HP Business Service Management

Windows，Linux オペレーティング・システム用

ソフトウェア・バージョン : 9.20

BSM 拡張性ガイド

ドキュメント・リリース日 : 2012 年 8 月 (英語版)

ソフトウェア・リリース日 : 2012 年 8 月 (英語版)

ご注意

保証

HP 製品，またはサービスの保証は，当該製品，およびサービスに付随する明示的な保証文によってのみ規定されるものとします。ここでの記載で追加保証を意図するものは一切ありません。ここに含まれる技術的，編集上の誤り，または欠如について，HP はいかなる責任も負いません。

ここに記載する情報は，予告なしに変更されることがあります。

権利の制限

機密性のあるコンピュータ・ソフトウェアです。これらを所有，使用，または複製するには，HP からの有効な使用許諾が必要です。商用コンピュータ・ソフトウェア，コンピュータ・ソフトウェアに関する文書類，および商用アイテムの技術データは，FAR12.211 および 12.212 の規定に従い，ベンダーの標準商用ライセンスに基づいて米国政府に使用許諾が付与されます。

著作権について

商標について

Adobe® および Acrobat® は，Adobe Systems Incorporated の商標です。

AMD および AMD Arrow ロゴは，Advanced Micro Devices, Inc. の商標です。

Google™ および Google Maps™ は，Google Inc. の商標です。

Intel®，Itanium®，Pentium®，および Intel® Xeon® は，米国およびその他の国における Intel Corporation の商標です。

iPod は Apple Computer, Inc. の商標です。

Java は，Oracle Corporation およびその関連会社の登録商標です。

Microsoft®，Windows®，Windows NT®，Windows® XP，および Windows Vista® は，Microsoft Corporation の米国登録商標です。

Oracle は，Oracle Corporation およびその関連会社の登録商標です。

UNIX® は The Open Group の登録商標です。

謝辞

本製品には，Apache Software Foundation(http://www.apache.org/) (英語サイト) によって開発されたソフトウェアが含まれています。

本製品には，JDOM Project(http://www.jdom.org/) (英語サイト) によって開発されたソフトウェアが含まれています。

本製品には，MX4J Project(http://mx4j.sourceforge.net) (英語サイト) によって開発されたソフトウェアが含まれています。

ドキュメントの更新情報

このマニュアルの表紙には，以下の識別番号が記載されています。

- ソフトウェアのバージョン番号は，ソフトウェアのバージョンを示します。
- ドキュメント・リリース日は，ドキュメントが更新されるたびに変更されます。
- ソフトウェア・リリース日は，このバージョンのソフトウェアのリリース期日を表します。

最新の更新のチェック，またはご使用のドキュメントが最新版かどうかのご確認には，次のサイトをご利用ください。

http://h20230.www2.hp.com/selfsolve/manuals

このサイトを利用するには，HP Passport への登録とサインインが必要です。HP Passport ID の取得登録は，次の Web サイトから行なうことができます。

http://h20229.www2.hp.com/passport-registration.html(英語サイト)

または，HP Passport のログイン・ページの[**New users - please register**]リンクをクリックします。

適切な製品サポート・サービスをお申し込みいただいたお客様は，更新版または最新版をご入手いただけます。詳細は，HP の営業担当にお問い合わせください。

オンライン・ヘルプの PDF バージョン

このドキュメントはオンライン・ヘルプの PDF バージョンです。この PDF は，複数のトピックをヘルプ情報から簡単に印刷すること，またはオンライン・ヘルプを PDF 形式で読むため用意されています。

このドキュメントは最終更新です。2012 年 12 月 21 日

サポート

次の HP ソフトウェアのサポート Web サイトを参照してください。

http://support.openview.hp.com

HP ソフトウェアが提供する製品，サービス，サポートに関する詳細情報をご覧いただけます。

HP ソフトウェア・オンラインではセルフソルブ機能を提供しています。お客様の業務の管理に必要な対話型の技術支援ツールに素早く効率的にアクセスいただけます。HP ソフトウェア・サポート Web サイトのサポート範囲は次のとおりです。

- 関心のある技術情報の検索
- サポート・ケースとエンハンスメント要求の登録とトラッキング
- ソフトウェア・パッチのダウンロード
- サポート契約の管理
- HP サポート窓口の検索
- 利用可能なサービスに関する情報の閲覧
- 他のソフトウェア・カスタマとの意見交換
- ソフトウェア・トレーニングの検索と登録

一部を除き，サポートのご利用には，HP Passport ユーザとしてご登録の上，ログインしていただく必要があります。また，多くのサポートのご利用には，サポート契約が必要です。HP Passport ID を登録するには，以下の Web サイトにアクセスしてください。

http://h20229.www2.hp.com/passport-registration.html(英語サイト)

アクセス・レベルに関する詳細は，以下の Web サイトにアクセスしてください。

http://h20230.www2.hp.com/new_access_levels.jsp

目次

拡張性ガイドの概要

本書では，Business Service Management アプリケーションをカスタマイズする方法，それらのアプリケーションの機能を拡張する方法，および API を使用して操作を実行する方法について説明します。本書は，詳細設定や拡張機能を設定する必要がある管理者およびインテグレータを対象としています。BSM をセットアップする必要のある管理者は，本書より先に『BSM アプリケーション管理ガイド』を参照してください。

本書では，次のアプリケーションで作業する手順について説明します。

- **サービス状況**。詳細については，23ページ「サービス状況ルール API」および50ページ「サービス状況の外部 API」を参照してください。
- **サービス・レベル管理**。詳細については，61ページ「SLM 外部 API」，75ページ「SLM ルール API」，および120ページ「SLM Web サービス API」を参照してください。
- **エンド・ユーザ管理**。詳細については，127ページ「EUM Admin の Open API」を参照してください。
- **オペレーション管理**。詳細については，129ページ「オペレーション管理の拡張」を参照してください。
- **レポート**。詳細については，391ページ「汎用レポート・エンジン API」を参照してください。

第1部分

サービス状況

第1章

サービス状況ルール API

注: BSM バージョン 9.00 以降では，インジケータのステータスおよび値をサンプルに基づいて計算するルール(28ページ「API サンプル・ルール」 および30ページ「API 継続時間ベースのサンプル・ルール」) を使用して，メトリックベースの状況インジケータ(HI) を計算します。

ルール API のドキュメントには，さまざまな KPI の計算方法が記載されています。**BSM バージョン 9.00 以降では，サンプル・ベース値の計算時にこれらのメソッドがメトリックベースの HI の計算に使用されます。**

本章では，ルール API を使用して新しいビジネス・ルールを作成する方法について説明します。ビジネス・ルールは，主要管理指標 (KPI) の計算に使用されます。KPI には，その KPI の計算方法を定義するビジネス・ルールを関連付ける必要があります。標準設定のサービス状況ルールについては，『BSM アプリケーション管理ガイド』の「サービス状況の計算ルールのリスト」の項に記載されています。

ルール API を使用してルールを作成することをお勧めします。ルール API により，Groovy スクリプト言語と Groovy Runtime Environment 1.7.3 以前を使用してルールを作成できます。ルール API を使用するには，Groovy と Java に関する知識を持ち，BSM の管理およびアプリケーションに慣れている必要があります。

ルール API クラスについては，『HP Rules API Reference』に Javadoc 形式で記載されています。これらのファイルは次のフォルダにあります。
\\< ゲートウェイ・サーバのルート・ディレクトリ>\AppServer\webapps\site.war\amdocs\eng\doc_lib\Service_Health\Rules_API\index.html

サービス状況 API ルール

サービス状況 API ルールのタイプは次のとおりです。

- **グループと兄弟ルール**:このタイプのルールでは，元のサンプル・データではなく，ほかの KPI から受け取ったデータに基づいて KPI を計算します。詳細は，25ページ「API グループと兄弟ルール」を参照してください。
- **サンプル・ルール**:このタイプのルールでは，サンプル・フィールドから取得した元のデータに基づいて KPI を計算します。計算に含まれるサンプルの数は，サンプル・ルール・パラメータの最大数によって制限されます。詳細は，28ページ「API サンプル・ルール」を参照してください。
- **継続時間ベースのサンプル・ルール**:このタイプのルールでは，サンプル・フィールドから取得した元のデータに基づいて KPI を計算します。継続時間のパラメータによって，計算に含められるサンプルが定義されます。詳細は，30ページ「API 継続時間ベースのサンプル・ルール」を参照してください。

API ルールの作成

ルール API でルールを作成するには，次の方法があります。

- [CI インジケータ]タブを使用して，特定の KPI のルールを作成する。
- テキスト・ファイルを使用して，複数の KPI の新しいルールを作成する。

- ルール・リポジトリ内の API ルールの複製を使用して，新しいルールを作成する。

これらの方法については，32ページ「ルール API を使用したルールの作成」を参照してください。

ツールチップおよびログ・ファイル

ルール API を使用する際にツールチップに KPI 情報を表示する方法については，38ページ「ツールチップ・エントリの使用方法」を参照してください。

39ページ「ルール API コードからログ・ファイルに書き込む方法」で説明するように，ルール API コードからログ・ファイルに書き込むことができます。

API グループと兄弟ルール

API グループと兄弟ルールでは，元のサンプル・データではなく，ほかのインジケータから受け取ったデータに基づいて KPI を計算します。データは，子 CI の KPI や，同じ CI に関連付けられているほかの KPI または HI から取得されます。

注: 兄弟ルールを作成する場合は，KPI の[計算順序]フィールドで定義されたように，KPI がその兄弟 KPI の後で計算されるようにする必要があります。詳細については，『BSM アプリケーション管理ガイド』の「KPI リポジトリ・ページ」を参照してください。

