CHAPTER 2

レポートの記述

Prime Performance Manager のレポートを記述するには、レポートの記述手順を慎重に計画し、正しい順序で実行する必要があります。この章では、パッケージ化されたシステム レポートである cpu.xml を修正するする手順について説明します。これは、独自のレポートの開発、テスト、デバッグに役立ちます。

この章は、次の項で構成されています。

一般的な推奨事項

新しいレポートを作成する前に、次の推奨事項を確認してください。

- 基本レポート構造が準備できるまで、すべてのサーバからの最初の新しいレポート XML ファイルの開発をオフラインで実行します。その後、初期テストおよびデバッグのためにステージング サーバに新しいレポートを導入します。
- ライブ システム上でのレポートの作成は避けます。新しいレポートを作成するためにステージング システムを使用します。正しく動作していることが確認されるまで本番システムに新しいレポートを導入しないでください。

レポートを記述する手順の概要

1. 既存のレポートを */opt/CSCOppm-gw/etc/pollers/system* ディレクトリから */opt/CSCOppm-gw/etc/pollers/user* ディレクトリにコピーします。
   - 必要に応じて、レポートの UserCapability.xml ファイルを修正します。

- 新しい MIB を使用する場合は、その MIB をコンパイルします。

2. レポートのプロパティ ファイルをコーディングします。
3. レポートをコーディングします。
4. システム GUI を使用してレポートをイネーブルにします。
5. レポートをテストし、デバッグします。

必要な MIB の決定

Cisco Prime Performance Manager は 600 を超えるシスコおよび業界標準 MIB とともに提供されます。

レポートに必要な MIB が使用可能かどうかを調べるには、次の手順を実行します。

ステップ 1　ゲートウェイ上の MIB のリストを確認します。

- コンパイル済み MIB のリストを表示するには、ヘルプ メニューから、[Reports] を選択し、次に [SNMP MIBs] を選択します。
- すべてのコンパイル済み MIB は、ゲートウェイの */opt/CSCOppm-gw/etc/mibs* ディレクトリにあります。

ステップ 2　MIB を追加する必要がある場合は、次の手順を実行します。

a. MIB をゲートウェイ サーバにコピーします。

b. MIB をコンパイルします。

MIB をコンパイルする手順については、「MIB のコンパイル」（P.2-2）を参照してください。

ステップ 3　事前にコーディングされているレポートの 1 つを修正する場合は、*SystemCapability.xml* ファイル内にそのレポートを参照するエントリがあることを確認します。そのファイルのシステム機能設定を変更する必要がある場合は、*UserCapability.xml* ファイル内で変更を行います。

（注）　*SytemCapability.xml* ファイルは修正しないでください。

MIB のコンパイル

新しい MIB（Cisco Prime Performance Manager の配布に含まれていない MIB）を必要とするレポートを作成する場合は、その MIB をレポートに使用する前にコンパイルしておく必要があります。

MIB をコンパイルするには、次の手順を実行します。

ステップ 1　root ユーザになって、MIB を /opt/CSCOppm-gw/etc/mibs ディレクトリにコピーします。

ステップ 2　次のコマンドを入力して、ユニットとゲートウェイの間のファイル同期を停止します。

```
# /opt/CSCOppm-gw/bin/ppm syncunits disable
```

ステップ 3　次のコマンドを入力して、システム内の MIB をコンパイルします。

```
# /opt/CSCOppm-gw/bin/ppm compilemibs
```

compilemibs コマンドにより、*snmpinfo.dat* ファイルが作成されます。

ステップ 4　MIB 内で検出されたエラーを修正します。エラーがなければ、コンパイル済み MIB（*snmpinfo.dat*）をリロードします。

ステップ 5　次のコマンドを入力して、ユニットとゲートウェイの間のファイル同期を再開します。

```
# /opt/CSCOppm-gw/bin/ppm syncunits enable
```

これでその MIB を参照するレポートを作成できるようになりました。

レポートする統計情報の決定

モニタリング対象のデバイスに対してレポートする Key Performance Indicator（KPI）を決定するには、次の手順を実行します。

ステップ 1　コンパイル済みの MIB を表示します。

ステップ 2　レポーティング要件に従って、ポーリングする MIB 変数を決定します。

ステップ 3　レポートする KPI が既存のレポートでカバーされていないかどうかを確認します。

レポート XML ファイルとプロパティ ファイルの作成

作成する XML ファイルには、次のものを使用できます。

- 既存の Prime Performance Manager レポートのコピーをベースにした新しいファイル。
- 自分のインストレーションにすでに存在するユーザ定義レポートの修正。

警告　**/opt/CSCOppm-gw/etc/pollers/system ディレクトリ内のレポート ファイルとプロパティ ファイルを修正してはなりません。**

レポート ファイルを作成するには、次の手順を実行します。

ステップ 1　あらかじめパッケージに組み込まれている既存のレポートを修正する場合は、そのレポートの *.xml* ファイルとそれに関連した *.properties* ファイルを */opt/CSCOppm-gw/etc/pollers/system* ディレクトリから */opt/CSCOppm-gw/etc/pollers/user* ディレクトリまたは自分で選択した作業ディレクトリにコピーします。

ステップ 2　コピーしたファイルの名前を変更します。*/opt/CSCOppm-gw/etc/pollers/system* ディレクトリ内にあるファイルと重複しない一意のファイル名にする必要があります。

ステップ 3　ユーザが作成した、*/user* ディレクトリ内の既存のファイルを使用する場合は、そのファイルとそれに関連した *.properties* ファイルをコピーし、それらの名前を変更します。

（注）　レポート ファイルが ../system および ../user ディレクトリに存在する場合は、/user のレポート ファイルが優先されます。

レポートのコーディング

これまでの作業により、レポートをコーディングする準備が整いました。以降の項では、cpu.xml ファイルについて一通り説明します。このファイルは、既存の Prime Performance Manager レポート ファイルの 1 つであり、レポート記述の「Hello World」の例として利用できます。関連ファイルである cpu.properties ファイルについても併せて説明します。

- cpu.xml ファイルは、標準の Cisco MIB である CISCO-PROCESS-MIB.my のポーリングを設定し、アクティブなシステム プロセスについてレポートします。また、Cisco エンティティ MIB（CISCO-ENTITY-MIB.my）も使用します。
- cpu.properties ファイルは、1 つの目的のためだけに使用されます。具体的には、システムが CPU レポートを表示する際に使用する変数およびテキスト文字列を定義します。

始める前に、次の点に注意してください。

- ここで示す内容はチュートリアルなので、基本的なこと、つまり、レポート内で使用する XML 要素の一通りの説明から始めます。
- 一部の高度な内容（Prime Performance Manager ゲートウェイ上で提供される *SystemCapability.xml* ファイルをベースにした UserCapability.xml ファイルのコーディングなど）は、このガイドの後の項で詳細に説明します。
- この章の後半には、一般的なレポート記述タスクに関連したリファレンス情報を説明している項があります（一般的なレポート記述タスクについては、「レポートのコーディング」（P.2-4）を参照）。これらのタスクの一部は、チュートリアルでも説明されています。

ヒント　このサンプル レポートの作業中は、cpu.xml ファイルのオンライン レポート ヘルプをブラウザで開いておくと、時間の節約になります。cpu.xml のヘルプ ページを表示するには、Cisco Prime Performance Manager のレポート ツリーにアクセスし、[CPU] を選択して CPU レポートを起動し、その後レポート ヘルプ ツールをクリックします。

