成され、Data Aggregator へインポートする準備ができました。

   重要: カスタム メトリック ファミリを作成し、*最初にテスト環境で*必ず確認してください。カスタム メトリック ファミリを作成するには、メトリック ファミリの XML ファイルを手動で編集する必要があります。この XML ファイルでのセマンティック エラーは、予測不能の結果を引き起こす場合があります。

詳細:

カスタム メトリック ファミリを作成する方法 (P. 41)

メトリック ファミリ XML ファイル内のデータ型使用状況

このセクションでは、メトリック ファミリとベンダー認定の XML ファイルにおけるデータ タイプの使用法について説明します。

ObjectID

ポーリングの戻り値ではなくポーリングされた OID にアクセスするには、*ObjectID* タイプを使用します。

Integer/Long/Double

数値は通常、ベンダー認定内で *Integer* または *Long* タイプに格納されます。 メトリック ファミリでは通常、*Double* タイプを使用します。

BigInteger/Double

Data Aggregator は、64 ビット カウンタのポーリング、および高速インターフェースのデータ収集をサポートします。ネットワーク メディアの速度が速くなるほど、32 ビット カウンタがラップする最小時間は短くなります。 64 ビット カウンタを使用すると、カウンタがラップするまでの時間を延長することができ、通常のレートでのポーリングが可能になります。MIB2 の ifxTable では 64 ビット カウンタが提供され、ベンダー認定は通常、*BigInteger* タイプに格納されます。 メトリック ファミリでは通常、*Double* タイプを使用します。

String/OctetString

文字列値は、ベンダー認定内で *String* タイプに格納されます。メトリック ファミリでは、*OctetString* タイプを使用します。

QName

メトリック ファミリで *FacetTypes* 属性用に使用される特別なタイプです。

注: CA Performance Management で提供されるスキーマ ファイルには、タイプに関する詳細情報が含まれています。XML Notepad などのスキーマ対応 XML エディタでこの情報を使用して、XML ファイルの作成時に役立てることができます。

このベンダー認定で MIB オブジェクト属性を定義するデータ タイプを含む、ProcessInfoVCForEmpireMIB.xml ファイルの一部の例。

```
      <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>
```

このメトリック ファミリでメトリック属性を定義するデータ タイプを含む、ProcessInfoMFWithComponent.xml ファイルの一部の例。

```
    <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>
```

詳細:

カスタム メトリック ファミリの XML ファイルの作成 (P. 44)

カスタム メトリック ファミリを定義する基本情報の追加

カスタム メトリック ファミリの基本プロパティは、ユーザが作成したその他のカスタム メトリック ファミリとの区別に役立ちます。 また、これらのプロパティは、収集したメトリック データを格納する場所を示しています。

XML ファイルのこのセクションに含まれているタグは以下のとおりです。

Name

メトリック ファミリ名を指定します。 各メトリック ファミリには、システム内部で識別するための一意の名前が必要です。

注: この名前は外部には表示されません。 ユーザ インターフェースにメトリック ファミリ名を表示するには、DisplayName エレメントを使用します。

DisplayName

ユーザ インターフェースに表示するメトリック ファミリ名を指定します。

Description

属性の外部説明を指定します。

CertificationType

メトリック ファミリのタイプを定義します。

注: ユーザ定義のメトリック ファミリでは、CUSTOM のみがサポートされています。

TableName

メトリック ファミリが収集するメトリックを格納するデータベース テーブル名を指定します。

注: データベース テーブルは一意（別のメトリック ファミリが使用しない値）であり、大文字とアンダースコア文字から構成される必要があります。たとえば、PROCESS_STATS などです。

注: それぞれの XML タグの詳細については、MetricFamily.xsd および Component.xsd ファイルで提供されるインライン ドキュメントを参照してください。 コード例については、ProcessInfoMFWithComponent.xml ファイルを参照してください。このサンプル ファイルでは、プロセス メトリックを収集するためのメトリック ファミリを定義します。

詳細:

カスタム メトリック ファミリの XML ファイルの作成 (P. 44)

基本プロパティ

TechnologyCertification セクションに表示されるネームスペースおよびスキーマ情報を提供します。

以下の情報は、TechnologyCertification セクションのエレメントの説明です。

Name

メトリック ファミリ名を指定します。 各メトリック ファミリには、システム内部で識別するための一意の名前が必要です。

注: この名前は外部には表示されません。 ユーザ インターフェースにメトリック ファミリ名を表示するには、DisplayName エレメントを使用します。

DisplayName

ユーザ インターフェースに表示するメトリック ファミリ名を指定します。

Description

属性の外部説明を指定します。

CertificationType

メトリック ファミリのタイプを定義します。

注: ユーザ定義のメトリック ファミリでは、CUSTOM のみがサポートされています。

TableName

メトリック ファミリが収集するメトリックを格納するデータベース テーブル名を指定します。

注: データベース テーブルは一意（別のメトリック ファミリが使用しない値）であり、大文字とアンダースコア文字から構成される必要があります。たとえば、PROCESS_STATS などです。

<AttributeList> セクションの追加

<AttributeList> セクションは、メトリック ファミリに対して収集する設定またはパフォーマンスのメトリックを定義します。1 つのデバイス タイプについて多数の属性を使用できる場合でも、インフラストラクチャを監視するためにどの属性が重要かについては慎重に計画してください。 属性データに対する一般的な 3 つの用途は以下のとおりです。

- レポート - これらの属性値は頻繁に変化する可能性があります。通常、これらの値はサイクルごとにポーリングされ、長期にわたる変更を実証するためにレポートで使用される変動値を提供します。 たとえば、バイト数（イン/アウト）、帯域幅、CPU 使用率、メモリ使用率などがあります。
- 設定 - これらの属性は、アイテムを識別するためのデータを提供します。 このデータはめったに変更されません。 そのため、このデータは通常、ディスカバリ中のみ収集されます（サイクルごとにポーリングされません）。 たとえば、名前、説明、および ID などがあります。
- 照合 - これらの属性の目的は、ユーザの環境における設定の変更を検出することです。 検出された変更によって、Data Aggregator データが正確であることを保証するためのディスカバリをトリガすることができます。

各属性には以下のタグを含めることができます。

<Name>

属性の一意の内部名（修飾名形式）です。

<Description>

このテキストは、CA Performance Center ビューにツールヒントとして表示されます。

<AttributeDisplayName>

このタグは、属性がレポートおよび CA Performance Center ダッシュボード内でどのように識別されるかを決定します。

<Type>

属性のデータ型。よく使用されるデータ型は、*Integer*、*Long*、*Double*、*String*、または *ObjectID* です。

<Polled>

True に設定すると、Data Aggregator はポーリング サイクルごとにこのメトリック データを収集します。 False に設定すると、データはアイテムのディスカバリでのみ収集されます。

<IsList>

属性はスカラー データまたはテーブル データのいずれかを提供します。 以下の定義について考えてみます。

- スカラー属性 - これらの属性は一度に 1 つの値を返し、デバイス データと共に格納されます。 たとえば、「現在のプロセスの数」の属性は 1 つの値を返します。
- テーブル属性 -- これらの属性は、テーブル内の各行に対する値を返します。 たとえば、プロセス メトリック ファミリはテーブル属性を使用します。各行は、1 つのプロセス アイテムに対応します。