グループと兄弟ルールのメソッドおよびフィールド

グループと兄弟ルールは，次のガイドラインに従って，ルール API インタフェース **GroupAndSiblingCalculator** を実装します。

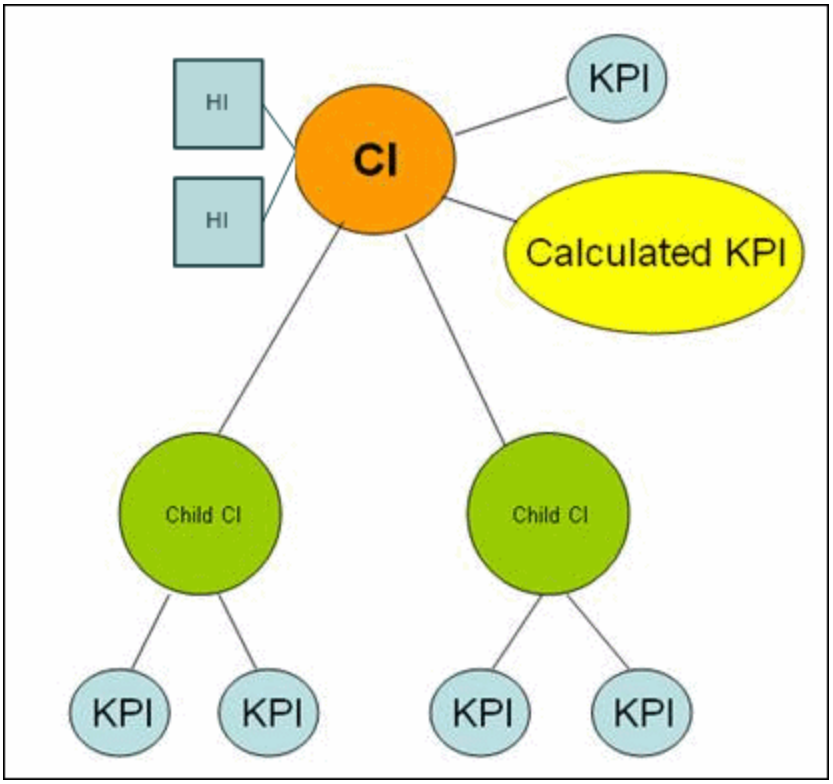
- このインタフェースでは，唯一のメソッドは **calculateKPI** です。メソッド・シグネチャは次のとおりです。

```
public void calculateKPI(CI ci, KPI kpi)
```

- **calculateKPI** メソッドには，現在の CI を表す **ci** パラメータと，API ルールで値が計算される KPI を表す **kpi** パラメータがあります。
  - **ci** パラメータのタイプは[**CI**]で，子 CI の KPI または兄弟 KPI，あるいは CI の HI へのアクセサとして使用されます。
  - **kpi** パラメータ・タイプは **KPI** で，計算結果の設定に使用されます。

次の図では，計算された KPI が兄弟 KPI または子 KPI に基づいて計算され，**kpi** パラメータによって表されます。

計算された KPI が割り当てられる CI は **ci** パラメータによって表され，ほかの KPI または HI へのアクセサです。



ルール API クラスについては,『HP Rules API Reference』に Javadoc 形 式 で記 載 されています。これらのファイルは次 のフォルダにあります。
\\<ゲートウェイ・サーバのルート・ディレクトリ>\AppServer\webapps\site.war\amdocs\eng\doc_lib\Service_Health\Rules_API\index.html

グループと兄 弟 ルールの詳 細 な例 については, 45ページ「例 - API グループと兄 弟 ルール」を参 照 してください。

API ルールは, 32ページ「ルール API を使 用 したルールの作 成 」で説 明 するように, サービス状 況 の [CI インジケータ] タブまたはルール・リポジトリ, あるいはテキスト・ファイル・テンプレートを使 用 して定 義 できます。

[CI インジケータ] タブまたはルール・リポジトリでの, グループと兄 弟 ルールの定 義

[CI インジケータ] タブまたはルール・リポジトリを使 用 してグループと兄 弟 ルールを定 義 するには, **calculateKPI** メソッド実 装 を [**KPI 計 算 スクリプト**] 領 域 に入 力 します。

calculateKPI メソッドのパラメータ **ci** と **kpi** がこのスクリプトで使 用 するために利 用 可 能 です。

詳 しい手 順 については, 33ページ「[CI インジケータ] タブでの API ルールの定 義 方 法 」または37ページ「ルール・リポジトリでの API ルールの定 義 方 法 」を参 照 してください。

[CI インジケータ] タブでの特 定 の子 KPI へのアクセス

[CI インジケータ] タブで特 定 の KPI のグループ・ルールを作 成 する際 に, 特 定 の子 KPI にアクセスできるように, コードを簡 素 化 するメカニズムが API に含 まれています。KPI 計 算 スクリプトを定 義 する場 合 は, **"<CI 名 >"."<KPI 名 >"** という形 式 で入 力 します。

この例 については, 45ページ「例 - API グループと兄 弟 ルール」の46ページ「例 - 特 定 の子 CI グループ・ルール」を参 照 してください。

テキスト・ファイルを使用した，グループと兄弟ルールの定義

テキスト・ファイルを使用してグループと兄弟ルールを定義するには，34ページ「テキスト・ファイル・ベースの API ルールの作成方法」で説明するように，**DashboardGroupAndSiblingTemplate.groovy** テンプレートを使用します。

テキスト・ファイルに **calculateKPI** メソッド本体を入力します。

API サンプル・ルール

サンプル・ルールでは，サンプル・フィールドから取得した元のデータに基づいて KPI を計算します。計算に含まれるサンプルの数は，サンプル・パラメータの最大数によって制限されます。

サンプル・ルールのメソッドおよびフィールド

サンプル・ルールでは，次のガイドラインを使用してルール API インタフェース **LeafCalculator** が実装されます。

- このインタフェースでは，唯一のメソッドは **calculateKPI** です。メソッド・シグネチャは次のとおりです。

```
public void calculateKPI(CI ci, KPI kpi, List<サンプル> samples)
```

- **calculateKPI** メソッドには，パラメータ **ci**，**kpi**，**samples**が含まれます。これらは現在の CI，その値がルールによって計算される KPI，**Maximum number of samples** パラメータに基づいてルール計算で使用されるサンプルを表します(このパラメータ値が 1 の場合は，このフィールドに 1 つのサンプルがリストされます) 。
  - **kpi** パラメータ・タイプは **KPI** で，計算結果の設定に使用されます。
  - **samples** パラメータは **Sample** オブジェクトの **List** で，サンプル・フィールド値を保持します。
- また，ルールでは **Sample** オブジェクトによって保持されるサンプル・フィールドを定義するための **sampleFields** フィールドを設定する必要があります。これらの値は，ルールによって使用される値です。

サンプル・ルールの詳細な例については，42ページ「例 - API サンプル・ルール」を参照してください。

API ルールは，32ページ「ルール API を使用したルールの作成」で説明するように，サービス状況の [CI インジケータ] タブまたはルール・リポジトリ，あるいはテキスト・ファイル・テンプレートを使用して定義できます。

ルール API クラスについては，『HP Rules API Reference』に Javadoc 形式で記載されています。これらのファイルは次のフォルダにあります。
\\<ゲートウェイ・サーバのルート・ディレクトリ>\AppServer\webapps\site.war\amdocs\eng\doc_lib\Service_Health\Rules_API\index.html

[CI インジケータ] タブまたはルール・リポジトリでの，サンプル・ルールの定義

[CI インジケータ] タブまたはルール・リポジトリを使用してサンプル・ルールを定義するには，各フィールドに次のように入力します。

- **サンプル・フィールド** :[**サンプル**] オブジェクトに保持されるサンプル・フィールドをリストします。サンプル名はカンマで区切って入力します(たとえば，`"u_iStatus", "dResponseTime"`) 。
- **KPI 計算スクリプト** :**calculateKPI** メソッド実装を入力します。メソッド・シグネチャを入力しないでください。**calculateKPI** メソッドのパラメータ **ci**，**kpi**，**samples** がこのスクリプトで利用可能です。
- **サンプルの最大数** :標準設定では，最新のサンプルが含まれます(標準設定 = 1) 。このフィールドを使用して，この設定を変更できます。

詳しい手順については，33ページ「[CI インジケータ] タブでの API ルールの定義方法」または37ページ「ルール・リポジトリでの API ルールの定義方法」を参照してください。

テキスト・ファイルを使用した，サンプル・ルールの定義

テキスト・ファイル・テンプレートを使用してサンプル・ルールを定義するには，34ページ「テキスト・ファイル・ベースの API ルールの作成方法」で説明するように，
［**DashboardSampleRuleTemplate.groovy**］テンプレート・ファイルを使用します。

テキスト・ファイルに **calculateKPI** メソッド本体を入力し，**sampleFields** フィールドを定義します。

API 継続時間ベースのサンプル・ルール

継続時間ベースのサンプル・ルールでは，サンプル・フィールドから取得した元のデータに基づいて KPI を計算します。継続時間のルール・パラメータによって，計算に含められるサンプルが定義されます。たとえば，継続時間が 15 分間と定義されている場合は，最後の 15 分間に収集されたすべてのサンプルが計算に含められます。

継続時間ベースのサンプル・ルールのメソッドおよびフィールド

継続時間ベースのサンプル・ルールでは，次のガイドラインを使用してルール API インタフェース **LeafCalculator** が実装されます。

- このインタフェースでは，唯一のメソッドは **calculateKPI** です。メソッド・シグネチャは次のとおりです。

```
public void calculateKPI(CI ci, KPI kpi, List<サンプル> samples)
```

- **calculateKPI** メソッドには，パラメータ **ci**，**kpi**，**samples**が含まれます。これらは現在の CI，その値がルールによって計算される KPI，ルール計算で使用されるサンプルのリストを表します。
  - **kpi** パラメータ・タイプは **KPI** で，計算結果の設定に使用されます。
  - **samples** パラメータは **Sample** オブジェクトの **List** で，サンプル・フィールド値を保持します。
- また，ルールでは **Sample** オブジェクトによって保持されるサンプル・フィールドを定義するための **sampleFields** フィールドを設定する必要があります。これらの値は，ルールによって使用される値です。