例 2-1 に、cpu.xml ファイルを示します。

例 2-1　cpu.xml ファイル

```
<?xml version="1.0"?>
<!-- Copyright (c) 2011 Cisco Systems, Inc. All rights reserved. -->

<!-- MIBS Used

     CISCO-PROCESS-MIB.my
     ENTITY-MIB.my

-->

<ns:PollerList
    xmlns:ns="http://cisco.com/ppm/poller">

    <Poll name="CPU" reportId="CPU">

    <Criteria>CISCO_PROCESS_MIB</Criteria>

    <PollDefinition>

      cpmCPUTotalTable = poll("cpmCPUTotalIndex,
                               cpmCPUTotalPhysicalIndex,
```

```
                                cpmCPUTotal5minRev,
                                cpmCPUTotal1minRev");

      cpmCPUThresholdTable = poll("cpmCPUTotalIndex,
                                   cpmCPUThresholdClass,
                                   cpmCPURisingThresholdValue,
                                   cpmCPUFallingThresholdValue");

      cpmCPUThresholdTable =
         cpmCPUThresholdTable.filter(cpmCPUThresholdClass == 1);

      cpmCPUTotalTable =
        cpmCPUTotalTable.leftJoin(cpmCPUThresholdTable,
   (cpmCPUTotalTable.cpmCPUTotalIndex == cpmCPUThresholdTable.cpmCPUTotalIndex));

    </PollDefinition>

   <ProcessPollResult>

      setCpuInfo();

      fiveMinUtil       = cpmCPUTotal5minRev / 100;
      oneMinUtil        = cpmCPUTotal1minRev / 100;
      risingThreshold   = cpmCPURisingThresholdValue / 100;
      fallingThreshold  = cpmCPUFallingThresholdValue / 100;

   </ProcessPollResult>

   <ProcessDBSummary name="CPU" baseTableName="CPU">

      <Var name="CPUSlot"     type="Integer" key="true">cpuSlot</Var>
      <Var name="CPUNum"      type="Integer" key="true">cpuNum</Var>
      <Var name="CPUDescr"    type="String" key="true">cpuDescr</Var>

      <Var name="CPUUtilMax5min" type="Double" operation="Max">fiveMinUtil</Var>
      <Var name="CPUUtilAvg5min" type="Double" operation="Avg">fiveMinUtil</Var>

      <Var name="CPUUtilMax1min" type="Double" operation="Max">oneMinUtil</Var>
      <Var name="CPUUtilAvg1min" type="Double" operation="Avg">oneMinUtil</Var>

      <Var name="CPURisingThreshold" type="Double">risingThreshold</Var>
      <Var name="CPUFallingThreshold" type="Double">fallingThreshold</Var>

    </ProcessDBSummary>

  </Poll>

  <CSV name="CPU" location="gateway" listen="CPU">
    <Column name="Slot">CPUSlot</Column>
    <Column name="Number">CPUNum</Column>
    <Column name="Description">CPUDescr</Column>

    <Util   name="CPUUtilMax5min">CPUUtilMax5min</Util>
    <Util   name="CPUUtilAvg5min">CPUUtilAvg5min</Util>
    <Util   name="CPUUtilMax1min">CPUUtilMax1min</Util>
    <Util   name="CPUUtilAvg1min">CPUUtilAvg1min</Util>

    <Column name="CPURisingThreshold">CPURisingThreshold</Column>
    <Column name="CPUFallingThreshold">CPUFallingThreshold</Column>
  </CSV>

<!-- *** CPU Utilization *** -->

  <WebReport name="wrnCPUUtil"
```

```
            category="level1Resources,level2CPU"
            reportId="CPU"
            context="Network,Node,CPUSlot,CPUNum,CPUDescr"
            textProps="cpu"
            sortWeight="2">

    <Criteria>CISCO_PROCESS_MIB</Criteria>

    <GraphView>
      <GraphSummary title="gstCPUUtil" />
      <Graph title="gtCPUUtilAvg" >
        <Util name="genCPUUtilAvg">CPUUtilAvg5min</Util>
      </Graph>
      <Graph title="gtCPUUtilPeak" >
        <Util name="genCPUUtilPeak">CPUUtilMax5min</Util>
      </Graph>
      <LeafGraph title="gtCPUUtil" >
        <Util name="genCPUUtilAvg">CPUUtilAvg5min</Util>
        <Util name="genCPUUtilPeak">CPUUtilMax5min</Util>
      </LeafGraph>
    </GraphView>

    <TableView baseTable="CPU">
      <IdLabel/>
      <Label colSpan="2" name="cpuUtil5Min" />
      <Label colSpan="2" name="cpuUtil1Min" />
      <Label colSpan="2" name="cpuUtilThresh" />
      <HeaderRow/>

      <Link name="node" context="Node">fqdnid</Link>
      <Link name="slot" context="CPUSlot">CPUSlot</Link>
      <Link name="cpu"  context="CPUNum">CPUNum</Link>
      <Link name="cpuDes"  context="CPUDescr">CPUDescr</Link>
      <Time/>


      <Util default="true" name="avg">    CPUUtilAvg5min</Util>
      <Util                name="peak">   CPUUtilMax5min</Util>

      <Util                name="avg">    CPUUtilAvg1min</Util>
      <Util                name="peak">   CPUUtilMax1min</Util>

      <Util                name="rising"> CPURisingThreshold</Util>
      <Util                name="falling">CPUFallingThreshold</Util>
    </TableView>

  </WebReport>

</ns:PollerList>
```

レポート XML の主要要素

cpu.xml レポートには、7 つの主要セクションがあります。

- **コメント**：cpu.xml ファイルの先頭には、ポーリングされる MIB を示すコメントが記述されます。なお、コメントは、XML ファイル内の任意の場所に配置できます。

すべてのレポート セクションは、PollerList と呼ばれる 1 つの XML セクション内に指定されます。PollerList セクションの最初の行は、「http://cisco.com/ppm/poller」を指定します。サンプル レポートで指定されているように、必ずこれを含めます。Poll セクションは、複数指定できます。

Poll セクションは、XML ファイルの主要要素であり、4 つのセクションをカプセル化します。

Poll は、*EventPoller.xsd* スキーマ ファイル内で定義される要素を含む複合要素タイプで、次の Processor 要素内にあります。

- **Criteria セクション**：*SystemCapability.xml* ファイルまたは *UserCapability.xml* ファイルに定義されている機能値を指定します。

  「MIB 基準の指定（Criteria セクション）」（P.2-8）を参照してください。

- **PollDefinition セクション**：ポーリング結果を保管するための仮想テーブルを作成するレポート マクロを呼び出します。

  「ポール名とレポート名の指定（Poll 要素）」（P.2-8）を参照してください。

- **ProcessPollResult セクション**：レポーティング マクロを呼び出し、計算式または他の方法を使用してポーリング データを修正または整形する操作を実行します。

  「ポール処理結果の指定（ProcessPollResult セクション）」（P.2-10）を参照してください。

- **ProcessDBSummary セクション**：レポート用のポーリングされたデータを保持する仮想テーブルに対し、テーブルと行の形式をセットアップおよび操作します。定義された処理間隔が終了するたびに、システムがその処理間隔のテーブル データをゲートウェイ上の物理データベースに書き込みます。

  「データベース スキーマへのデータの割り当て（ProcessDBSummary セクション）」（P.2-11）を参照してください。

最後の 2 つのセクションは、ユーザ レポートの外観と形式を指定します。

- **CSV**：opt/CSCOppm-gw/reports ディレクトリ内のファイルを表示する際または各 Web レポート上の CSV オプションへのエクスポートを使用する際に作成する CSV ファイルのレイアウトを指定します。

  「CSV 出力ファイルと形式の指定（CSV セクション）」（P.2-12）を参照してください。

- **WebReport**：ユーザがグラフ ビュー オプションを選択したときに表示されるグラフィカル レポートの属性を指定します。また、Web レポートではレポートに表示するものや表示方法も指定します。WebReport セクションには次が含まれます。

  - **属性**：Prime Performance Manager のメニュー内で使用されるこのレポートの名前、レポート ツリー内でこのレポートが表示される場所、このレポート データの取得元、この Web レポートのプロパティ ファイルなどを指定します。