注: このタグが 1 つの属性に対して True である場合、同じ属性リスト内のすべての属性に対して True に設定される必要があります。この場合、インデックス（*ObjectID* タイプ）および名前（*String* タイプ）属性をリストに定義する必要があります。

<IsDbColumn>

このタグは、属性値がデータベース テーブルに格納されることを示します。 設定メトリックの場合、このタグの値は False に設定します。

<RollupStrategy>

このタグは、各ポーリング値の合計を計算するか、平均を計算するかを指定します。 <Polled> と <IsDbColumn> が True に設定されている場合は、このタグが必要です。

<Baseline>

True に設定すると、この属性に対してベースラインが計算されます。また、<BaselineList> セクションに対応するベースライン定義を追加します。

<RollupStrategy> を含めるか、または <Baseline> を True に設定した場合は、以下のタグも必要です。

Baseline

この属性のベースラインを計算するかどうかを示します。 True に設定した場合、対応する BaselineList 定義が存在する必要があります。

Maximum

ロールアップ中にこの属性の最大を計算するかどうかを示します。データベース テーブルに「max_」列を作成します。

Minimum

ロールアップ中にこの属性の最小を計算するかどうかを示します。データベース テーブルに「min_」列を作成します。

Variance

ロールアップ中にこの属性の差異を計算するかどうかを示します。データベース テーブルに「var_」列を作成します。

StandardDeviation

ロールアップ中にこの属性の標準偏差を計算するかどうかを示します。 データベース テーブルに「std_」列を作成します。

RollupStrategy

個別にポーリングされた値のロールアップ中にすべてのサイクルで実行する操作を指定します。 値は、合計または平均のいずれかを計算するために「合計」または「平均」になります。

<Filterable>

このタグは、監視プロファイルのフィルタリングに属性を使用できることを示します。 フィルタを適用すると、このタグを持った属性はフィルタ式に使用可能になります。

注: それぞれの XML タグの詳細については、MetricFamily.xsd および Component.xsd ファイルで提供されるインライン ドキュメントを参照してください。 コード例については、ProcessInfoMFWithComponent.xml ファイルを参照してください。このサンプル ファイルでは、プロセス メトリックを収集するためのメトリック ファミリを定義します。

詳細:

カスタム メトリック ファミリの XML ファイルの作成 (P. 44)

AttributeList

Attribute List は、収集されてデータベース テーブルに格納された属性をリスト表示します。 以下の情報は、AttributeList セクションで使用されるエレメントの説明です。

Name

DB 列に名前を付けるために使用される一意の内部名を指定します。名前は QName 形式で指定されます。

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

注: この名前は外部には表示されません。 ユーザ インターフェースに属性名を表示するには、AttributeDisplayName エレメントを使用します。

AttributeDisplayName

オペレータと管理者のインターフェースに表示される値を指定します。AttributeDisplayName は複数の言語にローカライズできます。

Description

ユーザ インターフェースに属性の説明を表示します。属性名の上にマウス ポインタを置くと、説明はツールヒントでも表示されます。

Type

この属性のデータ型を示します。 最も使用頻度の高いデータ型は、*Integer*、*Long*、*Double*、*String*、または *ObjectID* です。

Polled

属性がポーリングされるかどうかを示します。 False に設定すると、ディスカバリ中にのみアクセスされます。

IsList

True に設定すると、テーブルから属性を取得します。テーブルの各行に対するコンポーネント アイテムも作成されます。

注: このプロパティを任意の属性で True に設定した場合、データベースに格納されているメトリック ファミリのすべての属性で、このプロパティを True に設定します。

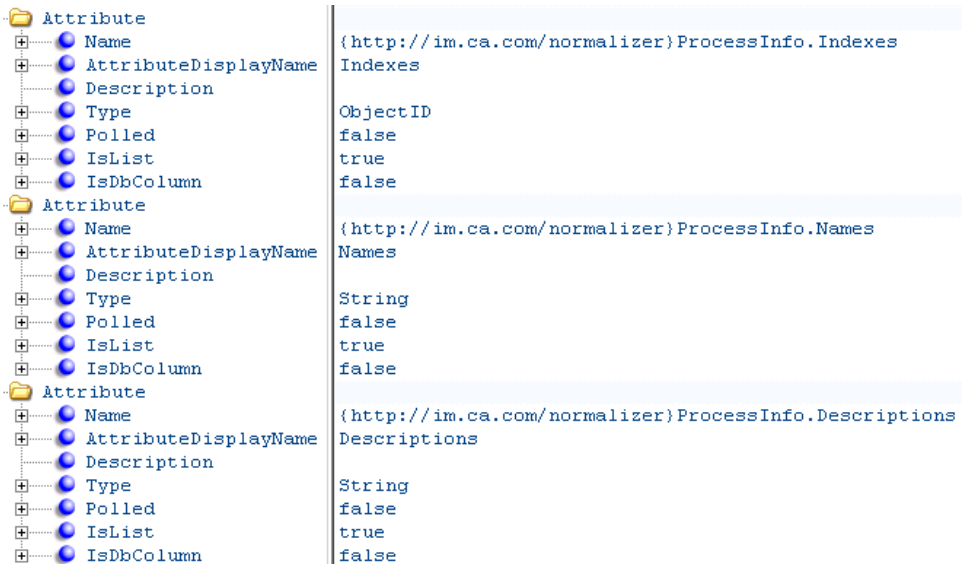
IsDbColumn

データベース テーブルに値を格納します。 IsDbColumn は通常、すべてのサイクルでポーリングされ、レポートに含まれる属性に使用されます。

注: ProcessInfoMFWithComponent.xml を使用せずに、http://host:port/genericWS/metricfamilies/<name> から XML ファイルをダウンロードしてユーザのメトリック ファミリで開始する場合、非推奨の RollupExpression および Units ノードを削除します。

一般属性

以下の図で、Indexes、Names、および Descriptions 属性はすべてのメトリック ファミリに存在します。



サポートされるすべてのベンダー認定は、以下のようにメトリック ファミリによって公開される URI を提供できます。

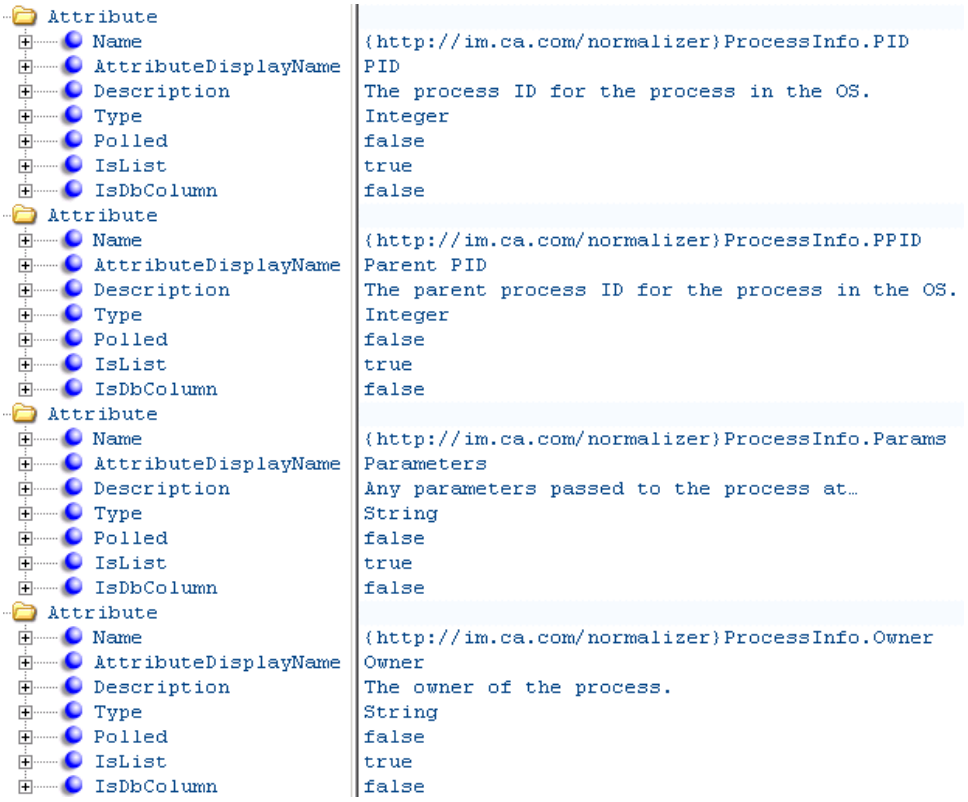
{http://im.ca.com/normalizer}Name.Indexes