このルールの詳細な例については，42ページ「例 - API サンプル・ルール」を参照してください。

API ルールは，32ページ「ルール API を使用したルールの作成」で説明するように，サービス状況の [CI インジケータ] タブ，テキスト・ファイル，またはルール・リポジトリで定義できます。

ルール API クラスについては，『HP Rules API Reference』に Javadoc 形式で記載されています。これらのファイルは次のフォルダにあります。
\\<ゲートウェイ・サーバのルート・ディレクトリ>\AppServer\webapps\site.war\amdocs\eng\doc_lib\Service_Health\Rules_API\index.html

[CI インジケータ] タブまたはルール・リポジトリでの，継続時間ベースのサンプル・ルールの定義

[CI インジケータ] タブまたはルール・リポジトリを使用して継続時間ベースのサンプル・ルールを定義するには，各フィールドに次のように入力します。

- **サンプル・フィールド**：[**サンプル**] オブジェクトに保持されるサンプル・フィールドをリストします。サンプル名はカンマで区切って入力します(たとえば，`"u_iStatus", "dResponseTime"`)。
- **KPI 計算スクリプト**：メソッド実装を入力します。メソッド・シグネチャは入力しません。**calculateKPI** メソッドのパラメータ **ci**，**kpi**，**samples** がこのスクリプトで利用可能です。
- [**データ タイムアウトなし**] および [**継続時間**]：(任意)「ルール・パラメータのリスト」で説明するように，タイムアウト期間と継続時間のパラメータを定義できます。

詳しい手順については，33ページ「[CI インジケータ] タブでの API ルールの定義方法」または37ページ「ルール・リポジトリでの API ルールの定義方法」を参照してください。

テキスト・ファイルを使用した，継続時間ベースのサンプル・ルールの定義

テキスト・ファイル・テンプレートを使用して継続時間ベースのサンプル・ルールを定義するには，34ページ「テキスト・ファイル・ベースの API ルールの作成方法」で説明するように，**DashboardDurationBasedSampleRuleTemplate.groovy** テンプレート・ファイルを使用します。

テキスト・ファイルに **calculateKPI** メソッド本体を入力し，**sampleFields** フィールドを定義します。

ルール API を使用したルールの作成

次の項で説明するように，ルール API を使用してルールを作成するには，いくつかの方法があります。

[CI インジケータ]タブを使用した，特定の KPI のルールの定義

サービス状況 KPI にはそれぞれ，次の適用可能な API ルールがあります。API グループと兄弟ルール，API サンプル・ルール，または API 継続時間ベースのサンプル・ルール。[CI インジケータ]タブで，いずれかの API ルールを KPI に割り当て，計算スクリプト(およびほかのルール詳細) を入力して，その KPI のルール・ロジックを定義できます。

その後は，[CI インジケータ]タブ内のルール詳細をいつでも編集して，その KPI のルール・ロジックを変更できます。

詳細は，33ページ「[CI インジケータ]タブでの API ルールの定義方法」を参照してください。

テキスト・ファイルを使用した，ルールの作成

API ルール(API グループと兄弟ルール，API サンプル・ルール，または API 継続時間ベースのサンプル・ルール) のそれぞれについて，対応するテンプレート・ファイルが **<データ処理サーバのルート・ディレクトリ>\BLE\rules\groovy\templates** ディレクトリ内にあります。いずれかのテンプレート・ファイルを使用して，新しいルールを定義するテキスト・ファイルを作成できます。作成したルールをルール・リポジトリに追加すると，標準で含まれているルールと同じように適用できます。

API コードの表示または変更はサービス状況内では実行できず，テキスト・ファイル内でのみ可能です。テキスト・ファイル内のコードを変更すると，これらの変更はサービス状況ルールを再ロードした後にルールが割り当てられているすべてのインスタンスに適用されます。

詳細は，34ページ「テキスト・ファイル・ベースの API ルールの作成方法」を参照してください。

ルール・リポジトリ内でのルールの定義

ルール・リポジトリには，次の 3 つの API ルールがあります。API グループと兄弟ルール，API サンプル・ルール，または API 継続時間ベースのサンプル・ルール。ルール・リポジトリを使用して，API ルールを複製し，計算スクリプト(およびほかのルール詳細) を入力して，ルール・ロジックを定義できます。

ルールが KPI に適用された後は，いつでも[CI インジケータ]タブでルールの詳細を編集して，特定の KPI のルール・ロジックを変更できます。

詳細は，37ページ「ルール・リポジトリでの API ルールの定義方法」を参照してください。

[CI インジケータ]タブでの API ルールの定義方法

KPI のそれぞれに，3 つの適用可能な API ルールがあります。[CI インジケータ]タブ内で，いずれかの API ルールを KPI に割り当て，計算スクリプト(およびほかのルール詳細)を入力して，その KPI のルール・ロジックを定義します。

1. **KPI への API ルールの割り当て**

   CI に割り当てられている特定の KPI に対し API ルールを割り当てるには，[**管理**]＞[**サービス状況**]＞[**CI インジケータ**]を選択します。[**新規 KPI**]を選択して新しい KPI を CI に割り当てるか，[**KPI の編集**]を選択して既存の KPI を変更します。この処理の詳細については，『BSM アプリケーション管理ガイド』の「CI への KPI および HI の割り当て方法」を参照してください。

   適用可能なビジネス・ルールのリストで，次の API ルールを 1 つ選択します。API グループと兄弟ルール，API サンプル・ルール，または API 継続時間ベースのサンプル・ルール。ルール・タイプの説明については，23ページ「サービス状況ルール API」を参照してください。

2. **KPI のルール・ロジックの定義**

   作成しているルールのタイプに応じて，次の説明に従って，ルールのメソッドとフィールドを定義します。

   - 25ページ「API グループと兄弟ルール」
   - 28ページ「API サンプル・ルール」
   - 30ページ「API 継続時間ベースのサンプル・ルール」

テキスト・ファイル・ベースの API ルールの作成方法

3 つの API ルールに対応する 3 つのルール・テンプレート・ファイルがあります。各テンプレートによりルールのインタフェースが実装されます。

いずれかのテンプレートを使用して新しいルールを定義するテキスト・ファイルを作成し，その新しいルールをビジネス・ルール・リポジトリに追加します。追加したルールは，標準設定で含まれているルールと同じように適用可能になります。

API コードの表示または変更はサービス状況 内では実行できず，テキスト・ファイル内でのみ可能です。テキスト・ファイル内のコードを変更すると，これらの変更はサービス状況 ルールを再ロードした後にルールが割り当てられているすべてのインスタンスに適用されます。

1. ## ルールのテキスト・ファイルの作成

   作成するルールのタイプに基づいて，**<データ処理サーバのルート・ディレクトリ>\BLE\rules\groovy\templates** ディレクトリに置かれているテンプレート・ファイルのいずれかをコピーし，その名前を変更します。

   テンプレートのコピー内で，次の説明に従って，ルールのメソッドとフィールドを定義します。

   - 25ページ「API グループと兄弟ルール」
   - 28ページ「API サンプル・ルール」
   - 30ページ「API 継続時間ベースのサンプル・ルール」

   ファイルを **<データ処理サーバのルート・ディレクトリ>\BLE\rules\groovy\rules** ディレクトリに保存します。

   ここで，テキスト・ファイルのルール・ロジックを使用するルールを，ルール・リポジトリに追加する必要があります。

2. ## ルール・リポジトリへのルールの追加

   a. [**管理**]＞[**サービス状況**]＞[**リポジトリ**]＞[**ビジネス ルール**]＞[**新規ルール**]を選択します。ルール追加の詳細については，『BSM アプリケーション管理ガイド』の「リポジトリのビジネス・ルール・テンプレートのカスタマイズ方法」を参照してください。

   b. [**名前**]フィールドに，作成するルールの名前を入力します(必須)。

   c. [**クラス名**]フィールドに，「**groovy:**<ファイル名>」と入力します。ファイル名は **<データ処理サーバのルート・ディレクトリ>\BLE\rules\groovy\rules** ディレクトリの名前と同一(大文字と小文字を区別)にする必要があります。

   d. API ルール・タイプに応じて，ルール・パラメータを作成します。

      - [**ルール パラメータ**]領域で，[**新規ルール パラメータ**]をクリックします。
      - API サンプル・ルールの場合：

        [**名前**]フィールドに「**サンプルの最大数**」と入力します。[**タイプ**]フィールドで[**Integer**]を選択します。[**標準設定値**]フィールドに「**1**」と入力します。

        [**OK**]をクリックして保存します。

      - API 継続時間ベースのサンプル・ルールの場合：

[**名前**]フィールドに「**継続時間**」と入力します。[**タイプ**]フィールドで[**Long**]を選択します。[**標準設定値**]フィールドに「**990**」と入力します。