    「WebReport セクションの属性の指定」（P.2-13）を参照してください。

  - **GraphView**：ユーザがグラフ ビュー オプションを選択したときに表示されるグラフ ビューの属性を指定します。

    「GraphView セクションのコーディング」（P.2-15）を参照してください。

  - **TableView**：ユーザがテーブル ビュー オプションを選択したときに表示されるテーブル ビューの属性を指定します。

    「TableView セクションのコーディング」（P.2-17）を参照してください。

（注） また、Web レポートは Prime Performance Manager グラフィカル ユーザ インターフェイスを使用しても作成および編集できます。

レポート マクロ

Prime Performance Manager ゲートウェイと一緒に提供されるレポート マクロを自分の XML コード内で使用できます。一部のレポート マクロは、XML の特定のセクション内でのみ使用できます。レポート マクロのリファレンス情報については、第 9 章「レポート マクロ リファレンス」を参照してください。

レポートに使用される MIB を示すコメントの追加

レポートの先頭に、レポート用にポーリングされる MIB を示したコメントを含めます。cpu.xml ファイルでは、次のコメントが含まれています。

```
<!-- MIBS Used

     CISCO-PROCESS-MIB.my
     ENTITY-MIB.my

-->
```

使用される MIB の拡張子は .my または .mib サフィックスである必要があります。

MIB 基準の指定（Criteria セクション）

このセクションでは、レポート コーディングにおける高度な内容が処理されます。レポート用に MIB を処理する際に使用される基準を指定します。

SystemCapability.xml により、Prime Performance Manager レポートのデフォルトの MIB 処理基準が指定されます。*SystemCapability.xml* ファイルは編集しないでください。MIB 処理基準を修正する必要がある場合は、UserCapability.xml ファイルを編集し、そこに独自の処理基準を指定します。

新しい機能または条件を追加するには、次のようなエントリを UserCapability.xml に追加します。

```
CISCO_PROCESS_MIB = isTableNotEmpty("cpmCPUTotalTable");
```

このエントリは、デバイス上の MIB テーブル cpmCPUTotalTable にデータがあるかを確認し、ある場合は CISCO_PROCESS_MIB を設定します。または、次を追加できます。

```
UDP_MIB  = hasVar("udpInDatagrams");
```

このエントリは、デバイス上に MIB 変数の udpInDatagrams が存在する場合に UDP_MIB 機能を設定します。これで使用可能なマクロは isTableNotEmpty() および hasVar() です。詳細については、第 9 章「レポート マクロ リファレンス」を参照してください。

ポール名とレポート名の指定（Poll 要素）

Poll 要素：

- ポーラー名およびレポート識別名を指定します 同一ポールにあって異なるカテゴリに分類されたレポートに対して、次の例のように複数のレポート ID を使用できます。

  ```
  name=Interface
  reportId=INTERFACE,INTERFACEINTRA,INTERFACEINTER,INTERFACEERRORS
  ```

ポーラーとレポート ID には一意の名前を割り当てる必要があります。また、同じポーリング セクションの Web レポート セクションで異なるレベルのレポート階層を割り当てる場合は、一義的なレポート ID を割り当てます。

（注）　name と reportId の値は大文字と小文字が区別されます。

たとえば、cpu.xml ファイルを /user ディレクトリ内の別の XML ファイルにコピーする場合は、*/opt/CSCOppm-gw/etc/pollers/system* ディレクトリ内や */opt/CSCOppm-gw/etc/pollers/user* ディレクトリ内で重複しない新しい名前をポーラーとレポート ID に与えます。

- ポーリングおよびポーリング結果の処理をセットアップする下位の主要 XML セクションが含まれる複合 XML 要素です。

ポーリングする MIB 変数の指定（PollDefinition セクション）

SNMP（poll）、CSV（csvPoll）、または CLI（xmlPoll）を使用してポーリングするには PollDefinition セクションが使用できます。SNMP に対して、MIB パラメータは、このセクションで指定します。このセクションではレポートに対して実行されるポーリングを定義し、Prime Performance Manager でレポートの生成に使用する内部データベース テーブルにポーリングされたデータを割り当てます。

例 2-2 に、cpu.xml レポートの例での PollDefinition セクションを示します。

例 2-2　　PollDefinition セクション

```
<PollDefinition>

      cpmCPUTotalTable = poll("cpmCPUTotalIndex,
                              cpmCPUTotalPhysicalIndex,
                              cpmCPUTotal5minRev,
                              cpmCPUTotal1minRev");

      cpmCPUThresholdTable = poll("cpmCPUTotalIndex,
                                   cpmCPUThresholdClass,
                                   cpmCPURisingThresholdValue,
                                   cpmCPUFallingThresholdValue");

      cpmCPUThresholdTable =
         cpmCPUThresholdTable.filter(cpmCPUThresholdClass == 1);

      cpmCPUTotalTable =
        cpmCPUTotalTable.leftJoin(cpmCPUThresholdTable,
  (cpmCPUTotalTable.cpmCPUTotalIndex == cpmCPUThresholdTable.cpmCPUTotalIndex));

    </PollDefinition>
```

このセクションでは、*CISCO-PROCESS-MIB.my* の特定の変数がポーリングされ、それらが 2 つのテーブルに保管されます。

- cpmCPUTotalTable
- cpmCPUThresholdTable

これらの MIB オブジェクトは、CPU ロード モニタリングを提供する CISCO-PROCESS-MIB 内の標準オブジェクトです。その他の情報およびコメントについては、*CISCO-PROCESS-MIB.my* ファイルを参照してください。

POLL マクロ

実際のポーリングは、POLL マクロを使用して行われます。このマクロの構文は次のとおりです。

POLL(*arg1*, *arg2 ... argn*)

ここで、引数はポーリングされる MIB 変数のリストであり、引用符で囲まれます。

FILTER マクロ

次に、PollDefinition セクションは、PollDefinition オブジェクトに対して定義されるマクロ、FILTER マクロを呼び出します。

```
cpmCPUThresholdTable =
         cpmCPUThresholdTable.filter(cpmCPUThresholdClass == 1);
```

FILTER マクロの構文は次のとおりです。

FILTER(*object*, *arg1*)

ここで、*object* はマクロに渡されるテーブル オブジェクトです。*arg1* はフィルタリング基準を指定します。ここでは、cpmCPUThreshold クラスに基づいてフィルタリングしています。テーブル内で cpmCPUThresholdClass 値が 1 のオブジェクトの CPU データは保持され、それに一致しないオブジェクトおよびそれらのデータはテーブルから削除されます。

ここまでで、cpmCPUTotalTable と cpmCPUThresholdTable の 2 つのテーブルができます。処理効率を最大限に高めるために、これら 2 つのテーブルを 1 つの仮想テーブルに結合します。PollDefinition セクションの次の行でそれが行われています。

```
cpmCPUTotalTable =
        cpmCPUTotalTable.leftJoin(cpmCPUThresholdTable,
  (cpmCPUTotalTable.cpmCPUTotalIndex == cpmCPUThresholdTable.cpmCPUTotalIndex))
```

これは、LEFTJOIN マクロを呼び出すことで行われます。このマクロは、cpmCPUThresholdTable と cpmCPUTotalTable を照らし合わせ、インデックス値が同じならば、2 つのテーブルの行を 1 つのテーブルに結合します。

これで、すべての処理対象データが含まれる 1 つのテーブルができました。

LEFTJOIN マクロ

LEFTJOIN マクロの構文は次のとおりです。

LEFTJOIN (*object*, *arg1*, *arg2*)

ここで、*object* と *arg1* はテーブルです。*arg2* は各行の一致条件です。

このマクロは、*object* と *arg1* の結合結果のテーブルを返します。

object の行と *arg1* の行は、条件（*arg2*）が true になる場合に結合されます。ただし、各オブジェクト行は、結果テーブルで *arg1* で指定したオブジェクトから回線を対応付けられていない場合も保持されます。

（注） setCpuInfo（）は ENTITY-MIB からデータを取得し、指定されたフィールドのデータが入力する特別なマクロです。

ポール処理結果の指定（ProcessPollResult セクション）

ProcessPollResult セクションは、CPU レポート用の情報を取得するために、データを操作し、それを処理する変数をセットアップします。

例 2-3 に、cpu.xml ファイルの ProcessPollResult セクションを示します。

例 2-3　ProcessPollResult セクション

```
<ProcessPollResult>

      setCpuInfo();

      fiveMinUtil       = cpmCPUTotal5minRev / 100;
      oneMinUtil        = cpmCPUTotal1minRev / 100;
      risingThreshold   = cpmCPURisingThresholdValue / 100;
      fallingThreshold  = cpmCPUFallingThresholdValue / 100;

   </ProcessPollResult>
```