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

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

単純属性

以下の図で、ほとんどの属性では値のみがデータベースに格納されます。ベースラインの評価など、それ以上の処理は実行されません。



サポートされるベンダー認定では、以下の形式でいくつかのメトリック ファミリ公開 URI を提供する必要があります。

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

<BaselineList> セクションの追加

環境を分析する場合には、正常な範囲外でアイテムがいつ動作するのかを知っておくことは有用な情報です。 「正常」な状態の特定を容易にするために、Data Aggregator はベースライン値を計算することができます。

注: レポートのベースラインの詳細については、「Data Aggregator 管理者ガイド」を参照してください。

カスタム メトリック ファミリでは、ある属性のベースライン情報を計算するよう Data Aggregator に指示することができます。 カスタム メトリック ファミリ内の <BaselineList> セクションは、ある属性に対するベースライン情報の計算方法を示しています。

重要: <AttributeList> セクション内の属性で <Baseline_> タグが True に設定されている各属性に対して、ベースライン計算をこのセクションに定義する必要があります。

このセクションで、ベースライン計算をセットアップするためのオプションは以下のとおりです。

Name

ベースライン定義の名前を指定します。

ID

ベースライン定義の一意の識別子を指定します。 値は、このメトリック ファミリのすべてのベースライン定義セット内で一意である必要があります。

PerformanceMetric

ベースラインが計算されるメトリックの名前を指定します。 AttributeList の Name を指定します。属性の Polled プロパティ セットを True に設定する必要があります。

StartDate

廃止されました。 存在する場合は、0 に設定します。

EndDate

廃止されました。 存在する場合は、0 に設定します。

DaysOfWeek

廃止されました。 存在する場合は、0 に設定します。

Period

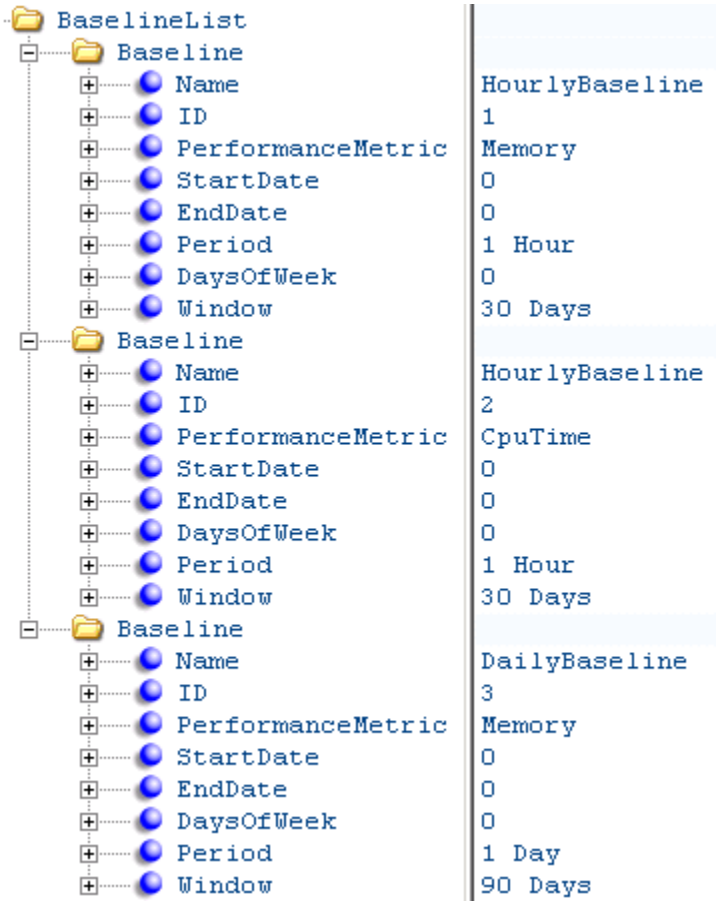
時間単位または日単位でベースラインを計算するように指定します。

Window

廃止されました。 存在する場合は、30 日に設定します。

注: それぞれの XML タグの詳細については、MetricFamily.xsd および Component.xsd ファイルで提供されるインライン ドキュメントを参照してください。 コード例については、ProcessInfoMFWithComponent.xml ファイルを参照してください。このサンプル ファイルでは、プロセス メトリックを収集するためのメトリック ファミリを定義します。

以下の図は、BaselineList エレメントの使用例を示します。



詳細:

カスタム メトリック ファミリの XML ファイルの作成 (P. 44)

<ComponentList> セクションの追加

*コンポーネント*は、デバイスに関連付けられる項目です（たとえば、デバイスは CPU、インターフェース、およびプロセス コンポーネントに関連付けることができます）。 コンポーネント タイプを使用すると、デバイスに関連付けられているアイテムを分類するのに役立ちます。 メトリック ファミリでは、テーブル属性は、関連するコンポーネントのリストが必要です。Data Aggregator に現在は存在していないコンポーネントについては、メトリック ファミリ XML でもこれらの新しいコンポーネントを定義する必要があります。

メトリック ファミリ XML ファイルのコンポーネント情報は、以下の 2 つの目的で機能します。

- ディスカバリ - ディスカバリによってデバイスのアイテムが作成されると、コンポーネント値はそのアイテムのカテゴリ（要素）を識別します。
- 同期 - Data Aggregator と CA Performance Center 間の同期において、コンポーネントの定義はアイテムをマップし、インターフェースでアイテムがどのように表示されるかを決定します。

<ComponentList> セクションでは、カスタム メトリック ファミリに関連付けられているコンポーネント アイテムに対して作成する要素をリスト表示します。 たとえば、プロセス メトリックを収集するメトリック ファミリは、プロセスを実行中のデバイスに関連付けられます。 このセクションでは「プロセス」の要素を以下のように定義できます。

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

この例は、プロセス メトリック ファミリに対して作成されたすべてのアイテムが「プロセス」コンポーネントとして分類されることを示しています。

重要: このセクションにリスト表示された各コンポーネントは、Data Aggregator 内に定義する必要があります。 コンポーネントが Data Aggregator 内にデフォルトで定義されていない場合は、そのコンポーネントを <ComponentDefinitionList> セクション内に定義します。