[**OK**]をクリックして保存します。

これらの手順を繰り返して，[**データ タイムアウトなし**]ルール・パラメータ(タイプ:Long，標準設定値 = 990)を追加します。

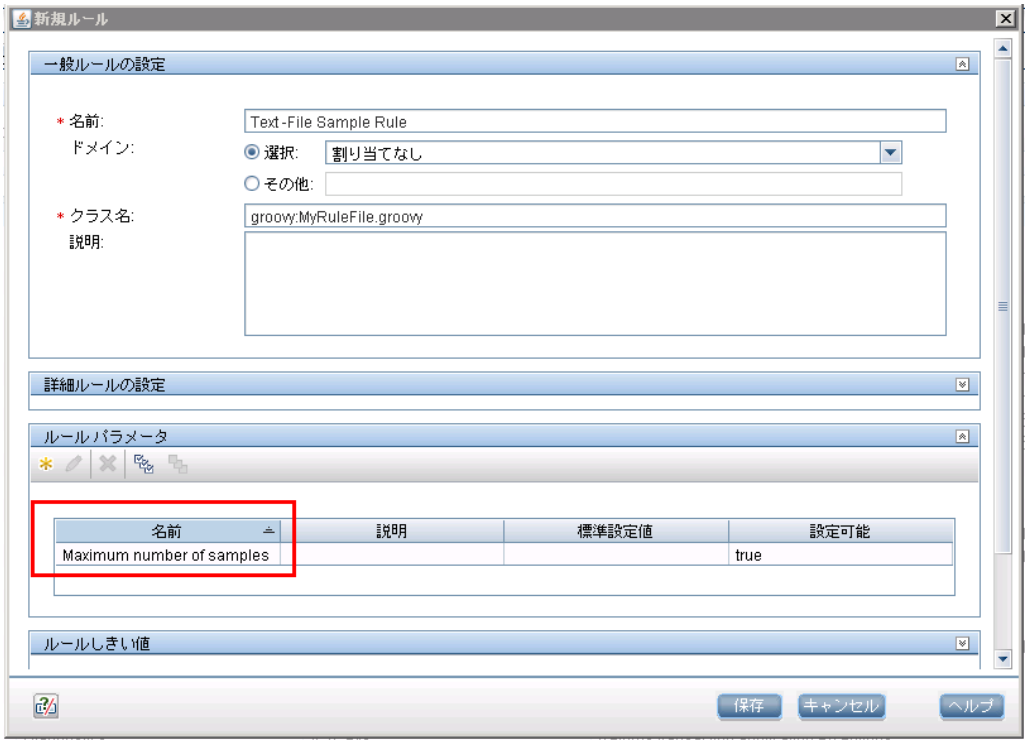
e. しきい値パラメータ(致命的，重大，軽微，警告，情報，演算子)を作成します(グループと兄弟ルールを定義している場合は，この手順を省略します。グループと兄弟ルールの場合はルール・コードによってステータスが計算されるため，しきい値はありません)。

- **しきい値パラメータ**領域で，[**新規作成**]をクリックします。
- [**名前**]フィールドに「**critical**」と入力します。[**タイプ**]フィールドで[**Float**]を選択します。

  [**演算子**]パラメータを定義する際には，[**タイプ**]フィールドで[**String**]を選択します。
- [**OK**]をクリックして保存します。

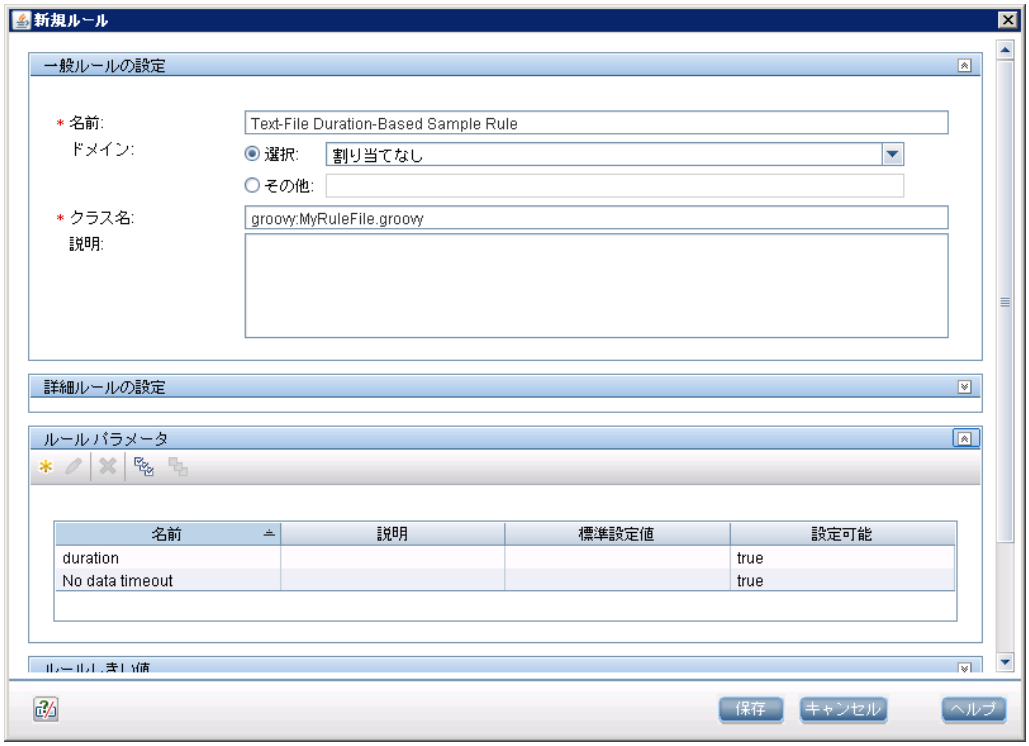
ほかのしきい値パラメータ(重大，軽微，警告，情報，演算子)について，これらの手順を繰り返します。

次の図は，ルール・パラメータが追加された後のサンプル・ルールを示しています。



次の図は，ルール・パラメータが追加された後の継続時間ベースのサンプル・ルールを示して

います。



3. ### KPI の適用可能なルール・リストへのルールの追加

   関連する KPI にすでにアタッチされている適用可能なルールのリストに，新しいルールを追加します。詳細については，『BSM アプリケーション管理ガイド』の「[新規 KPI]/[KPI の編集]ダイアログ・ボックス」の「[メイン設定]領域」に記載されている「適用可能なルールのパラメータ」を参照してください。

4. ### 新しいツールチップへのツールチップ・パラメータの追加

   この手順を実行してルールを作成した場合，対応するツールチップにはツールチップ・パラメータが設定されていません。新しいツールチップにツールチップ・パラメータを追加する方法については，38ページ「ツールチップ・エントリの使用方法」を参照してください。

5. ### テキスト・ファイル編集後のルールの再ロード

   ルールを作成した後でテキスト・ファイルに変更を加えた場合は，次の手順を実行して，変更内容を適用します。

   a. ブラウザで，JMX ポート <29810 ＋ workerID>(たとえば，workker_1 の場合は 29811) にアクセスします。

   b. **BSM-Platform** 内で，**MarbleWorker** というサービスを選択し，**reloadRules** メソッドを呼び出します。このメソッドは，このワーカによってサービスが提供されるすべてのカスタマに適用されます。

ルール・リポジトリでの API ルールの定義方法

ルール・リポジトリ内で，複数の KPI に適用できる API ルールを作成します。これを行うには，3 つの API ルールのいずれかを複製し，特定のルール・パラメータの標準設定のルール値を設定します。ルールが KPI に適用された後は，いつでも[CI インジケータ]タブでそのスクリプトを編集して，その KPI のルール・ロジックを変更できます。

1. **API ルールの複製**

   [**管理**]＞[**サービス状況**]＞[**リポジトリ**]＞[**ビジネス ルール**]を選択します。[ビジネス ルール リポジトリ]ページで，次のルールのいずれかを複製します。API グループと兄弟ルール，API サンプル・ルール，または API 継続時間ベースのサンプル・ルール。

   ルールを複製する方法の詳細については，『BSM アプリケーション管理ガイド』の「リポジトリのビジネス・ルール・テンプレートのカスタマイズ方法」を参照してください。

2. **ルールの詳細の編集**

   a. 編集する新しいルールを開きます。

   b. [**名前**]フィールドで，複製したルールの名前を変更します。

   c. [**KPI 計算スクリプト**]ルール・パラメータを編集します。[**標準設定値**]フィールドに，ルール計算スクリプトを入力します。入力するコードがこのルールの標準設定のコードになり，このルールが割り当てられるすべての KPI の[CI インジケータ]タブに表示されます(ほかのフィールドは変更しないでください) 。

   d. サンプル・ルールまたは継続時間ベースのサンプル・ルールを作成している場合は，[**サンプル フィールド**]ルール・パラメータを編集します。入力するサンプル・フィールドがこのルールの標準設定のサンプル・フィールドになり，このルールが割り当てられるすべての KPI の[CI インジケータ]タブに表示されます(ほかのフィールドは変更しないでください) 。

      これらのルール・パラメータの詳細については，作成するルールのタイプに応じて，次の項を参照してください。

      - 25ページ「API グループと兄弟ルール」
      - 28ページ「API サンプル・ルール」
      - 30ページ「API 継続時間ベースのサンプル・ルール」

      ルール API クラスについては，『HP Rules API Reference』に Javadoc 形式で記載されています。これらのファイルは次のフォルダにあります。
      \\<ゲートウェイ・サーバのルート・ディレクトリ>\AppServer\webapps\site.war\amdocs\eng\doc_lib\Service_Health\Rules_API\index.html

3. **KPI の適用可能なルール・リストへのルールの追加**

   関連する KPI にすでにアタッチされている適用可能なルールのリストに，新しいルールを追加します。詳細については，『BSM アプリケーション管理ガイド』の「[新規 KPI]/[KPI の編集]ダイアログ・ボックス」に記載されている「適用可能なルール・パラメータ」を参照してください。