次のことを行う必要があります。

1. CISCO-PROCESS-MIB に定義されている 2 つの時間間隔での CPU 平均使用率についてパーセント値を計算します。cpmCPUTotal5minRev は 5 分間隔、cpmCPUTotal1minRev は 1 分間隔です。
2. さらに、CPU 平均ピーク使用率についてパーセント値を計算します。この数値の計算に使用されるデータは、cpmCPURisingThresholdValue 値と cpmCPUFallingThresholdValue 値に保管されています。
3. SETCPUINFO マクロを呼び出すことから始めます。このマクロは、インデックスの指定がなければ、テーブル内にデータが返される CPU ごとに、CPU の説明、CPU 番号、および CPU スロットを単純に設定します。

CPU 平均使用率と CPU 平均ピーク使用率の決定に使用されるパーセント値は、各 MIB 変数のデータを 100 で割ることにより計算されます。

これで、Prime Performance Manager レポートに表示するデータが得られました。

次に、計算された値を Prime Performance Manager ゲートウェイのデータベースに割り当てます。これは、XML コードの次のセクション（ProcessDBSummary セクション）で行われます。

データベース スキーマへのデータの割り当て（ProcessDBSummary セクション）

ProcessDBSummary セクションは、レポートに表示される順序でデータが並ぶテーブル行を仮想データ テーブル内にセットアップします。

例 2-4 に、cpu.xml レポートの ProcessDBSummary セクションを示します。

例 2-4　ProcessDBSummary セクション

```
<ProcessDBSummary name="CPU" baseTableName="CPU">

      <Var name="CPUSlot"     type="Integer" key="true">cpuSlot</Var>
      <Var name="CPUNum"      type="Integer" key="true">cpuNum</Var>
      <Var name="CPUDescr"    type="String" key="true">cpuDescr</Var>

      <Var name="CPUUtilMax5min" type="Double" operation="Max">fiveMinUtil</Var>
      <Var name="CPUUtilAvg5min" type="Double" operation="Avg">fiveMinUtil</Var>

      <Var name="CPUUtilMax1min" type="Double" operation="Max">oneMinUtil</Var>
      <Var name="CPUUtilAvg1min" type="Double" operation="Avg">oneMinUtil</Var>
```

```
      <Var name="CPURisingThreshold" type="Double">risingThreshold</Var>
      <Var name="CPUFallingThreshold" type="Double">fallingThreshold</Var>

    </ProcessDBSummary>
```

ProcessDBSummary セクションの最初の行は、ポールの名前とレポートの名前を指定します。

ProcessDBSummary セクション内の残りのコードは、基本的に、仮想データベースの各行にセットアップされるカラムをセットアップします。

一部のテーブル カラムの操作変数で指定されている操作は、*EventPoller.xsd* スキーマ ファイルに定義されています。

このコードは、カラムをセットアップする変数でテーブル カラムをセットアップします。たとえば、このセクションの最初の 3 行は、CPU スロット番号、CPU 番号、CPU の説明をセットアップします。

```
<Var name="CPUSlot"     type="Integer" key="true">cpuSlot</Var>
      <Var name="CPUNum"      type="Integer" key="true">cpuNum</Var>
      <Var name="CPUDescr"    type="String" key="true">cpuDescr</Var>
```

レポートの [Average Utilization] パートと [Peak Utilization] パートに使用される 2 つのカラムには、平均値と最大値の計算が必要です。これらは、次のように `operation="Max"` や `operation = "Avg"` が含まれる行で指定されます。

```
      <Var name="CPUUtilMax5min" type="Double" operation="Max">fiveMinUtil</Var>
      <Var name="CPUUtilAvg5min" type="Double" operation="Avg">fiveMinUtil</Var>

      <Var name="CPUUtilMax1min" type="Double" operation="Max">oneMinUtil</Var>
      <Var name="CPUUtilAvg1min" type="Double" operation="Avg">oneMinUtil</Var>
```

ポーリング データおよびそのデータから計算された値は、ユーザが Prime Performance Manager ユーザ インターフェイスで選択する任意の CPU レポートで使用できるように、メモリ内の仮想テーブルに保持されます。

定義されたポーリング間隔が終了するたびに、その指定されたポーリング間隔でセットアップされた仮想テーブルのデータが Prime Performance Manager ゲートウェイ上の物理データベースに書き込まれます。

ここまでで、各ポーリング対象デバイスをレポートするための、MIB のポーリングされた変数からのデータの抽出、データの操作、およびテーブルのセットアップが完了しました。

次に、レポートの形式について指定することができます。これは、次の 2 つのセクション（CSV セクションと WebReport セクション）で行われます。

CSV 出力ファイルと形式の指定（CSV セクション）

CSV セクションは、ユーザが CSV レポート オプションを選択した際にデータがカンマ区切り値（CSV）ファイルに書き込まれるときのデータのレイアウトを指定します。名前タグはこのレポート用に生成されるすべてのファイルのファイル名のプレフィックスを指定します。

例 2-5 に、cpu.xml レポートの CSV セクションを示します。

例 2-5　　CSV セクション

```
<CSV name="CPU" location="gateway" listen="CPU">
    <Column name="Slot">CPUSlot</Column>
    <Column name="Number">CPUNum</Column>
    <Column name="Description">CPUDescr</Column>

    <Util   name="CPUUtilMax5min">CPUUtilMax5min</Util>
    <Util   name="CPUUtilAvg5min">CPUUtilAvg5min</Util>
```

```
    <Util   name="CPUUtilMax1min">CPUUtilMax1min</Util>
    <Util   name="CPUUtilAvg1min">CPUUtilAvg1min</Util>

    <Column name="CPURisingThreshold">CPURisingThreshold</Column>
    <Column name="CPUFallingThreshold">CPUFallingThreshold</Column>
  </CSV>
```

CSV セクションの最初の行は、3 つの値を指定します。

- CSV ファイルの名前。
- レポートの場所はこの場合 Prime Performance Manager ゲートウェイです。
- 基本テーブル名が使用中の listen 変数。

CSV セクションの残りの行は、テキスト文字列またはデータベース カラム データを単純に指定します。

```
Column name="Slot">CPUSlot</Column>
    <Column name="Number">CPUNum</Column>
    <Column name="Description">CPUDescr</Column>

    <Util   name="CPUUtilMax5min">CPUUtilMax5min</Util>
    <Util   name="CPUUtilAvg5min">CPUUtilAvg5min</Util>
    <Util   name="CPUUtilMax1min">CPUUtilMax1min</Util>
    <Util   name="CPUUtilAvg1min">CPUUtilAvg1min</Util>

    <Column name="CPURisingThreshold">CPURisingThreshold</Column>
    <Column name="CPUFallingThreshold">CPUFallingThreshold</Column>
```

これで、Web レポートをセットアップする準備が整いました。

Web レポートのセットアップ（GraphReport セクションと TableView セクション）

WebReport セクション（XML 要素）には次のものが含まれます。

- **属性**：Prime Performance Manager のメニュー内で使用されるこのレポートの名前、レポート ツリー内でこのレポートが表示される場所、このレポート データの取得元、この Web レポートのプロパティ ファイルなどを指定します。
- **GraphView セクション**：レポートのグラフ ビューをセットアップします。
- **TableView セクション**：レポートのテーブル ビューをセットアップします。

WebReport セクションの属性の指定

WebReport セクションの属性は、レポートが表示される場所と使用するデータを決める重要な情報を指定します。

各属性は、XML レポートのプロパティ ファイルで指定されている値を設定します。cpu.xml レポートのプロパティ ファイルの詳細については、「WebReport セクションの属性の指定」（P.2-13）を参照してください。