注: それぞれの XML タグの詳細については、MetricFamily.xsd および Component.xsd ファイルで提供されるインライン ドキュメントを参照してください。 コード例については、ProcessInfoMFWithComponent.xml ファイルを参照してください。このサンプル ファイルでは、プロセス メトリックを収集するためのメトリック ファミリを定義します。

詳細:

カスタム メトリック ファミリの XML ファイルの作成 (P. 44)

<ComponentDefinitionList> セクションの追加

<ComponentDefinitionList> セクションは、コンポーネントに関する詳細を定義します。 詳細には、コンポーネントのデータベースに格納する追加属性の定義に加えて、コンポーネントおよびその属性と CA Performance Center との同期に関する情報が含まれます。

このセクションは、以下の条件が満たされる場合にのみ必要です。

- <IsList_> タグが True に設定されている属性をカスタム メトリック ファミリの XML ファイルで一覧表示する。
- <ComponentList> セクション内のコンポーネントは事前定義済みのコンポーネントではない。

カスタム メトリック ファミリのこのセクションには、以下のタグが含まれます。

<Name>

<Description>

<CertificationType> - **このタグは必ずカスタムに設定されています。**

AttributeList

（オプション）計算され、コンポーネントのデータベースに格納される属性のリストを定義します。 コンポーネント属性を使用してコンポーネントの追加情報（プロセス コンポーネントの引数など）を CA Performance Center で利用できるようにするには、これらの属性を CA Performance Center と同期します。 このセクションには、各属性の以下のタグが含まれます。

Name

QName 形式で一意の内部名を指定します。

Type

この属性のデータ型を指定します。

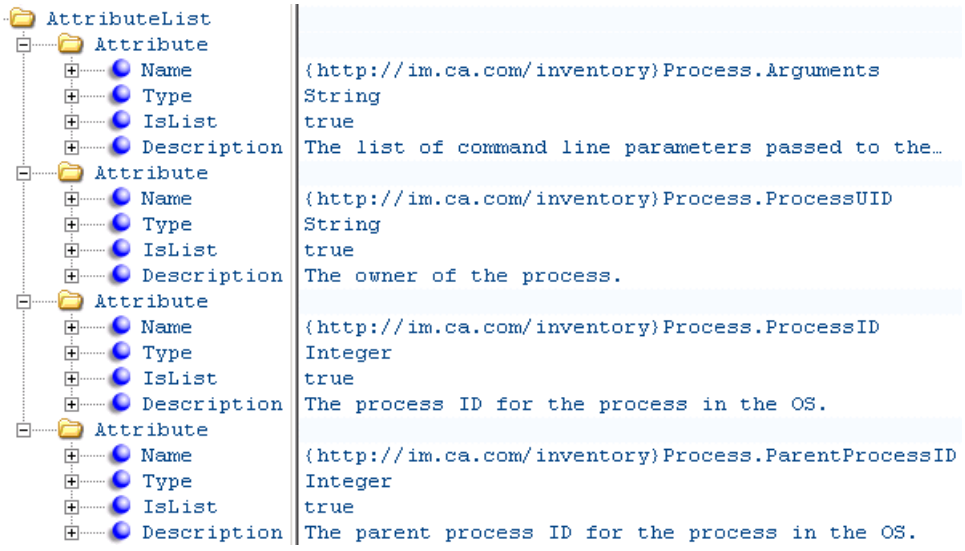
IsList

この属性がスカラーまたはテーブル ベースかどうかを示します。通常、この値はメトリック ファミリで収集される対応する属性の値と等しくなります。

Description

ユーザ インターフェースに表示される、この属性の説明を指定します。

以下の図は、AttributeList の使用例を示します。



ItemSyncDefinition

コンポーネントを CA Performance Center と同期するために必要な詳細を説明します。 ItemSyncDefinition は、同期中にコンポーネント プロパティとして CA Performance Center に送信する追加の属性をリスト表示します。これらの属性は CA Performance Center ユーザ インターフェースで表示できるようになります。 サブタイプを持つアイテムのみが同期されるため、<ItemSyncDefinition> は基本アイテム タイプとサブタイプを定義する必要があります。基本アイテム タイプを使用して論理的なグループを作成し、すべてのサブタイプに利用可能なデフォルト情報および同期動作を提供できます。 一部のユース ケースでは、新しい基本アイテム タイプによりパフォーマンスを向上させることができます。ほとんどのユース ケースでは、「コンポーネント」アイテム タイプとして使用されます。サブタイプ アイテムは、これらのサブタイプに固有の追加プロパティを提供することが可能ですが、これらのプロパティは同期する必要があります。 たとえば、メトリック ファミリは、プロセス サブタイプのプロセス引数プロパティを定義できます。このセクションには、以下のタグが含まれます。

ItemTypeName

CA Performance Center で使用するアイテム ベース タイプの名前を定義します。

ItemSubTypeName

CA Performance Center で使用するアイテム サブタイプの名前を定義します。

ItemTypeLabel

このタイプの単一コンポーネントを表示するときに使用する、ユーザ インターフェースのラベルを指定します。

ItemTypeLabelPlural

このタイプの複数コンポーネントを表示するときに使用する、ユーザ インターフェースのラベルを指定します。

IsDeviceComponent

このアイテムがデバイスのコンポーネントかどうかを指定します（通常は True に設定）。 True に設定された場合、このタグは、このコンポーネントが「デバイス コンポーネント」ビューにリスト表示されることを示します。

GroupBy

CA Performance Center でこのタイプのコンポーネントをグループ化するかどうかを定義します。 True に設定された場合、［インベントリ］メニューの下に新しいメニュー エントリが追加され、これらのコンポーネントがその独自のグループ ビューにリスト表示されるようにします。

Context

このタイプのコンポーネントに対するコンテキスト タイプを CA Performance Center で自動的に作成するかどうかを指定します（通常は False に設定）。

Categorize

このタイプのコンポーネントに対する動的グループを CA Performance Center で作成するかどうかを指定します（通常は False に設定）。

Mapped

複数の Data Aggregator インスタンスが CA Performance Center と同期するときに、コンポーネントが属する Data Aggregator を識別します。

注: このエレメントは現在サポートされていないため、False に設定されます。

ItemPropertyList

アイテム タイプのプロパティとして、CA Performance Center と同期しているすべての属性をリスト表示します。 このセクションには、各属性の以下のタグが含まれます。

Name

このプロパティの CA Performance Center での内部名を定義します。

注: Name タグの最大文字長は 32 文字です。この限度を超えることはできません。

Label

このプロパティを CA Performance Center ユーザ インターフェースで表示するときのラベルを指定します。

AttributeName

CA Performance Center 同期中にこのプロパティを格納するコンポーネント属性を参照します。

Justification

このプロパティを CA Performance Center ユーザ インターフェースに表示するときのテキスト配置を指定します。

デフォルト： 左

DisplayWidth

このプロパティを CA Performance Center ユーザ インターフェースに表示するときに使用する列幅（mm 単位）を定義します。

OrderBySQL

複数の Data Aggregator インスタンスが CA Performance Center と同期するときに、コンポーネントが属する Data Aggregator を識別します。