ツールチップ・エントリの使用方法

次の項では，ツールチップ・エントリを使用して，ルール API で計算された情報を表示する方法について説明します。

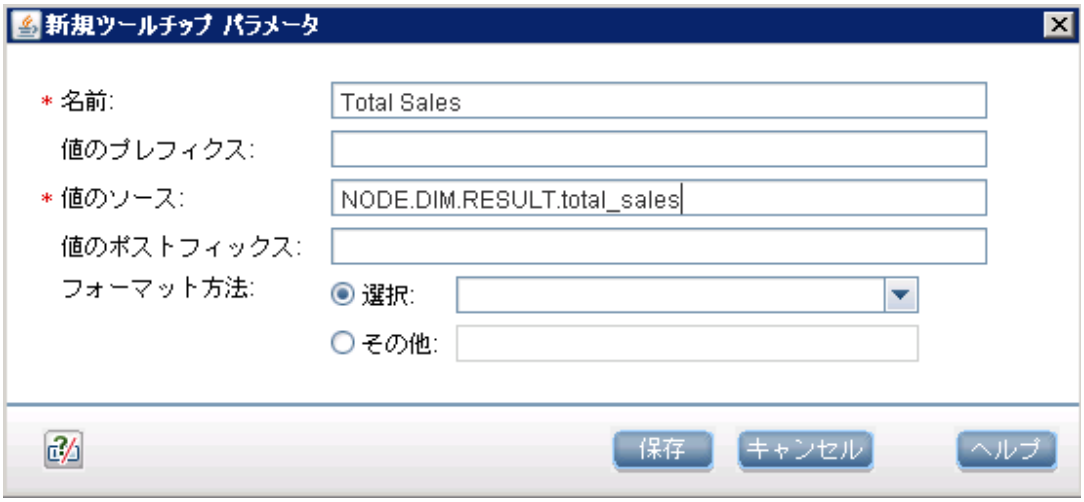
1. [**管理**]＞[**サービス状況**]＞[**リポジトリ**]＞[**ビジネス ルール**]を選択します。ルール・リポジトリ・ページで，新しいルールに必要なツールチップ・エントリを追加します。次の表は，一般的なツールチップ・エントリと，それらに対応する値のソースおよび書式設定方式を示しています。

ツールチップ・パラメータ	値のソース	書式設定方式
ビジネス・ルール	NODE.DIM.RULE.ID_CUST	ruleIDtoString
CI 名	NODE.PROPS.BamNodeNameKey	toLowerCase
ステータス保持開始	NODE.DIM.RESULT.LastStatusChange	returnDateAsString
ステータス	NODE.DIM.RESULT.Status	getStatusString
値	NODE.DIM.RESULT.Value	returnNumOfDigitAfterPoint

詳細については，『BSM アプリケーション管理ガイド』の「[新規ツールチップ パラメータ]/[ツールチップ パラメータの編集]ダイアログ・ボックス」を参照してください。

2. **kpi.setTooltip** メソッドを使用した場合は，前述のように，ルール・リポジトリで対応するツールチップ・エントリを設定する必要があります。[**値のソース**]フィールドに，コードで使用されているのと正確に一致するツールチップ・エントリ名を入力し，[**フォーマット方法**]フィールドは空のままにします。

   たとえば，コードにメソッド呼び出し **kpi.setTooltip("total_sales", value)** が含まれている場合は，[**値のソース**]フィールドに「**NODE.DIM.RESULT.total_sales**」と入力します。



ルール API コードからログ・ファイルに書き込む方法

API ルールで，**logger** オブジェクトを使用してルール・メソッドからログ・ファイルに書き込むことができます。ログには，debug，info，warn，error，fatal という 5 つのレベルがあります。これらのレベルそれぞれで，特定の logger メソッドを使用します。

標準設定では，重大度が「error」と「fatal」のログ・メソッド呼び出しだけが，ログ・ファイルに書き込まれます。この設定は，ログ設定ファイル内で変更できます。

ルール API を使用してログ・ファイルに書き込むには，次の手順を実行します。

1. ルール・メソッド内で，次のいずれかのメソッドを実装します(メソッドは重大度の順にリストされています) 。
   - **logger.debug("<API ルール名 > :log message");**
   - **logger.info("<API ルール名 > :log message");**
   - **logger.warn("<API ルール名 > :log message");**
   - **logger.error("<API ルール名 > :log message");**
   - **logger.fatal("<API ルール名 > :log message");**

   API ルールの名前をログ・メッセージ内に入力して，各ログ・メッセージをそのソース・ルールで識別します。
2. ルール API のログ・ファイルは **<データ処理サーバのルート・ディレクトリ> \HPBSM\log\marble_worker_<worker#>\RulesAPI** ディレクトリにあります。

ルール・タイプに応じて，次のいずれかのファイルを開いて，ログ・メッセージを表示します。

- **groupAndSiblingRule.log**(API グループと兄弟ルールの場合)
- **sampleRule.log**(API サンプル・ルールの場合)
- **durationBasedSampleRule.log**(API 継続時間ベースのサンプル・ルールの場合)

ログ・ファイルに書き込まれる重大度レベルを変更するには，次の手順を実行します。

1. 標準設定では，重大度が「error」と「fatal」のログ・メソッド呼び出しだけが，ログ・ファイルに書き込まれます。この設定を変更するには，**<データ処理サーバのルート・ディレクトリ> HPBSM\conf\core\Tools\log4j\marble_worker\dashboard_rules.properties** にあるログ設定ファイルを開きます。
2. ルール・タイプに対応する行で，文字列 **${loglevel}** を，ログする重大度レベル (DEBUG，INFO，WARN，ERROR，FATAL) に置き換えます。ルール・タイプに応じて，次のいずれかの行を編集します。
   - グループと兄弟ルールの場合
     :log4j.category.com.mercury.am.rules.dashboard.blDashboardRules.simplifiedRule.groupAndSiblingRule.DashboardGroupAndSiblingRule = **${loglevel}**, bam.app.rules.api.group.appender
   - サンプル・ルールの場合
     :log4j.category.com.mercury.am.rules.dashboard.blDashboardRules.simplifiedRule.leaf.DashboardSimplifiedSampleBasedRule = **${loglevel}**,

bam.app.rules.api.leafsample.appender

- 継 続 時 間 ベースのサンプル・ルールの場 合
  :log4j.category.com.mercury.am.rules.dashboard.blDashboardRules.
  simplifiedRule.leaf.DashboardSimplifiedTimeBasedRule = **${loglevel}**,
  bam.app.rules.api.leafduration.appender

ルール API 計算に CI プロパティを含める方法

CI クラスの **getPropertyValue** メソッドとKPI クラスの **getCiProperty** メソッドを API ルール内で使 用し，CI プロパティを含 められます。このメソッド では，修 飾 子 が次 のいずれかの CI プロパティのみにアクセスできます。

- **BLE_ATTRIBUTE** - SLM とサービス状 況
- **BLE_ONLINE_ATTRIBUTE** - サービス状 況 のみ

この属 性 を CI クラスに追 加 するには，クラスをエクスポートしてから定 義 を編 集し，サーバにインポートし直 す必 要 があります。エクスポートしたクラスを開 いて編 集 するには，次 の xml を必 要 な属 性 に追 加 します。

```
<Attribute name="<attribute-name>" type="double" display-name="<attribute-display-name>">
  <Attribute-Qualifiers>
    <Attribute-Qualifier name="BLE_ATTRIBUTE"/>
  </Attribute-Qualifiers>
</Attribute>
```

例 - API サンプル・ルール

この項では，API サンプル・ルールの例を示します。次の例について説明します。

- 42ページ「例 - 平均可用性ルール」
- 42ページ「例 - 平均パフォーマンス・ルール」
- 43ページ「例 - ルール・パラメータ・フィルタを使用した平均パフォーマンス・ルール」

例 - 平均可用性ルール

次のルールでは，u_iStatus サンプル・フィールドに基づいて，サンプルの平均可用性を計算します。

ルールのロジックは(使用可能なサンプル数/サンプル合計数)* 100 です。

```
// This rule uses the u_iStatus sample field. def sampleFields =
["u_iStatus"];
public void calculateKPI(CI ci, KPI kpi, List<Sample> samples) {
    // Keep total number of samples for this calculation cycle.
    def totalSamples = samples.size();
    // Create a variable to count available samples.
    def availableSamples = 0;
    /**     * Go over the given samples.If a sample's u_iStatus is
equal to 0,     * the sample is considered available.
    */     samples.each {Sample currentSample->     if
(currentSample.u_iStatus == 0) {         // Increase the count of
available samples.
        availableSamples++;
        }     }     if (totalSamples > 0) {         // Set KPI
value, converted to percentage.     kpi.setValue
((availableSamples/totalSamples)*100.0);     } }
```

例 - 平均パフォーマンス・ルール

次のルールでは，dResponseTime サンプル・フィールドと u_iStatus サンプル・フィールドに基づいて，平均パフォーマンスを秒数で計算します。

u_iStatus の値が 0 のサンプル(使用可能なサンプル)だけが，計算に使用されます。ルールのロジックは，「sum(dResponseTime)/使用可能なサンプル数」です。

```
// This rule uses the u_iStatus and dResponseTime sample field. def
sampleFields = ["u_iStatus", "dResponseTime"]; public void
calculateKPI(CI ci, KPI kpi, List<Sample> samples) {     // Create
a variable to count available samples.     def availableSamples =
0;     // Create a variable to sum response times of available
samples.     def totalResponseTime = 0;     /**      * Go over the
```

```
given samples.If a sample's u_iStatus is equal to 0,      * the
sample is considered available.     */     samples.each {Sample
currentSample ->         if (currentSample.u_iStatus == 0) {
            // Increase the count of available samples.
            availableSamples++;
            // Add the current sample's dResponseTime value to
totalResponseTime.
            totalResponseTime += currentSample.dResponseTime
        }     }     if (availableSamples > 0) {         // Set KPI
value, converted to percentage.
        kpi.setValue((totalResponseTime / availableSamples))     }
}
```

例 - ルール・パラメータ・フィルタを使用した平均パフォーマンス・ルール

次のルールでは，dResponseTime サンプル・フィールドと u_iStatus サンプル・フィールドに基づいて，平均パフォーマンスを秒数で計算します。

u_iStatus の値が 0 のサンプル(使用可能なサンプル) だけが，計算に使用されます。

このルールではオプションのルール・パラメータ，Response time limit を使用します。このルール・パラメータの値がサービス状況管理で設定されている場合，このルール・パラメータの値より dResponseTime の値の方が大きいサンプルは，計算に使用されません。