例 2-6 に、cpu.xml ファイルの WebReport セクションに指定されている属性を示します。

例 2-6　　WebReport の属性

```
<WebReport name="wrnCPUUtil"
             category="level1Resources,level2CPU"
```

```
reportId="CPU"
context="Network,Node,CPUSlot,CPUNum,CPUDescr"
textProps="cpu"
sortWeight="2">
```

例 2-6 に示されているコードは、次の属性を設定しています。

- **name**：一意の名前を指定します。これは、プロパティ ファイル（cpu.properties）で指定されます。Prime Performance Manager のメニューに表示されるレポート名になります。名前はプロパティ ファイルにマップする必要はありません。プロパティ ファイルにマッピングされていないここで指定される値を次に示します。
- **category** : Prime Performance Manager のレポート ツリー内でこのレポートが表示される場所を指定します。

  たとえば、category=level1Resources,level2CPU は、この CPU レポートが [Resources] レポート カテゴリの下に表示され、テキストである CPU で識別されることを指定しています。level1Resources と Resources の間および level2CPU と CPU の間の関係はプロパティ ファイルから取得されます。プロパティ ファイルがない場合、このカテゴリのレポートは level1Resources カテゴリおよび level2CPU サブ カテゴリにあります。
- **reportId**：ポーリング定義で定義された reportId にマッピングする必要があります。
- **context**：レポートで使用可能なドリルダウン オプションと、レポートを表示できる場所を指定します。
- **textProps**：使用するプロパティ ファイルの名前を指定します（cpu.properties）。
- **sortWeight**：割り当てられたカテゴリ内でのレポートの順序を指定します。これはスキーマで定義された多くの選択可能なプロパティの 1 つです。

  行 sortWeight=2 はレポート タイプがレポート一覧に表示される順序を指定します。

プロパティ ファイルの使用方法

プロパティ ファイルの目的は 1 つだけです。それは、Web GUI に表示されるテキスト文字列を保持する変数をセットアップすることであり、これらの変数は、ユーザがグラフ ビューまたはテーブル ビューを選択した際に使用されます。

（注）　CSV 名は XML 内の CSV column/util タグ内で指定され、プロパティ ファイルにはありません。

例 2-7 に、cpu.xml レポートのプロパティ ファイルを示します。

例 2-7　cpu.properties ファイル

```
level1Resources = Resources
level2CPU       = CPU

#CPU Utilization
wrnCPUUtil      = CPU Utilization
gstCPUUtil      = CPU Utilization

gtCPUUtilAvg     = CPU Average Utilization

genCPUUtilAvg    = Average Utilization

gtCPUUtilPeak    = CPU Average Peak Utilization

gtCPUUtil        = CPU Average and Peak Utilization
```

```
genCPUUtilPeak   = Peak Utilization

avg              = Avg
peak             = Peak

rising           = Rising
falling          = Falling

node             = Node
slot             = Slot
cpu              = CPU
cpuDes           = CPU Description

cpuUtil5Min   = 5 Min Util
cpuUtil1Min   = 1 Min Util
cpuUtilThresh = Threshold
```

GraphView セクションのコーディング

GraphView セクションでは、次のようなレポート グラフ ビューを設定します。

- グラフ サマリー テーブルのタイトル。
- 描画されるグラフのタイトルとカラム名。1 つのグラフ ビューで、グラフをいくつでも表示できます。グラフ列は KPI である場合があります。
- リーフ グラフ（該当する場合）。
- 列、Util、およびバイト数も使用できます。列には値がそのまま表示されます。Util は値に 100 を掛け、パーセンテージで表示します。

表示されるテキストは、例 2-8 に示されているように、レポートのプロパティ ファイルに定義された変数を使用して指定する必要があります。バイト値を必要に応じて KB、MB、および GB 形式で表示するために Bytes タグを使用できます。

例 2-8 に、cpu.xml レポートの GraphView セクションを示します。

例 2-8　　GraphView セクション

```
<GraphView>
      <GraphSummary title="gstCPUUtil" />
      <Graph title="gtCPUUtilAvg" >
        <Util name="genCPUUtilAvg">CPUUtilAvg5min</Util>
      </Graph>
      <Graph title="gtCPUUtilPeak" >
        <Util name="genCPUUtilPeak">CPUUtilMax5min</Util>
      </Graph>
      <LeafGraph title="gtCPUUtil" >
        <Util name="genCPUUtilAvg">CPUUtilAvg5min</Util>
        <Util name="genCPUUtilPeak">CPUUtilMax5min</Util>
      </LeafGraph>
    </GraphView>
```

GraphSummary 要素は、Prime Performance Manager グラフィカル ユーザ インターフェイスでレポート グラフの上に表示されるタイトルを指定します。

```
<GraphSummary title=”gstCPUUtil” />
```

この場合、タイトルは、プロパティ ファイル内の *gstCPUUtil* 変数で指定されたもの（「CPU Utilization」）になります。

最初のレポート表示時のサマリー テーブルの非表示化

レポートが初めて表示されるときにサマリー テーブルを非表示にする場合は、minimized 属性を次の例のように使用できます。

```
<GraphSummary title="ifDashboard" minimized="true"/>
```

（注）　ユーザ単位でサマリー タイトルの表示を変更できます。

Graph セクションのコーディング

GraphView セクション内の Graph セクションは、グラフ ビューに表示されたグラフを識別するテキスト文字列を指定します。この CPU レポートでは、2 つのグラフがプロパティ ファイルの定義に従って識別されます。識別されるグラフは、CPU Average Utilization と CPU Average Peak Utilization です。

Graph セクション内では、次のことができます。

- Column 要素をコーディングすることにより、データベースの値を「そのまま」表示できます。列要素（または util）も操作を実行できます。次に例を示します。

  ```
  (Column name="someName") (CPUUtilAvg5min + CPUUtilMax5min)/2 (/Column)
  ```

- Util 要素をコーディングすることにより、データに対して操作を実行できます。

cpu.xml レポートの Graph セクション内のコードは、基準値の 100 倍の Util 値を指定しています。

LeafGraph セクションのコーディング

レポートがサブ レポートを使用して最低レベルに到達するまでドリルダウンしたとき、レポート対象のオブジェクトのインスタンスが 1 つしか残っていなければ、そのオブジェクトのさまざまなデータ メトリックを 1 つのグラフに合成するために LeafGraph をコーディングできます。

たとえば、CPU レポートの場合、ユーザは、各プロセッサの [CPU Description] カラム内にあるテキストをクリックして、CPU Average and Peak Utilization レポートと呼ばれる合成されたレポートを表示できます。このようなサブレポートは、LeafGraph セクションで定義されます。

CPU レポートの LeafGraph セクションは、次のように指定されています。

```
<LeafGraph title="gtCPUUtil" >
        <Util name="genCPUUtilAvg">CPUUtilAvg5min</Util>
        <Util name="genCPUUtilPeak">CPUUtilMax5min</Util>
      </LeafGraph>
```

LeafGraph セクションの title 属性は、プロパティ ファイルの変数（gtCPUUtil）を指定しています。この変数は、リーフ レポートのタイトルである CPU Average and Peak Utilization を指定しています。行はデフォルトのチャート データ シリーズの出力形式です。次のようにタイプ属性を追加して基本バーまたはスタックされたパーセンテージ列の出力形式を指定できます。

```
<LeafGraph title="gtCPUUtil" type="bar">
```

または、

```
<LeafGraph title="gtCPUUtil" type="StackedPercentageColumn">
```

個別にチャートの出力形式を変更できます。詳細については、『*Cisco Prime Performance Manager 1.4 User Guide*』の「Managing Reports, Dashboards, and Views」の章を参照してください。

Util セクションは、リーフ レポート Peak Utilization and Average Utilization の上部のテーブルに表示される 2 つのレポート項目のタイトルを指定しています。Util 変数は、表示されるテーブル データを指定しています。