注: このエレメントは現在サポートされていないため、False に設定されます。

DatabaseType/MaxLength

このプロパティで使用する推定データベース タイプのデフォルト最大長の上書き値を指定します。

注: それぞれの XML タグの詳細については、MetricFamily.xsd および Component.xsd ファイルで提供されるインライン ドキュメントを参照してください。 コード例については、ProcessInfoMFWithComponent.xml ファイルを参照してください。このサンプル ファイルでは、プロセス メトリックを収集するためのメトリック ファミリを定義します。

詳細:

カスタム メトリック ファミリの XML ファイルの作成 (P. 44)

ComponentDefinitionList

カスタム メトリック ファミリで、以下の図に示される *Process* のような新しいコンポーネントを導入する場合、このセクションで定義します。ComponentDefinition は、メトリック ファミリのコンポーネントを作成するためのプロパティをすべて指定します。 プロパティには、コンポーネントのデータベースに格納する追加属性の定義、およびコンポーネントとその属性の CA Performance Center との同期に関する情報が含まれます。

CA Performance Center との同期

Data Aggregator と CA Performance Center を同期すると、コンポーネント アイテムが［インベントリ］メニューに表示されます。 また、同期によって CA Performance Center でそれらのアイテムをグループ化できるようにし、そのアイテム タイプのコンテキスト ページを有効にします。

Data Aggregator は、デバイス、インターフェース、およびさまざまなデバイス コンポーネント タイプを CA Performance Center に同期します。 レポートに使用できるデータを含むインターフェースのみが、CA Performance Center と同期されます。 これらには、アクティブに監視されているインターフェース、および廃棄されたインターフェースがあり、履歴データが含まれます。

Data Aggregator は、1 つ以上の IP アドレスを持つすべてのインターフェースについて、IP アドレスを同期します。 利用可能な場合、Data Aggregator はあるインターフェースの IP アドレスに対するサブネット情報を同期します。

カスタム メトリック ファミリの他のコンポーネントと CA Performance Center を同期するには、<ItemSyncDefinition> セクションを使用できます。このセクションは、コンポーネントのタイプとサブタイプ、およびそれらが CA Performance Center にどのように表示されるかを定義します。

検出されたデバイスおよびコンポーネントは通常、CA Performance Center との同期を開始するまで、最長で 5 分ほどかかります。アイテム数が多い場合、同期が完了するまでの時間が 5 分を超えることがあります。

同期の進行中、Data Aggregator は、現在の同期が完了するまで待機してから、新しく検出されたデバイスとコンポーネントを同期します。

アイテム プロパティを 32 文字を超える <Name_> タグ値と同期する

ItemPropertyList セクションの <Name> タグ値は、32 文字を超えないようにしてください。 ただし、それが同期の問題を引き起こす場合は、アイテムプロパティを、32 文字を超える <Name> タグ値と同期することができます。

以下の手順に従います。

1. メトリック ファミリを削除します。
   a. IMDataAggregator/apache-karaf-2.3.0/deploy ディレクトリを見つけます。
   b. インポートしたメトリック ファミリとベンダー認定用に作成した XML ファイルを削除します。 それらの名前を以下に示します。
      - im.ca.com-normalizer-<technology>.xml
      - im.ca.com-inventory-<technology>.xml
      - im.ca.com-certifications-snmp-<vendor>.xml
2. 以下のコマンドを使用して Data Aggregator を再起動します。

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

   Data Aggregator が再起動されたら、CA Performance Center に以前インポートしたメトリック ファミリとベンダー認定が表示されないことを確認します。 また、そのカスタム認定に関連して以前検出されたコンポーネントがすべて削除されていることを確認します。
3. CA Performance Center で［管理］ - ［データ ソース］をクリックします。
4. ［Data Aggregator］を選択し、［再同期］ボタンをクリックします。

   残っているメトリック ファミリのコンポーネントが Data Aggregator と CA Performance Center の間で同期されます。
5. カスタム メトリック ファミリ XML ファイルを編集し、<Name> タグ内の文字数が 32 文字未満になるようにします。
6. 修正したメトリック ファミリの XML ファイルをインポートします。

（オプション）<ComponentReconciliation> セクションの追加

<ComponentReconciliation> セクション内で、カスタム メトリック ファミリに対して照合アルゴリズムを定義できます。 照合アルゴリズムは、監視対象デバイスの設定変更をサポートするために使用します。 デバイスが、あるメトリック ファミリをどのようにサポートするのかは、その設定と考えることができます。 たとえば、インターフェースの数およびその設定は、デバイスが［インターフェース］メトリック ファミリをサポートする方法を表しています。 デバイスのインターフェースに対する変更が発生した場合、Data Aggregator は監視が最新のものであることを保証するために、これらのインターフェースの表現を更新する必要があります。照合アルゴリズムは、この設定の更新中に適用されます。

特定のメトリック ファミリのサポートでは、個々のコンポーネント アイテムが Data Aggregator で作成されます。 これらのコンポーネント アイテムは、特定のメトリック ファミリをサポートするデバイス設定を表します。 Data Aggregator は、例のように［インターフェース］メトリック ファミリを使用して、ポートのコンポーネント アイテムを作成して、デバイスの各ネットワーク インターフェースを表します。

最初のディスカバリおよびコンポーネント アイテムの作成は、デバイス コレクションを通してデバイスに監視プロファイルが適用されたときに発生します。 監視対象デバイスの変更のサポートが必要であれば、続いてコンポーネントのディスカバリが発生します。 以降のコンポーネント ディスカバリでは、一連のコンポーネント アイテムを更新するのに必要な変更を決定するために、照合アルゴリズムが適用されます。

デバイスの設定が変更された場合には、判断が必要な 4 つのシナリオがあります。

- 新規コンポーネント - このような新規コンポーネント アイテムが Data Aggregator で必要なため、デバイスが変更されました。
- 変更されないコンポーネント - あるデバイスの設定変更によって、既存のいくつかのコンポーネントは変わらなかったため、これらのコンポーネント アイテムは変更されていません。

- 変更されたコンポーネント - デバイス設定が一部のコンポーネントを変更しましたが、未だ存在しています。 Data Aggregator 内のコンポーネント アイテムは、新しい設定を反映するために更新する必要があります。
- 削除されたコンポーネント - あるデバイスは、デバイスに存在していない 1 つ以上のコンポーネントを変更します。 これらのコンポーネント アイテムは削除または廃止できます。

照合アルゴリズムは、既存のコンポーネント アイテムと新規のディスカバリ結果を比較することができる属性のセットを定義します。 属性の値が比較され、既存のコンポーネント アイテムと一致する新規のディスカバリ結果が判断され、また、新規または変更済みのコンポーネントが表示されます。 この比較によって以下の結果が生じます。

- ディスカバリ結果が既存のコンポーネント アイテムと一致しない場合は、新規コンポーネント アイテムが作成されます。
- ディスカバリ結果が既存のコンポーネント アイテムのすべての属性と一致する場合は、コンポーネント アイテムは変更されません。
- ディスカバリ結果の一部が一致し、いくつかの属性値が異なる場合、既存のコンポーネント アイテムは新しい属性値に更新されます。
- ディスカバリの結果により、既存のいくつかのコンポーネント アイテムが新しい結果と一致しない、と判断されることがあります。 この場合、これらのコンポーネントはデバイスに存在しないと判断され、削除また廃棄することができます。