注: 同じ名前のルール・パラメータが，ルール・リポジトリでこのルール用に設定されている必要があります。詳細については，『BSM アプリケーション管理ガイド』の「[新規ルール パラメータ]/[ルール パラメータの編集]または[新規ルールしきい値]/[ルールしきい値の編集]」を参照してください。

ルールのロジックは，「sum(dResponseTime)/使用可能なサンプル数」です。

```
/ This rule use the u_iStatus and dResponseTime sample fields. def
sampleFields = ["u_iStatus", "dResponseTime"]; public void
calculateKPI(CI ci, KPI kpi, List<Sample> samples) {     // Create
a variable to count available samples.     def availableSamples =
0;     // Create a variable to sum response times of available
samples.     def totalResponseTime = 0;     /**      * Get the
value of the rule parameter named "Response time limit"      * from
the KPI, as defined for the KPI in サービス状況 Admin.     * This rule
parameter is optional, so responseTimeLimit can be null.     */
    Long responseTimeLimit = kpi.getRuleParameter("Response time
limit")

    /**      * Go over the given samples.If a sample's u_iStatus is
equal to 0,      * the sample is considered available.     */
```

```
    samples.each {Sample currentSample ->         if
(currentSample.u_iStatus == 0) {             /**             *
Check the value of the rule parameter.            * If it is not
null (meaning the user has set a value),             * and the
sample's dResponseTime is greater than the             * rule
parameter value, the value is not valid.            */
           boolean isSampleValid = true;             if
(responseTimeLimit != null) {                 // Check if
ResponseTime exceeds the rule parameter value.                 if
(currentSample.dResponseTime > responseTimeLimit) {
                   // The sample is not valid.
                   isSampleValid = false;                }
           }             if (isSampleValid) {                 //
Increase the count of available samples.
               availableSamples++;                 // Add the
sample's dResponseTime value to totalResponseTime.
               totalResponseTime += currentSample.dResponseTime
           }         }     }     if (availableSamples > 0) {
       // Set KPI value, converted to percentage.
       kpi.setValue((totalResponseTime / availableSamples))     }
}
```

例 - API グループと兄弟ルール

この項では，API グループと兄弟ルールの例を示します。次の例について説明します。

例 - 最悪の子ルール

次のルールでは，アクティブなステータスだけに基づいて，計算された CI の子 CI のすべての KPI から最悪ステータスを見つけます。子 CI のタイプは計算された KPI と同じです。アクティブなステータスは，**危険域**，**重要警戒域**，**警戒域**，**注意域**，**OK** です。

```
public void calculateKPI(CI ci, KPI kpi) {     // Get the
calculated KPI's type ID (as defined in the サービス状況 KPI
Repository).     int kpiId = kpi.getType();     // Get a list of
all of the KPIs of the calculated CI's child CIs, which are of the
same     // type as the calculated KPI.    List<KPI> childKpiList =
ci.getChildrenKPIsByID(kpiId);     // Create a variable to set the
status of the calculated KPI,     // only if an active status is
found.     boolean isActiveStatusFound = false;     // Set the
current worst status to OK; if a worse status is found this will be
updated.    Status worstStatus = Status.OK;     // Go over the list
of child KPIs.     childKpiList.each{KPI childKPI->         // Get
the child KPI's status.        Status childKpiStatus =
childKPI.status;         // Check if the child KPI's status is an
active status.         if(childKpiStatus.isActive()){
            // Mark that an active status was found.
            isActiveStatusFound = true;             // Check if the
child KPI's status is worse than the current worst status.
            if(childKpiStatus.isWorse(worstStatus)){
                // Update the worst status.
                worstStatus = childKpiStatus;             }
        }     }     // Check if an active status was found in the
child KPI.     if(isActiveStatusFound){         // Set the
calculated KPI status.         kpi.setStatus(worstStatus);     } }
```

例 - 最悪の兄弟ステータス・ルール

次のルールでは，アクティブなステータスだけに基づいて，兄弟 KPI から最悪ステータスを見つけます。アクティブなステータスは，**危険域**，**重要警戒域**，**警戒域**，**注意域**，**OK** です。

```
public void calculateKPI(CI ci, KPI kpi) {     // Get a list of all
the KPIs for the CI.    List<KPI> ciKpiList = ci.getAllKPIs();
    /**      * Create a variable to set the status of the
calculated KPI,      * only if an active status is found.     */
    boolean isActiveStatusFound = false;     // Set the current
worst status to OK; if a worse status is found this will be
updated.    Status worstStatus = Status.OK;     // Go over the list
of the CI's KPIs.     ciKpiList.each {KPI ciKPI ->         /**
        * Check that the CI's KPI is not the calculated
KPI.        * This is needed because getAllKPIs method returns all
the KPIs for the CI.        */         if (ciKPI !=  kpi) {
            /**              * The ciKPI represents a sibling KPI
of the calculated KPI.             * Get the sibling KPI's status.
 */             Status siblingKpiStatus = ciKPI.status;
            // Update worstStatus if necessary.             if
(siblingKpiStatus.isActive()) {                 isActiveStatusFound
= true;                 if (siblingKpiStatus.isWorse(worstStatus))
{                   worstStatus = siblingKpiStatus;
    }             }         }     }     // Check if an active
status was found in the sibling KPI.     if (isActiveStatusFound)
{         // Set the calculated KPI's status.         kpi.setStatus
(worstStatus);     } }
```

例 - 特定の子 CI グループ・ルール

次のルールでは，KPI ステータスを特定の子 CI(RTSM ID = "96c2df2b544683c7f79bb382d1d7b3a9") の可用性 KPI に基づいて計算します。

子 CI の可用性 KPI 値が 100 の場合，計算される KPI のステータスは OK に設定されます。ほかのすべての値の場合は KPI のステータスが[**危険域**]に設定されます。

ステータスが設定されるのは，子 CI が存在し，[可用性]KPI があり，その[可用性]KPI に値がある場合だけです。

```
public void calculateKPI(CI ci, KPI kpi) {     /**     * Get the
Availability KPI for the child CI "tx_10 from virtual_host_3".    *
The RTSM ID of "tx_10 from virtual_host_3" is
"96c2df2b544683c7f79bb382d1d7b3a9".    *     * Note:Within the UI,
the following line can be written as     * KPI childKPI = "tx_10
from virtual_host_3"."Availability"     */     KPI childKPI =
ci.getChildKpiByChildId(KpiType.Availability,
```

```
"96c2df2b544683c7f79bb382d1d7b3a9");          // Check if childKPI
is not null.It is null if no child CI with this RTSM ID exists, or
if this CI does not have the Availability KPI.     if (childKPI !=
null) {              // Check if the child KPI has a value.
        if (childKPI.valueExist) {             if (childKPI.value
== 100.0) {                 kpi.status = Status.OK         }
        else {                 kpi.status = Status.CRITICAL
            }         }     } }
```

例 - 可用性 KPI およびパフォーマンス KPI に基づく兄弟ルール

次のルールでは，兄弟の[可用性]KPI および[パフォーマンス]KPI のステータスに基づいて，KPI ステータスを計算します。

これらの KPI が存在しないか，アクティブなステータスを持たない場合，ステータスは設定されません。

これらの兄弟 KPI が存在し，両方とも OK ステータスの場合，計算された KPI ステータスは OK に設定されます。その他の場合，そのステータスは[**危険域**]に設定されます(アクティブなステータスは，**危険域**，**重要警戒域**，**警戒域**，**注意域**，**OK** です) 。

```
public void calculateKPI(CI ci, KPI kpi) {     /**      * Get the
sibling KPI of type Availability.     * If Availability KPI does
not exist, null will be returned.     */     KPI availabilityKPI =
ci.getKPI(KpiType.Availability);     // Get the sibling KPI of type
Performance.    KPI performanceKPI = ci.getKPI
(KpiType.Performance);     if (availabilityKPI != null &&
performanceKPI != null) {         // Both KPIs exist for this
CI.Check if the KPIs status is active.         if
(availabilityKPI.status.isActive() &&
performanceKPI.status.isActive()) {             // Check the KPI's
status.             if (availabilityKPI.status == Status.OK &&
                performanceKPI.status == Status.OK) {
                /**                   * Both statuses are active and
both are OK.Set this KPI's status to OK.                 */
                kpi.status = Status.OK             }
            else {                 /**                  * Both
statuses are active, and not both are OK.                   * Set
this KPI's status to CRITICAL                   */
                kpi.status = Status.CRITICAL             }
        }     } }
```

例 - CI タイプ別のグループ平均値

次のルールでは，計算された KPI と同じ CI タイプの，子 CI の KPI の平均ステータスを計算します。

タイプが「bpm_tx_from_location」の子 CI だけが計算に使用されます。このタイプの子 CI が存在しないか，値を持つ子 CI KPI がない場合は，KPI に値が設定されません。

```
public void calculateKPI(CI ci, KPI kpi) {     // Get the
calculated KPI's type ID (as defined in the サービス状況 KPI
Repository).     int kpiId = kpi.getType();     // Get a list of
the KPIs of the child CIs, which are of the same CI type as the
calculated     // KPI, whose CI type is  "bpm_tx_from_
location".    List<KPI> bpmTxFromLocationChildKpiList =
ci.getChildrenKPIsByIDAndCiType(kpiId, "bpm_tx_from_location")
    // Create a variable to sum the total values from child
KPIs.    // If no child exists or no child has value the variable
will remain null.    Double totalChildValue = null;     // Write
information to the log file.     logger.debug
("DashboardGroupAvgValueByCiTypeRule :number of child CIs with type
bpm_tx_from_location:" + bpmTxFromLocationChildKpiList.size())
    // Go over the list of child KPIs.
    bpmTxFromLocationChildKpiList.each {KPI childKPI ->         //
Sum values of the child KPIs using the Utils class, which handles
null values.         totalChildValue = Utils.sum(totalChildValue,
childKPI.value);     }     // Set the calculated KPI's value to the
average value, using the Utils class.    // If totalChildValue is
null, null value will be set.     kpi.value = Utils.divide
(totalChildValue, bpmTxFromLocationChildKpiList.size()); }
```