TableView セクションのコーディング

cpu.xml レポートの最後のセクションは、レポートのテーブル ビューの形式を設定します。

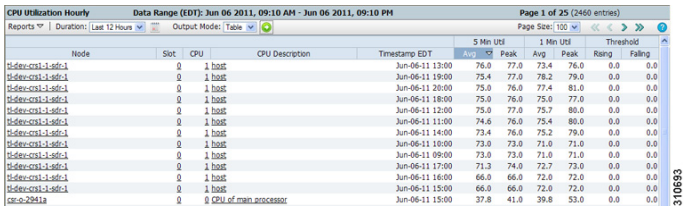
例 2-9 に、cpu.xml レポートの TableView セクションを示します。

例 2-9　　TableView セクション

```
<TableView baseTable="CPU">
      <IdLabel/>
      <Label colSpan="2" name="cpuUtil5Min" />
      <Label colSpan="2" name="cpuUtil1Min" />
      <Label colSpan="2" name="cpuUtilThresh" />
      <HeaderRow/>

      <Link name="node" context="Node">fqdnid</Link>
      <Link name="slot" context="CPUSlot">CPUSlot</Link>
      <Link name="cpu"  context="CPUNum">CPUNum</Link>
      <Link name="cpuDes"  context="CPUDescr">CPUDescr</Link>
      <Time/>


      <Util default="true" name="avg">    CPUUtilAvg5min</Util>
      <Util                name="peak">   CPUUtilMax5min</Util>

      <Util                name="avg">    CPUUtilAvg1min</Util>
      <Util                name="peak">   CPUUtilMax1min</Util>

      <Util                name="rising"> CPURisingThreshold</Util>
      <Util                name="falling">CPUFallingThreshold</Util>
    </TableView>
```

図 2-1 に、CPU レポートのテーブル ビューを示します。

図 2-1　　テーブル ビューの例



TableView セクション内のコードは、テーブルのレイアウトをセットアップする属性を指定します。

- 図 2-1 に示されているように、見出しには、2 つの副見出しを持つものがあります。たとえば、[5 Min Util] には [Avg] と [Peak] の副見出しがあり、[Threshold] には [Rising] と [Falling] の副見出しがあります。

  これらは、3 つの Label セクションで指定されます。

  ```
  <Label colSpan="2" name="cpuUtil5Min" />
  <Label colSpan="2" name="cpuUtil1Min" />
  <Label colSpan="2" name="cpuUtilThresh" />
  ```

colSpans 属性は、主見出しが 2 つのサブカラムにわたることを指定しています。name 属性は、プロパティ ファイルの指定に従って見出しのテキストを定義する変数を指定します。たとえば、cpuUtil5Min は、5 Min Util という文字列を指定しています。

- いくつかのレポート カラムには、他の項目へのリンクが含まれています。たとえば、[CPU Description] カラムには、選択された CPU のレポートを表示するリンクが含まれています。また、[CPU Description] カラムには、選択されたノードのレポートを表示するリンクが含まれています。

  これらのリンクは、次のように Link セクションで定義されます。

```
<Link name="node" context="Node">fqdnid</Link>
<Link name="slot" context="CPUSlot">CPUSlot</Link>
<Link name="cpu"  context="CPUNum">CPUNum</Link>
```

  各 Link セクションの name 属性は、項目を説明する文字列（プロパティ ファイル内の変数で定義される）を指定します。たとえば、slot ならば「Slot」が指定されます。context 属性は、ProcessDBSummary セクション内のカラムを指定します。たとえば、slot の場合、CPUSlot のデータ カラムが表示されます。

  Time セクションは、時間範囲レポートごとのデータが含まれる DBSummary テーブルのカラムと、サブカラムの見出しを指定します。たとえば、[Threshold] カラムの [Rising] サブカラムに対し、このコードは Util 値を指定しています。

```
<Util name="rising"> CPURisingThreshold</Util>
```

  Util 操作をデータに適用したくない場合は、代わりに Column 要素をコーディングできます。この場合、データは変更されずに表示されます。

しきい値設定可能フィールドを使用したレポートの作成

多くの Prime Performance Manager のテーブルおよびグラフ ビュー フィールドは、Prime Performance Manager Threshold Editor を使用してしきい値を作成できることをユーザに示すために + を使用して事前設定されています。（しきい値の作成の詳細については、『*Cisco Prime Performance Manager 1.4 User Guide*』の「Managing Thresholds」を参照してください）。

レポート XML ファイルの WebReport 宣言に定義されている数値列のどの値も、その値がキー値またはリンク値ではなくそして「thresholdable="false"」属性がないかぎりは、しきい値として設定できます。

しきい値を作成する場合は、主要業績評価指標（KPI）プロパティを編集します。KPI は上昇または下限のいずれかです。上昇しきい値では、クリティカル アラームしきい値はメジャー アラームしきい値より高くする必要があり、メジャー アラームしきい値はマイナー アラームよりも高くする必要があります。下限しきい値では、クリティカル アラーム エントリはメジャー アラームよりも低くする必要があり、メジャー アラームはマイナー アラームより低くする必要があります。

予想される上昇または下限 KPI 値のしきい値設定可能フィールドの作成に tcaRising 属性を使用できます。この属性は、レポートの XML ファイルの GraphView または TableView 宣言の下に置かれます。tcaRising=true（デフォルト）を設定した場合、このフィールドの KPI は上昇になります。tcaRising=false を設定すると、KPI は下限になります。

tcaRising 属性を使用できないフィールドは次のとおりです。

- すべての LeafGraph フィールド。
- すべてのダッシュボード レポート XML ファイル フィールド。ダッシュボード レポートのファイル名は、*Dash.xml になります。

次は、ping.xml に基づくしきい値設定可能フィールドの例です。

```
<!-- *** ICMP Ping Availability *** -->
```

```
  <WebReport name="wrnICMPPingAvail"
             category="level1Availability,level2ICMPPing"
             reportId="ICMP_PING"
             context="Network,Node,"
             sortWeight="22"
             textProps="ping">

    <GraphView>

       <GraphSummary title="gstICMPPingAvail" />
       <Graph title="gtICMPPingAvailAvg" >
          <Util name="genICMPPingAvailAvg" default="true"
                descending="false" tcaRising="false">PingAvailability</Util>
       </Graph>

       <LeafGraph title="gtICMPPingAvailAvg" type="bar">
          <Util name="genICMPPingAvailAvg">PingAvailability</Util>
NOTE: the tcaRising attribute is not added here.
       </LeafGraph>

    </GraphView>

    <TableView baseTable="ICMP_PING">
      <IdLabel/>
      <Label colSpan="1" name="icmpPingAvail" />
      <Label colSpan="3" name="latency" />
      <HeaderRow/>

      <Link name="node"  context="Node">fqdnid</Link>
      <Time/>

      <Util name="genICMPPingAvailAvg" default="true"
            descending="false" tcaRising="false">PingAvailability</Util>

      <Column name="avg">PingLatencyAvg</Column>
      <Column name="max">PingLatencyMax</Column>
      <Column name="min">PingLatencyMin</Column>

    </TableView>
```

オンライン ヘルプ ファイルの提供

レポートを作成した後、そのレポート用のオンライン ヘルプ ファイルを提供できます。レポートのカスタマイズされたオンライン ヘルプを提供した場合、ユーザがレポートの [Help] ページで [Custom Help] を選択すると、そのオンライン ヘルプが表示されます。

独自のオンライン ヘルプ ファイルを作成するには、次の手順を実行します。

ステップ 1　レポートのタイトルが含まれる HTML ファイルを作成します。たとえば、レポートの名前が *test.xml* の場合、*test.xml.custhlp*.html という名前の HTML ファイルを作成します。

ステップ 2　ファイルの中身を記述します。

ステップ 3　このヘルプ ファイルをゲートウェイ上の次のディレクトリにコピーします。

/opt/CSCOppm-gw/apache/share/htdocs/reportHelp/user/

オンライン ヘルプ ファイルの手動生成

システムによって生成されるオンライン ヘルプ ファイルを手動で生成する場合は、ゲートウェイのコマンド ラインから次を入力します。

```
ppm docreps
```