照合アルゴリズムでは、一致の 2 つのタイプとして ExactMatch と BestofMatch を定義できます。

ExactMatch

ディスカバリ結果が、指定したコンポーネント アイテムのすべての属性と一致する必要があることを指します。

例：ExactMatch 照合アルゴリズム

以下の例は、ExactMatch 一致タイプを定義する照合アルゴリズムを示しています。

```
<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>
```

このアルゴリズムは、ディスカバリの結果が既存コンポーネント アイテムの 3 つのすべての属性と一致した場合のみ完全一致になることを示しています。 完全一致にならない場合は、以下のいずれかの状態に応じて結果が生じます。

- 3 つの属性値のいずれかが異なる場合は、新規コンポーネント アイテムが作成されます。
- 3 つの属性値が一致し、その他の属性値が異なる場合は、既存のコンポーネント アイテムが新しい属性値で更新されます。

BestofMatch

既存のコンポーネント アイテムの属性と同じ数だけ一致する必要がある属性の最小数を指定します。 各属性には 1 つの「必須」キーが含まれています。 「必須」キーが［true］に設定されていれば、その属性は一致属性のうちの 1 つです。

例:BestofMatch 照合アルゴリズム

以下の例は、BestofMatch 一致タイプを定義する照合アルゴリズムを示しています。

```
<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>
```

このアルゴリズムは以下の要件を定めています。

- ディスカバリ結果が BestofMatch 一致になるには、一覧表示された属性の少なくとも 2 つと既存のコンポーネント アイテムが一致する必要があります。 BestofMatch 一致にならない場合は、以下の状態に応じて結果が生じます。
  - 2 つ未満の属性が一致した場合、新しいコンポーネント アイテムが作成されます。
  - 少なくとも 2 つの属性値が一致し、その他の属性値が異なる場合は、既存のコンポーネント アイテムが新しい属性値で更新されます。

    注: 一致させる属性値のうちの 1 つは必須属性である必要があります。
- Name 属性の「必須」キーは［true］に設定されます。 これは、一致させる 3 つの属性うちの 1 つが Name 属性である必要があることを意味します。

BestOf 一致アルゴリズムが照合定義に含まれている場合、以下の結果が発生します。

- ディスカバリ結果が既存の 1 つのコンポーネント アイテムにのみ一致する場合、結果は一致です。
- ディスカバリ結果が既存の複数のコンポーネント アイテムに一致する場合、一致する属性の数が考慮されます。 最もよく一致する属性を持った既存のコンポーネントが、ディスカバリ結果に一致すると見なされます。

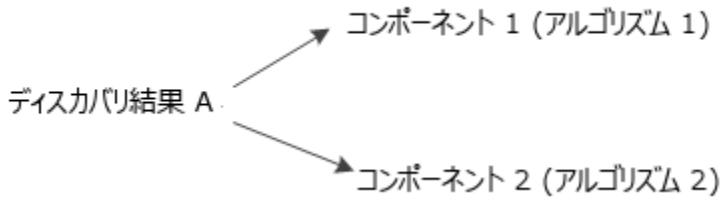
  注: 一致属性の数が複数のコンポーネントで同じ場合、一致するコンポーネント アイテムはランダムで選択されます。結果として、karaf ログは、この条件に関する警告を記録します。

ユーザは複数の一致アルゴリズムを照合定義に追加することができます。複数の一致アルゴリズムが追加された場合は、一致の優先順位があります。MatchAlgorithmList の一番上の一致アルゴリズムの優先順位が最も高くなります。 一番下の一致タイプの優先順位が最も低くなります。

照合定義に*複数*の一致タイプが含まれている場合は、以下の結果が生じます。

- ディスカバリ結果が既存の 1 つのコンポーネント アイテムのみに一致する場合、結果は一致です。 アルゴリズムの優位性は重要ではありません。
- ディスカバリ結果が、既存の複数のコンポーネント アイテムに一致する場合は、最も高い優位性を持つアルゴリズムが優先されます。 最も高い優位性のアルゴリズムに起因するコンポーネント アイテムが一致となります。

  例：



アルゴリズム 1 がより高い優位性を持っているため、ディスカバリ結果 A はコンポーネント 1 一致します。

- 複数のディスカバリ結果が、既存の 1 つのコンポーネント アイテムに一致する場合は、最も高い優位性を持つアルゴリズムが優先されます。最も高い優位性のアルゴリズムの結果として一致したコンポーネント アイテムが一致となります。

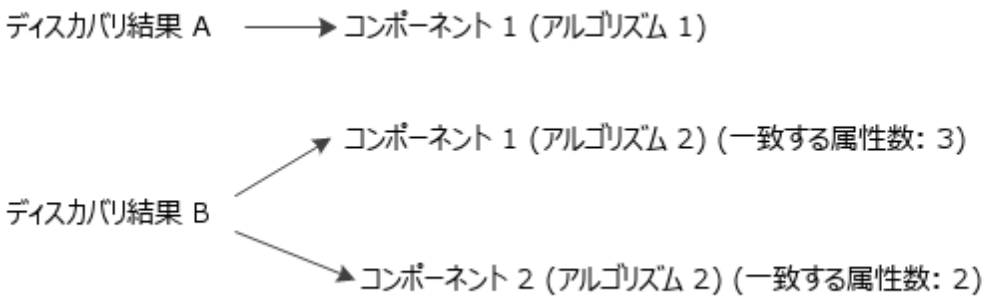
  例：

  ディスカバリ結果 A ⟶ コンポーネント 1 (アルゴリズム 1)

  ディスカバリ結果 B ⟶ コンポーネント 1 (アルゴリズム 2)

  アルゴリズム 1 がより高い優位性を持っているため、ディスカバリ結果 A はコンポーネント 1 に一致します。

  例：



この例では、ディスカバリ結果 B は、コンポーネント 2 よりも多くの属性を持つコンポーネント 1 に一致します。 2.ディスカバリ結果 A はコンポーネント 1 に一致します。これは、より優位性の高いアルゴリズムで一致するためです。 このため、コンポーネント 1 の一致はディスカバリ結果 A となり、ディスカバリ結果 B はコンポーネント 2. 2 に一致します。

注: 照合アルゴリズムを定義していない場合、Data Aggregator は Item.Name 属性を使用してコンポーネントを照合します。

詳細:

カスタム メトリック ファミリの XML ファイルの作成 (P. 44)

（オプション）<ReconfigDetectionAttr> セクションの追加

メトリック ファミリの専用スカラー属性を使用すれば、メトリック ファミリの変更を検出することができます。 この方法は、よりよく機能し、生成されるネットワーク トラフィックが少ないため、<componentReconciliation> セクションを追加するより効率的です。 変更があるかどうかを判断するために、スカラー属性のみがポーリングされます。

メトリック ファミリに変更を検出するスカラー属性がある場合は、<ReconfigDetectionAttr> セクションで、そのスカラー属性を指定できます。

例：[インターフェース]メトリック ファミリにおける変更検出の設定

1. ポート再設定属性を変更についての監視対象にするには、変更検出タグ、<ReconfigDetectionAttr> を指定します。

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

詳細:

カスタム メトリック ファミリの XML ファイルの作成 (P. 44)