例 - 最悪の状況インジケータ・ルール

次のルールでは，アクティブなステータスだけに基づいて，計算された CI のすべての状況インジケータ (HI) から最悪ステータスを見つけます。アクティブなステータスは，**危険域**，**重要警戒域**，**警戒域**，**注意域**，**OK** です。

```
public void calculateKPI(CI ci, KPI kpi) {     // Get all health
indicators.    List<HI> his = ci.getHIs();     // Create a variable
to set the status of the calculated KPI,     // only if an active
status is found.     boolean isActiveStatusFound = false;     //
Set the current worst status to OK;     // if a worse status is
found this will be updated.    Status worstHiStatus = Status.OK;
    his.each {HI hi ->         Status hiStatus = hi.getStatus();
        // Check if the current HI status is an active status.
        if (hiStatus.isActive()) {             // Mark that an
active status was found.             isActiveStatusFound = true;
            // Check if the child KPI's status is worse than the
current worst status.             if (hiStatus.isWorse
(worstHiStatus)) {                 // Update the worst status.
                worstHiStatus = hiStatus;             }         }
    }     // Check if an active status was found in the child KPI.
```

```
    if (isActiveStatusFound) {          // Set the calculated KPI
status.          kpi.setStatus(worstHiStatus);      } }
```

例 - Groovy Closure の使用

次のルールでは，計算された CI の子 CI に対して重要警戒域ステータスの可用性 KPI が少なくとも 1 つ存在する場合，計算された KPI のステータスは危険域に設定されます。

このルールは Groovy Closure を示します。詳細については，http://groovy.codehaus.org/Closures (英語サイト) を参照してください。

```
public void calculateKPI(CI ci, KPI kpi) {     /**      * Use
Groovy Closure with the CI class getChildrenKPIs method,      * to
get List of KPIs from the CI`s child CIs, where      * 1. KPI type
is Availability      * 2. Status is MAJOR           Closure
description:     { KPI childKPI ->          childKPI.type ==
KpiType.Availability.getID("DASHBOARD") && childKPI.status ==
Status.MAJOR      }      The Closure defines one parameter named
childKPI of type KPI.     Each KPI from the CI`s child CIs will be
passed to the Closure by the getChildrenKPIs method.     The
Closure body returns a boolean value based on the logical
expression result.     Each KPI that the Closure body will return
true for, will be part of the returned List      The expression
KpiType.Availability.getID("DASHBOARD") returns an int representing
the Availability KPI ID from the サービス状況 KPI Repository.      */

    List<KPI> kpiList = ci.getChildrenKPIs {KPI childKPI ->
        childKPI.type == KpiType.Availability.getID("DASHBOARD") &&
childKPI.status == Status.MAJOR     }     // Check if such a KPI
exists.     if (kpiList.isEmpty()) {          // No such KPI
exists.        // Write to a log file at debug level.
        logger.debug "Closure Rule:no Availability KPI with  MAJOR
status exist"     }     else {          //  At least one
Availability KPI with MAJOR status exists.         logger.debug
("Closure Rule:At least one Availability KPI  with  MAJOR status
exist")         // Set calculated KPI status to CRITICAL.
        kpi.status = Status.CRITICAL;     } }
```

第2章

サービス状況の外部 API

本項の内容

- 51ページ「インジケータ・データの取得 API」：この API を使用して，KPI の経過時間ごとのステータス，KPI 定義，インジケータ・ステータスにアクセスできます。
- 56ページ「状況インジケータの状態リセット API」：一部のイベント・フローでは，問題が発生したことを示す HI が表示されることがあります。ただし，問題が解決されたとしても，イベントによって問題は終了されていません。問題の処理後，この API を使用して HI の状態を[**正常域**]にリセットできます。
- 58ページ「サービス状況のデータベース・クエリ API」：この API を使用してデータベースをクエリし，XML 形式のビューのリストを返すことができます。

インジケータ・データの取得 API

次の外部 API を使用して，KPI の経過時間ごとのステータス，KPI の定義，インジケータのステータスにアクセスできます。

本項の内容

サービス・ログ・ファイルは，次の場所にあります。**<ゲートウェイ・サーバのルート・ディレクトリ>\log\EJBContainer\serviceHealthExternalAPI.log**

XML 形式および JSON 形式で戻り値がサポートされています。

認証は基本アクセス認証方法を使用して行う必要があります。詳細および例については，http://en.wikipedia.org/wiki/Basic_access_authentication(英語サイト) を参照してください。

KPI の経過時間ごとのステータスの取得

次を使用して，KPI の経過時間ごとのステータスを取得できます。

API 構文

```
http://<ゲートウェイ・サーバ>/topaz/servicehealth/customers/<カスタマ ID>/
kpiOverTime?ciIds=<CI ID>&startDate=<開始日>&endDate=<終了日>
```

API では，次のパラメータを使用します。

- **customerId :**カスタマ ID(非 HP SaaS デプロイメントには **1** を使用)。
- **ciId** :必須。カンマ区切りの CI ID を使用します。
- **startDate :**必須。KPI のステータスの開始時間(1970 年 1 月 1 日からの日付を秒単位で表す値)。
- **endDate** :必須。KPI のステータスの終了時間(1970 年 1 月 1 日からの日付を秒単位で表す値)。
- **view :**オプション。ローカル影響ビューのコンテキストで結果を取得します(標準設定はグローバル・ビュー)。詳細については，『BSM ユーザ・ガイド』の「サービス状況の KPI のリスト」を参照してください。
- **kpiId :**オプション。カンマ区切りの KPI 内部 ID をリポジトリ UI と同じように使用します(標準設定では，すべての KPI で空)。詳細については，『BSM アプリケーション管理ガイド』の「サービス状況の KPI のリスト」を参照してください。

次に，API およびその出力の例を示します。

```
http://host.devlab.ad/topaz/servicehealth/customers/1/kpiOverTime?
ciIds=0b656ce308022a6739e3e726497fda6a&startDate=1296499370
&endDate=1296501466
```

```
<kpiStatuses>
    <kpiStatus>
        <ciId>0b656ce308022a6739e3e726497fda6a</entityId>
        <ciDisplayLabel>ATM 1610</ciDisplayLabel>
        <kpiType>6</kpiType>
        <kpiDisplayName>Application Performance</kpiDisplayName>
        <timeStamp>1296499370</timeStamp>
        <status>20</status>
        <statusDisplayName>OK</statusDisplayName>
        <duration>311</duration>     </kpiStatus>
    <kpiStatus>
        <ciId>0b656ce308022a6739e3e726497fda6a</entityId>
        <ciDisplayLabel>ATM 1610</ciDisplayLabel>
        <kpiType>6</kpiType>
        <kpiDisplayName>Application Performance</kpiDisplayName>
        <timeStamp>1296499681</timeStamp>
        <status>-2</status>
        <statusDisplayName>No Data</statusDisplayName>
        <duration>1785</duration>     </kpiStatus>
    <kpiStatus>
        <ciId>0b656ce308022a6739e3e726497fda6a</entityId>
        <ciDisplayLabel>ATM 1610</ciDisplayLabel>
        <kpiType>6</kpiType>
        <kpiDisplayName>Application Performance</kpiDisplayName>
        <timeStamp>1296501466</timeStamp>
        <status>20</status>
        <statusDisplayName>OK</statusDisplayName>
        <duration>13334</duration>     </kpiStatus>
    <kpiStatus>
        <ciId>0b656ce308022a6739e3e726497fda6a</entityId>
        <ciDisplayLabel>ATM 1610</ciDisplayLabel>
        <kpiType>7</kpiType>
        <kpiDisplayName>Application Availability</kpiDisplayName>
        <timeStamp>1296428400</timeStamp>
        <status>0</status>
        <statusDisplayName>Critical</statusDisplayName>
        <duration>69663</duration>
    </kpiStatus> </kpiStatuses>
```

出力フィールドは次のとおりです。

フィールド	説明
ciId	CI ID
ciDisplayLabel	CI 表示ラベル
kpiType	KPI ID(以下の53ページ「KPI 定義の取得」を参照)

フィールド	説明
kpiDisplayName	KPI の表示名
timeStamp	KPI のステータスの開始時間。1970 年 1 月 1 日からの日付を秒単位で表す値
status	KPI ステータス(以下の「インジケータ・ステータスの取得」を参照)
statusDisplayName	KPI ステータスの表示名
duration	KPI のステータスの継続時間(秒単位)。

リターン・コード

API では，次のリターン・コードを返します。

名前	エラー・コード	説明
BAD_REQUEST	400	• 開始日が終了日より後になっている • 開始日付が将来です • startDate，endDate，または ciIDs が欠落しています
UNAUTHORIZED	401	ユーザは選択されたビューに対する権限がありません
INTERNAL_ SERVER_ERROR	500	• 結果のサイズが最大クォータを超過した • 一般エラー

KPI 定義の取得

システムで定義されている KPI を取得する方法は次のとおりです。

API 構文

```
http://<ゲートウェイ・サーバ>

/topaz/servicehealth/customers/<CustomerId>/
repositories/indicators/kpis/<kpiId>
```

API では，次のパラメータを使用します。

- **customerId** :カスタマ ID(非 HP SaaS デプロイメントには **1** を使用)。
- **kpiIds** :オプション。すべての KPI について空のままにするか(標準設定)，KPI 内部 ID をリポジトリ UI と同じように入力して，特定の KPI を選択します。詳細については，『BSM アプリケーション管理ガイド』の「サービス状況の KPI のリスト」を参照してください。