ベスト プラクティス

ここでは、レポートを作成する際に従う必要のあるベスト プラクティスについていくつか説明します。

/system ディレクトリ内のファイルを編集しない

/opt/CSCOppm-gw/etc/pollers/system ディレクトリ内のファイルを編集してはなりません。既存のレポート ファイルとそれに関連したプロパティ ファイルを新規のカスタマイズされたレポートのひな形として使用する場合は、それらのファイルを */opt/CSCOppm-gw/etc/pollers/user* ディレクトリにコピーし、その場所でそれらを編集します。ファイルの名前が同じである場合、ユーザのファイルが使用されます。これは、既存のシステム ファイルのカスタマイズ バージョンを作成する最適な方法です。また、Web Report Editor グラフィカル ユーザ インターフェイスを使用してユーザ ディレクトリの下にコピーを保存するレポートを記述できます。

SystemCapability.xml ファイルを編集しない

SystemCapability.xml ファイルを編集してはなりません。このファイルは、*/opt/CSCOppm-gw/etc* ディレクトリにあります。独自のレポートに使用される機能を修正するには、必要に応じて *UserCapability.xml* ファイルを編集します。このファイルも */opt/CSCOppm-gw/etc* ディレクトリにあります。

新しい .xml ファイルおよび .properties ファイルのユーザ ディレクトリへの追加

新しい *.xml* ファイルまたは *.properties* ファイルを追加する必要がある場合は、それらのファイルを */opt/CSCOppm-gw/etc/pollers/user* ディレクトリに追加し、編集します。

既存のレポートで使用されるスキーマを修正する際は注意すること

レポートのコードを作成するとき、以前に作成されたレポートで使用されるスキーマを修正する際は注意してください。内部スキーマを、それに影響を及ぼす既存のコーディングを考慮せずに修正すると、内部データベースが不安定になって、データベースの再初期化が必要になる場合があります。

レポートには一意のポーラー名、レポート ID、およびデータベース テーブル名を使用

レポートには常に一意のポーラー名、レポート ID、およびデータベース テーブル名を使用してください。既存の名前を使用すると、重大なシステムの問題が発生する可能性があります。

プロパティ ファイルを使用して共通の設定を指定

レポートの XML ファイルに関連付けられた *.properties* ファイルを使用して、レポートの複数のセクションで使用される設定や複数のレポート間で使用される設定を指定します。これにより、処理の一貫性が保証され、同じ変数を複数回コーディングすることにより発生する可能性のある誤りも排除できます。

タスク リファレンス

ここでは、レポートの作成中に実行する可能性のある一般的なタスクのリファレンス情報を提供します。

1 分および 5 分のレポートを有効にします

1 分および 5 分のレポートはデフォルトでは有効になっていません。レポートに対応する XML レポート定義ファイルを編集して 1 分および 5 分のレポートを有効にできます。また、ユーザが Prime Performance Manager グラフィカル ユーザ インターフェイスを使用して 1 分および 5 分のレポートを有効にできるようにできます。ProcessDBSummary セクションと WebReport セクションに interval 属性を追加する必要があります。

（注） レポートの xml の間隔の設定を必要とすることはあまりありません。レポート間隔は通常はグラフィカル ユーザ インターフェイスのレポート状態から制御されます。使用可能なオプションを制限する場合はレポート XML で設定できますが、レポートが非常に大きいテーブルを生成できる場合は 1 分または 5 分などの低レベル値をレポートに設定することは通常はできません。

ここでは、cpu.xml レポート定義ファイル内で、interval 属性を CPU 使用率レポートに追加して 5 分のレポーティングをイネーブルにする例を示します。

```
<ProcessDBSummary name="CPU" baseTableName="CPU"
interval="Min1Min5Min15HourlyDailyWeeklyMonthly">
<WebReport name="wrnCPUUtil"
             category="wrcVendor,wrcCategory"
             reportId="CPU"
             context="Network,Node,CPUSlot,CPUNum,CPUDescr"
             textProps="cpu"
             sortWeight="2" interval="Min5Min15HourlyDaily">
```

1 分または 5 分間隔は、間隔属性の Min1Min5 部分なので含まれています。1 分および 5 分間間隔は、次のように定義されます。

```
interval="1Min5Min15MinHourlyDailyWeeklyMonthly
```

この属性は、デフォルトでは定義されません。定義されていなければ、15 分、毎時、および毎日のレポートがデフォルトとして設定されます。

また、1 分および 5 分レポートは GUI を使用してグローバルに有効にできます。

- [Reports/Group Settings] タブで、1 分および 5 分レポートをグローバルに有効にします。
- [Reports/Group Status] タブで、特定のレポートの 1 分および 5 分レポートを有効にします。また、これはデバイス レベルでも設定できます。

（注） 1 分および 5 分レポートの有効化は、ご使用のユニットに必要なリソースの量が大幅に増加します。

既存のレポートへの新しいカラムの追加

新しいカラムを既存のレポートに追加するには、次の手順を実行します。

ステップ 1　必要な MIB 属性が PollDefinition セクションでポーリングされていない場合は、ポーリング定義を修正します。

ステップ 2　「ProcessPollResult セクションの修正」（P.2-23）の説明に従って必要な計算式を含めます。

ステップ 3　<ProcessDBSummary> 要素内で *Var* 要素を使用して、データベース テーブルに変数を追加します（「ProcessDBSummary セクションの修正」（P.2-22）を参照）。

ステップ 4　変更内容を保存し、ゲートウェイとユニットを再起動します。

ProcessDBSummary セクションの修正

ProcessDBSummary セクションに変数を追加するには、次の手順を実行します。

ステップ 1　必要な MIB 属性が PollDefinition セクションでポーリングされていない場合は、ポーリング定義を修正します（「ポール定義の修正」（P.2-23）を参照）。

ステップ 2　計算式が必要な場合は、ProcessPollResult セクションにその計算式を追加します（「ProcessPollResult セクションの修正」（P.2-23）を参照）。

ステップ 3　次の例のように、*Var* を使用して変数を追加します。

例 1：式の計算を必要としない変数の場合、ProcessDBSummary セクションでコードを次のように追加します。

```
<ProcessDBSummary name="CPU" baseTableName="CPU">
<Var name="CPUSlot"     type="Integer" key="true">cpuSlot</Var>
</ProcessDBSummary>
```

例 2：式の計算を必要とする変数の場合、次のように、計算式を ProcessPollResult セクションに追加し（「ProcessPollResult セクションの修正」（P.2-23）を参照）、その結果を ProcessDBSummary セクションに追加します。

```
<ProcessPollResult>

  fiveMinUtil       = cpmCPUTotal5minRev / 100;

</ProcessPollResult>

<ProcessDBSummary name="CPU" baseTableName="CPU">

       <Var name="CPUUtilMax5min" type="Double" operation="Max">fiveMinUtil</Var>
```

```
        <Var name="CPUUtilAvg5min" type="Double" operation="Avg">fiveMinUtil</Var>
</ProcessDBSummary>
```

ポール定義の修正

ポール定義を修正するには、次の手順を実行します。

ステップ 1　このレポートの SystemCapability で定義した条件を含めます。

例：

```
<Criteria>CISCO_PROCESS_MIB</MIBLevel>
```

ここで、CISCO_PROCESS_MIB は SystemCapability.xml で定義される機能です。

ステップ 2　次のように、poll 関数を PollDefinition セクションに含めます。

```
cpmCPUTotalTable = poll("cpmCPUTotalIndex,cpmCPUTotal5minRev");
```

ここで、cpmCPUTotalIndex と cpmCPUTotal5minRev は、CISCO_PROCESS_MIB の MIB 属性です。

ProcessPollResult セクションの修正

ProcessPollResult セクションを修正するには、次の手順を実行します。

ステップ 1　必要な MIB 属性が PollDefinition セクションでポーリングされていない場合は、「ポール定義の修正」（P.2-23）を参照してください。

ステップ 2　計算式を記述し、それを変数に代入します。

たとえば、cpu.xml の場合、cpmCPUTotal5minRev はポーリングされた変数であり、計算される式は cpmCPUTotal5minRev / 100 です。