<ExpressionGroupList> セクションの追加

式は、メトリック ファミリ内に定義されているかメトリック ファミリに関連付けられているアイテムの属性に対する値を計算する方法を Data Aggregator に指示します。 カスタム メトリック ファミリに含まれている式により、これらの値がレポートおよびダッシュボード ビューの中でどのように表示されるが決定されます。 式は、<ExpressionGroupList> セクションに含まれています。

お使いのメトリック ファミリの属性がテーブル属性である場合、そのメトリック ファミリ XML には <ExpressionGroupList> セクションが含まれている必要があります。また、少なくとも以下の 2 つの関連アイテムに式グループが必要です（宛先認証）。

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

また、カスタム メトリック ファミリの中で定義された各コンポーネントについて、1 つの式グループを定義する必要があります。
<ExpressionGroupList> セクション内のタグは以下のとおりです。

- <ExpressionGroup> - Item、DeviceComponent、およびカスタム メトリック ファミリが監視する各コンポーネント タイプに対して、これらのタグのうちの 1 つを作成します。 このタグには以下のタグが含まれます。
  - <DestCert> - 式が関係する関連アイテムの名前。
  - <ExpressionList> - このタグには、グループ内の属性に対して定義された式が含まれます。 このタグには以下のタグが含まれます。
    - <Expression> - 各式リストの中に、これらのタグのうち少なくとも 1 つが含まれています。 これらのタグは、特定の属性値の算出方法を定義します。 このタグには以下のタグが含まれます。
      – <DestAttr>
      – <Expression>

注: それぞれの XML タグの詳細については、MetricFamily.xsd および Component.xsd ファイルで提供されるインライン ドキュメントを参照してください。 コード例については、ProcessInfoMFWithComponent.xml ファイルを参照してください。このサンプル ファイルでは、プロセス メトリックを収集するためのメトリック ファミリを定義します。

詳細:

カスタム メトリック ファミリの XML ファイルの作成 (P. 44)

ExpressionGroupList

以下の情報は、属性リスト内のエレメントからの計算の実行についての説明です。 ほとんどの計算は、プロセス ID などの簡易割り当てです。

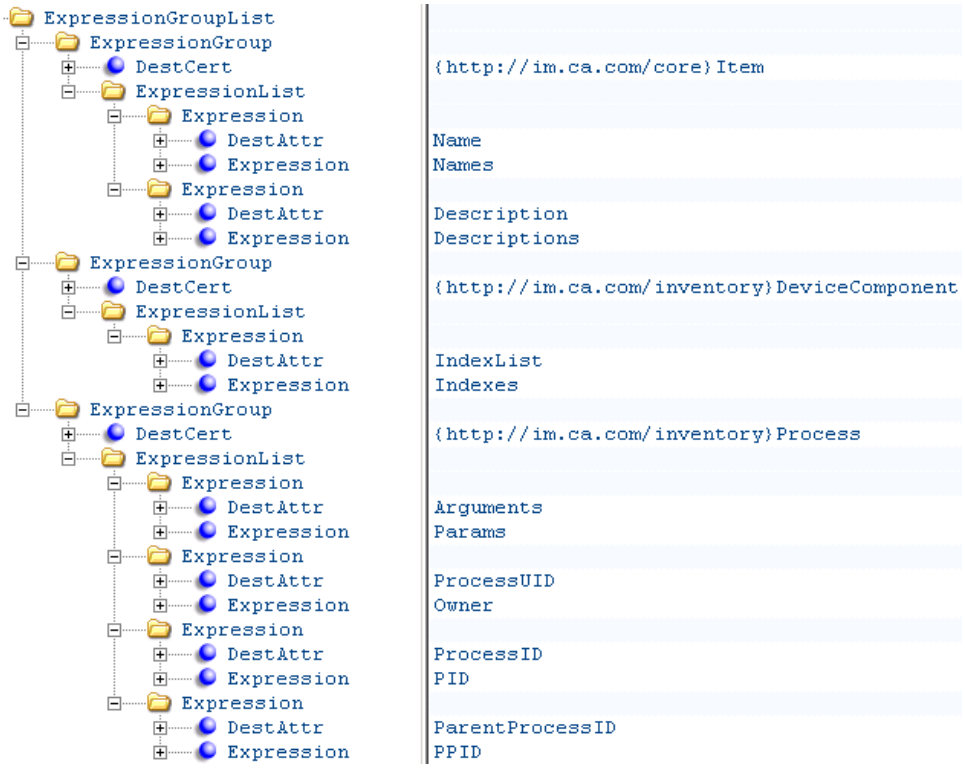
注: メトリック ファミリとベンダー認定の ExpressionGroup を混同しないよう注意してください。メトリック ファミリは、以下の形式の URI によってその属性を公開します。

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

属性は、ベンダー認定の ExpressionGroup で参照されます。

メトリック ファミリの ExpressionGroup は、DestCert および DestAttr URI を使用して指定されるさまざまなアイテムをデータベースに入力します。値は Expression で計算されます。通常、属性リスト内のエレメントの簡易割り当てです。

たとえば、以下の図は、{http://im.ca.com/inventory}Process.ProcessID がメトリック ファミリの AttributeList からの PID 属性にどのようにマップされるかを示します。



以下の DestCert URI が存在する必要があります。

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**	ComponentDefinitionList (P. 65) で定義される属性名。 この例では、**component** プロセスは属性の Arguments、ProcessUID、ProcessID、および ParentProcessID を提供します。

テスト環境でのカスタム メトリック ファミリの結果の確認

カスタム メトリック ファミリ XML を完了して、テスト環境へインポートした後で、結果を確認します。 カスタム メトリック ファミリにより、本番環境で確実に正しい結果が生じるようにするためには、検証は重要なプロセスです。

重要: カスタム メトリック ファミリを作成し、*最初にテスト環境で*必ず確認してください。 カスタム メトリック ファミリを作成するには、メトリック ファミリの XML ファイルを手動で編集する必要があります。 この XML ファイルでのセマンティック エラーは、予測不能の結果を引き起こす場合があります。

以下の手順に従います。

1. テスト システムでカスタム メトリック ファミリを使用して、ベンダー認定を作成します。 ベンダー認定を作成する際に、以下の結果を検証します。
   a. カスタム メトリック ファミリ XML ファイルは、ベンダー認定ウィザードで表示されます。
   b. カスタム メトリック ファミリを選択すると、正しいメトリック ファミリの属性が表示されます。
2. ベンダー認定により、ディスカバリ中に適切なアイテムが作成されることを確認します。
3. アイテムが、メトリック ファミリ XML ファイルで指定されたメトリック データをポーリングし、収集することを確認します。

4. 収集されたデータが正しいかどうかを検証します。
5. アイテム同期が設定されている場合は、アイテム データが CA Performance Center と正しく同期されていることを確認します。
6. ディスカバリをもう一度実行し、情報が正しく更新されていることを確認します。

詳細:

カスタム メトリック ファミリを作成する方法 (P. 41)

カスタム メトリック ファミリの XML ファイルのインポート

カスタム メトリック ファミリ XML を作成し、それをテスト環境で確認した後、Data Aggregator インストール環境にインポートします。

以下の手順に従います。

1. Data Aggregator データ ソースの［監視設定］メニューから［メトリック ファミリ］をクリックします。

   ファクトリおよびカスタムのメトリック ファミリも含めて、メトリック ファミリのリストが表示されます。 ファクトリ メトリック ファミリには鍵の記号が表示されます。
2. ［インポート］をクリックします。
3. カスタム メトリック ファミリ XML ファイルを参照して選択し、［開く］をクリックして、次に［インポート］をクリックします。