ステップ 3　次のように、計算式を <ProcessPollResult> セクションに含めます。

```
<ProcessPollResult>

     fiveMinUtil       = cpmCPUTotal5minRev / 100;

</ProcessPollResult>
```

fiveMinUtil は、ユーザによって与えられた意味のある名前です。これは、ProcessDBSummary セクションなどの他の XML セクション内で使用できます。

Cisco Prime Network 内でのレポートの相互起動のセットアップ

Cisco Prime Performance Manager を Cisco Prime Networ と組み合わせて使用する場合、Cisco Prime Network Vision のデバイス ショートカット メニューからの Prime Performance Manager レポートの相互起動をセットアップできます。

これを行うには、次の 2 つの方法があります。

- Prime Performance Manager GUI の使用
- **ppm crosslaunch** CLI コマンドの使用

Prime Performance Manager GUI の使用

Prime Performance Manager GUI を使用して相互起動をセットアップするには、次の手順を実行します。

ステップ 1　[System] メニューから、[Prime Network Integration] を選択します。

ステップ 2　[Prime Network] タブをクリックします。

[Prime Network Gateway] ページが表示されます。

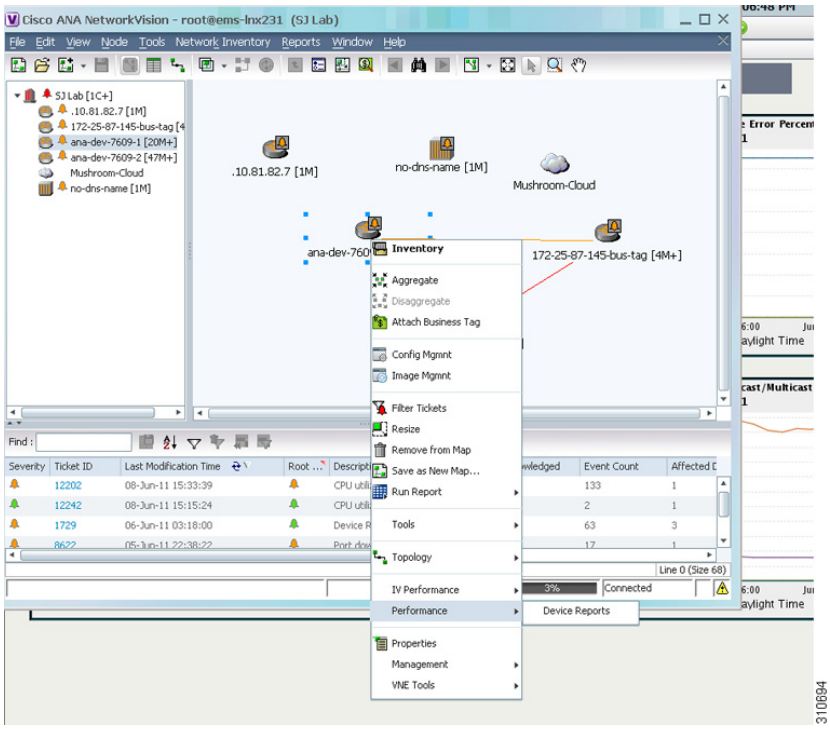
ステップ 3　[Prime Network Gateway] ページ上で、Prime Network ゲートウェイにログインするためのパラメータを入力します。

ステップ 4　[Install Cross Launch] アイコンをクリックします。

Prime Performance Manager ゲートウェイは、Prime Network ゲートウェイと通信し、Cisco Prime Performance Manager レポートのメニュー選択をデバイス ショートカット メニューに追加する BQL スクリプトを自動的にインストールします。

図 2-2 に例を示します。

図 2-2　相互起動のメニュー選択のあるデバイス ショートカット メニュー



このプロセスは、[Performance] > [Device Reports] のメニュー選択を Prime Network のデバイス ショートカット メニューに追加します。

これで、Prime Network ユーザは、インストールされたレポートをサポートするノードまたはインターフェイスからノード レベルまたはインターフェイス レベルのレポートを起動できるようになりました。

コマンド ラインの使用

Prime Performance Manager CLI を使用して相互起動をセットアップするには、ゲートウェイ上で次のコマンドを入力します。

```
ppm crosslaunch
```

独自レポートの相互起動の追加

Prime Performance Manager ゲートウェイ上の *CSCOpm-gw/etc/bql/xl* ディレクトリには、Prime Network 内の相互起動ポイントを設定および削除するための BQL 例がいくつか含まれています。

表 2-1　Cisco Prime Performance Manager と一緒に提供される BQL 相互起動スクリプト

スクリプト名	説明
l2vpn-atm.bql l2vpn-atm-remove.bql	L2VPN ATM レポートの相互起動ポイントを追加、削除
l2vpn-epl-evpl.bql l2vpn-epl-evpl-remove.bql	L2VPN EPL EVPL レポートの相互起動ポイントを追加、削除
l2vpn-frame-relay.bql l2vpn-frame-relay-remove.bql	L2VPN Frame Relay レポートの相互起動ポイントを追加、削除
l2vpn-tdm.bql l2vpn-tdm-remove.bql	L2VPN TDM レポートの相互起動ポイントを追加、削除
l2vpn-vpls.bql l2vpn-vpls-remove.bql	L2VPN VPLS レポートの相互起動ポイントを追加、削除
l3vpn-logical.bql l3vpn-logical-remove.bql	L3 VPN VRF レポートの相互起動ポイントを追加、削除
l3vpn.bql l3vpn-remove.bql	L3 VPN レポートの相互起動ポイントを追加、削除
mpls-in-segment.bql mpls-in-segment-remove.bql	MPLS In Segment レポートの相互起動ポイントを追加、削除
mpls-interface.bql mpls-interface-remove.bql	MPLS Interface レポートの相互起動ポイントを追加、削除
mpls-ldp.bql mpls-ldp-remove.bql	MPLS LDP レポートの相互起動ポイントを追加、削除
mpls-out-segment.bql mpls-out-segment-remove.bql	MPLS Out Segment レポートの相互起動ポイントを追加、削除
chassis.bql chassis-remove.bql	
hostServer.bql hostServer-remove.bql	
module.bql module-remove.bql	
virtual-machine.bql virtual-machine-remove.bql	

相互起動ポイントを設定する BQL スクリプトすべてに、それぞれの相互起動を削除するための BQL ファイルが用意されていることに注意してください。

次の例では、Prime Network 内にノード レベルの相互起動ポイントを追加する相互起動 BQL を示します。

例 2-10 に、Prime Performance Manager レポートをノード レベルで起動するための [Performance] > [Device Reports] メニュー選択をユーザに提供して、相互起動ポイントを追加する方法を示します。

例 2-10　node.bql ファイル

```
<command name="Set">
    <param name="imo">
        <value>
            <management.IExternalLaunch>
                <ID
type="Oid">{[ExternalLaunch(ContextImoType=com.sheer.imo.IManagedElement)(Name=deviceRepor
t)]}</ID>
                <MenuCaption type="String">Device Reports</MenuCaption>
                <MenuPath type="String">Performance</MenuPath>
                <LineToExecute
type="String">$ppmProtocol$://$ppmWebAddress$:$ppmWebPort$/ppm/jsp/navMain.jsp?displayType
=reportTab&amp;FQDN=Node=$com.sheer.imo.IManagedElement.DeviceName$</LineToExecute>
            </management.IExternalLaunch>
        </value>
    </param>
    <param name="replace">
        <value>true</value>
    </param>
</command>
```

例 2-11 に、この相互起動ポイントを削除するためのサンプル BQL を示します。

例 2-11　node-remove.bql ファイル

```
<command name="Delete">
    <param name="oid">
        <value>
            <management.IExternalLaunch>
                <ID
type="Oid">{[ExternalLaunch(ContextImoType=com.sheer.imo.IManagedElement)(Name=deviceRepor
t)]}</ID>
            </management.IExternalLaunch>
        </value>
    </param>
</command>
```

独自のレポートを作成した後は、付属の BQL サンプルを修正し、相互起動をイネーブルにすることで、Prime Network 内に相互起動ポイントを追加できます。