   カスタム メトリック ファミリがインポートされます。

   重要: データの損失を回避するには、ベンダー認定、メトリック ファミリ、またはコンポーネントを作成または更新するたびに、展開ディレクトリを必ずバックアップします。

詳細:

カスタム メトリック ファミリを作成する方法 (P. 41)

廃止されたコンポーネントの削除の自動化

管理者は、廃止されたコンポーネントがネットワークから自動で削除されるようにできます。 プロセスを自動化するスクリプトを書く前に、廃止されたコンポーネントを削除するために Data Aggregator に含まれているスクリプトを使用する方法を理解します。 たとえば、週単位の cron ジョブをセットアップし、1 か月経過した廃止コンポーネントを削除できます。

Data Aggregator に含まれている remove_retired_items スクリプトは 2 つの部分で構成されます。 スクリプトの最初の部分は、設定したフィルタに基づいて、廃止されたコンポーネントに関するデータを識別して返します。スクリプトの 2 番目の部分は、廃止されたコンポーネント リストの削除を実行します。 プロセスを自動化するには、このスクリプトがどのように構築されているかを理解します。

注: remove_retired_items スクリプトの使用に関する詳細については、「Data Aggregator 管理者ガイド」を参照してください。

例: 廃止されたコンポーネントのリストをデバイス IP アドレスでフィルタリング

この例では、10.252.1.1 のプライマリ IP アドレスを持つデバイスについて廃止されたコンポーネントをすべて検索します。 IP アドレスによる直接のコンポーネント フィルタが可能ではないので、IP アドレスによるフィルタリングは 2 段階のプロセスとなります。廃止されたコンポーネントをフィルタするには、まずコンポーネントが関連付けられているデバイスの IP アドレスをメモします。IP アドレス情報で、デバイスに対するデバイス アイテム ID を判断します。 次に、そのデバイス アイテム ID を使用して、廃止されたコンポーネントを判断します。 最後に、廃止されたコンポーネントを削除します。

注: この例は curl コマンドを使用しますが、慣れているコマンドを使用してもかまいません。

1. filterDeviceIP.xml ファイルを作成します。 このファイルを使用して、10.252.1.1 のプライマリ IP アドレスがあるデバイスに対するデバイス アイテム ID を返します。 ファイルは以下の例のようになります。

```
<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. 以下のコマンドを実行します。

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

   -X

   対象のフィルタを作成します。

   hostname:*port*

   Data Aggregator のホスト名およびポート番号を指定します。

   デフォルト ポート： 8581

   -H

   ポストしているファイルのコンテンツ タイプを示します。

   -T

   ポストしているファイルを示します。

以下の結果が HTTP レスポンスとして返されます。

```
<?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>
```

デバイス アイテム ID 107881 が返されます。 結果にはデバイスに関する詳細情報も表示されます。

3. filterRetired.xml ファイルを作成します。 このファイルを使用して、デバイス アイテム ID が 107881 であるデバイスと関連付けられている廃止されたコンポーネントを返します。 このファイルは以下の例のようになるはずです。

```
<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. 以下のコマンドを実行します。

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

   以下の結果が HTTP レスポンスとして返されます。

```
<?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>
```

   フィルタ条件に適合する廃止された 2 つのコンポーネントが返されます。 コンポーネントのアイテム ID は 128452 および 128451 です。

5. deleteRetiredList.xml ファイルを作成します。 このファイルを使用して、廃止されたコンポーネントの返されたリストを削除します。 ファイルは以下の例のようになります。

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

6. 以下のコマンドを実行します。

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

   以下の結果が HTTP レスポンスとして返されます。

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

   廃止されたコンポーネントは正常に削除されました。

第 6 章: トラブルシューティング

このセクションには、以下のトピックが含まれています。

エラー メッセージに関する詳細の表示 (P. 83)
トラブルシューティング： メトリック ファミリが不完全 (P. 83)
トラブルシューティング： メトリック ファミリがサポートされていません (P. 85)

エラー メッセージに関する詳細の表示

症状：

Web サービスが失敗したとき、暗号化されたエラー メッセージを受信しました。

解決方法：

この問題のデバッグに役立つ詳細については、IMDataAggregator/apache-karaf-2.3.0/data/log ディレクトリ内の karaf.log ファイルを確認してください。 Web サービスがリクエスト処理に失敗すると必ず、スタックトレースおよびより詳細なエラー メッセージがログ ファイルに出力されます。 ログ ファイルには、通常その問題を引き起こした特定のタグが含まれます。

トラブルシューティング : メトリック ファミリが不完全

症状:

カスタム メトリック ファミリを正常にインポートしましたが、その後、不完全なメトリック定義が見つかりました。 たとえば、<Name> プロパティの最大長は 32 文字です。 この制限を超過すると、同期の問題が発生する場合があります。

解決方法:

以下の手順に従い、カスタム メトリック ファミリを慎重に削除します。

1. IMDataAggregator/apache-karaf-2.3.0/deploy ディレクトリを見つけます。
2. メトリック ファミリに対して作成および展開された XML ファイルを削除します。 それらの名前を以下に示します。
   - im.ca.com-normalizer-<technology>.xml
   - im.ca.com-inventory-<technology>.xml

   該当する場合、ベンダー認定に対して作成されたファイルも削除します。
   - im.ca.com-certifications-snmp-<vendor>.xml
3. 以下のコマンドを実行して、Data Aggregator を再起動します。

   ```
   service dadaemon restart
   ```

   Data Aggregator が再起動されたら、CA Performance Center に以前インポートしたメトリック ファミリとベンダー認定が表示されないことを確認します。 また、そのカスタム認定に関連して以前検出されたコンポーネントがすべて削除されていることを確認します。
4. CA Performance Center で［管理］ - ［データ ソース］をクリックします。
5. ［Data Aggregator］を選択し、［再同期］ボタンをクリックします。

   残っているメトリック ファミリのコンポーネントが Data Aggregator と CA Performance Center 間で同期されます。
6. カスタム メトリック ファミリの XML ファイルを編集および修正します。
7. 修正したメトリック ファミリの XML ファイルをインポートします。

トラブルシューティング : メトリック ファミリがサポートされていません

症状:

デバイスのコレクションでメトリック ファミリをポーリングするために、監視プロファイルを作成しました。しかし、［ポーリングされるメトリック ファミリ］テーブルで、これらのメトリック ファミリの 1 つに「サポートされていません」というステータスが表示されます。

解決方法:

問題を解決するには、以下の手順に従います。

1. ポーリングされたデバイスが SNMP クエリに応答することを確認します。
2. サポートされていないメトリック ファミリに移動して、それをクリックします。
3. ベンダー認定でメトリック ファミリがサポートされていることを確認します。 ベンダー認定が定義されていない場合は、カスタム ベンダー認定を作成します。
4. デバイスで、すべてのキー ベンダー認定属性がサポートされることを確認します。すべてのキー ベンダー認定属性がサポートされている場合は、デバイスに移動し、カスタム ベンダー認定を追加したメトリック ファミリを選択し、［メトリック ファミリの更新］をクリックします。

   デバイス設定が更新されます。