次に，API およびその出力の例を示します。

```
http://host.devlab.ad/topaz/servicehealth/customers/1/repositories/-
indicators/kpis/
```

```
<kpis>
    <kpi>
        <id>1</id>
        <name>Legacy System</name>
    </kpi>
    <kpi>
        <id>1311</id>
        <name>Value</name>
    </kpi>     <kpi>
        <id>1310</id>
        <name>Exceptions</name>
    </kpi> </kpis>
```

出力フィールドは次のとおりです。

フィールド	説明
id	リポジトリUIなどのKPI内部ID。詳細については,『BSMアプリケーション管理ガイド』の「サービス状況のKPIのリスト」を参照してください。
name	KPI名

リターン・コード

API では，次のリターン・コードを返します。

名前	エラー・コード	説明
NOT_FOUND	404	KPIが見つからない
INTERNAL_ SERVER_ERROR	500	一般エラー

インジケータ・ステータスの取得

インジケータ・ステータスを取得する方法は次のとおりです。

API 構文

```
http://<ゲートウェイ・サーバ>/topaz/servicehealth/customers/<CustomerId>/
repositories/indicators/statuses
```

API では，次のパラメータを使用します。

- **customerId** :カスタマID(非 HP SaaS デプロイメントには **1** を使用)。

次に，API およびその出力の例を示します。

```
http://host.devlab.ad/topaz/servicehealth/customers/1/repositories/-
indicators/statuses
```

```
<targets>
    <target>
        <id>20</id>
    <name>OK</name>
    </target>     <target>
        <id>15</id>
        <name>Warning</name>
    </target>     <target>
        <id>10</id>
        <name>Minor</name>
    </target>     <target>
        <id>5</id>
        <name>Major</name>
    </target>     <target>
        <id>0</id>
        <name>Critical</name>
    </target>     <target>
        <id>-1</id>
        <name>Info</name>
    </target>     <target>
        <id>-2</id>
        <name>No Data</name>
    </target>     <target>
        <id>-4</id>
        <name>Downtime</name>
    </target> </targets>
```

出力フィールドは次のとおりです。

フィールド	説明
id	KPI ステータス内部 ID
name	KPI ステータス名

リターン・コード

API では，次のリターン・コードを返します。

名前	エラー・コード	説明
INTERNAL_ SERVER_ERROR	500	一般エラー

状況インジケータの状態リセット API

一部のイベント・フローでは，問題が発生したことを示すHIが表示されることがあります。ただし，問題が解決されたとしても，イベントによって問題は終了されていません。問題に対処した後で，HIの状態を[**正常**](標準設定)にリセットする必要がある場合があります。サービス状況内のHI状態のリセットに関する詳細については，『BSMユーザ・ガイド』の「状況インジケータ・コンポーネント」を参照してください。

HIの状態リセットAPIを使用すれば，BSMユーザ・インタフェースを使用していないユーザがHTTPベースのRESTプロトコルを使用してイベントベースのHIを標準設定の状態にリセットできます。

特定のCIのすべてのHIをリセットすることも，特定のHIをリセットすることもできます。

このREST APIでは大文字と小文字が区別され，**PUT**メソッドを使用します。

注: このAPIはシステム全体のパフォーマンスに影響を与える可能性があります。このAPIを使用する前にHPプロフェッショナル・サービスにご相談ください。

API構文

- CIに関連したすべてのHIをリセットするには，次の手順を実行します。

```
http://<ゲートウェイ・サーバ>
/topaz/servicehealth/customers/<CustomerId>/cis/<CI ID>/his/reset
```

- 特定のHIをリセットするには，次の手順を実行します。

```
http://<ゲートウェイ・サーバ>
/topaz/servicehealth/customers/<CustomerId>/cis/<CI ID>/his/<HI
名>/reset
```

- HIの特定のサブコンポーネントをリセットするには，次の手順を実行します。

```
http://<ゲートウェイ・サーバ>
/topaz/servicehealth/customers/<CustomerId>/cis/<CI ID>/his/<HI
名>/reset?subcomponent=<サブコンポーネント名>
```

HI名とは，インジケータ・リポジトリで定義されたHIの名前であり，HIの表示ラベルではありません。

リターン・コードとログ・ファイル

APIでは，次のリターン・コードを返します。

名前	エラー・コード	説明
OK	200	成功
UNAUTHORIZED	401	ユーザはカスタマに対して権限がありません
NOT_FOUND	404	• CIが見つかりません • HIが見つかりません • 不良要求(構文エラー)

名前	エラー・コード	説明
INTERNAL_ SERVER_ERROR	500	• RTSM エラー • リポジトリ・エラー • オンライン・エンジン・エラー

サービス・ログ・ファイルは，次の場所にあります。**<ゲートウェイ・サーバのルート・ディレクトリ> \log\EJBContainer\serviceHealthExternalAPI.log**

また，このサービスは HI リセットごとに監査ログに書き込みを行います。

サービス状況のデータベース・クエリ API

サービス状況 API を使用してデータベースをクエリし，XML 形式のビューのリストを返すことができます。

ヒント: XSLT を使用して，XML 出力をほかの形式に変換できます(通常はテキスト形式または HTML)。たとえば，基本の XSLT 変換を使用すると，モバイル・デバイスに合うようにフォーマットされた HTML レポートを作成できます。作成したレポートはモバイル・ポータルを介して送信し，重要な Business Service Management ビューとして，ユーザの携帯電話に表示できます。

クエリ構文

クエリの基本的な構文は次のとおりです。

```
http://<ゲートウェイ・サーバ>/topaz/bam/BAMOpenApi?customerId=<カスタマID>&userName=<ユーザ名>&password=<パスワード>&command=<コマンド・パラメータ>
```

定義した **command** パラメータに応じて，追加のパラメータを含めることもできます。

クエリで使用される主なパラメータ

次の表は，クエリで定義する必要のあるパラメータのリストです。

パラメータ	説明
customerID	BSM カスタマは 1 を指定する必要があります。HP Software-as-a-Service カスタマは一意のカスタマ ID を指定する必要があります。
userName	BSM で定義されているユーザ名を指定します。クエリでは，ログイン資格情報は暗号化されません。
password	入力したユーザ名に対応するパスワードを指定します。クエリでは，ログイン資格情報は暗号化されません。
command	次のいずれかの値を指定します。 **getViews**: すべてのビューを実行時サービス・モデル(RTSM)から取得するために指定します。ほかのパラメータは必要ありません。 **getNodes**: 指定したビューのすべての子ノードを取得するために指定します(**viewName** パラメータで子ノードを取得するためのビューも指定する必要があります)。このコマンド・パラメータを使用する場合は，**showTooltip**，**depth**，**layout**，**xsltURL**，**responseContentType** の各パラメータも設定できます。
viewName	**getNodes** コマンド・パラメータが定義されている場合は，このパラメータをクエリに含めて，取得するビューを指定します。すべてのビューとそのノードを取得するには，値を **ticker_all_views** に設定します。
showTooltip	**getNodes** コマンド・パラメータが定義されている場合は，このパラメータをクエリに含めて，サービス状況の KPI ツールチップ・データを表示するかどうかを指定できます。データを表示する場合は **true**，表示しない場合は **false** を指定します。標準設定値は **false** です。

パラメータ	説明
depth	**getNodes** コマンド・パラメータが定義されている場合は，このパラメータをクエリに含めて，表示するビューのレベル数を指定できます。標準設定の値は **1** です。
layout	**getNodes** コマンド・パラメータが定義されている場合は，このパラメータをクエリに含めて，クエリ結果のレイアウトを **hierarchical**(階層)または **flat**(フラット)で指定できます。フラット・モードでは，すべてのノードはフラット・リストで取得されます。階層モードでは，ノードはビューと同じ階層内に取得されます。標準設定値は **flat** です。
xsltURL	**getNodes** コマンド・パラメータが定義されている場合は，このパラメータをクエリに含めて，.xml 形式のクエリ結果を変換する .xslt ファイルへの URL を指定できます。
responseContentType	**getNodes** コマンド・パラメータが定義されており，**xsltURL** パラメータがクエリに含まれている場合は，このパラメータをクエリに含めて応答 MIME タイプを指定できます。

クエリの例

次に，クエリおよび返されるデータの例を示します。

次のクエリは，実行時サービス・モデル(RTSM)のすべてのビューを示すフラット・リストを返します。

```
http://myserver/topaz/bam/BAMOpenApi?customerId=1
&userName=admin&password=admin&command=getViews
```

次のクエリは，Service Measurements ビューの KPI ステータスとツールチップ情報を示す階層型ツリーを，3 層目の子ノード分まで返します。

```
http://myserver/topaz/bam/BAMOpenApi?customerId=1&userName=
admin&password=admin&command=getNodes&viewName=Service%20
Measurements&showTooltip=true&depth=3&layout=hierarchical
```

第2部分

サービス・レベル管理

第3章

SLM 外部 API

SLM 外部 API を使用すると，SLA の設定プロパティや SLA の計算結果を取得し，外部アプリケーションでこれらのデータを利用できます。

この API により，BSM 外部で SLM データの利用と処理ができるようになります。詳細については，次を参照してください。

- 62ページ「SLA 設定データの取得」
- 65ページ「SLA 計算結果の取得」
- 67ページ「カレンダーの取得」
- 69ページ「追跡期間の取得」
- 71ページ「KPI の取得」
- 73ページ「インジケータ・ステータスの取得」

サービス・ログ・ファイルは，次の場所にあります。**<ゲートウェイ・サーバのルート・ディレクトリ>\log\EJBContainer\slmExternalAPI.log**

XML 形式および JSON 形式で戻り値がサポートされています。

認証は基本アクセス認証方法を使用して行う必要があります。詳細と例については，http://en.wikipedia.org/wiki/Basic_access_authentication(英語サイト) を参照してください。

SLA 設定データの取得

SLA 設定データを取得する方法は次のとおりです。

API 構文

```
http://<ゲートウェイ・サーバ>/topaz/slm/customers/<CustomerId>/sla/<slaId>
```

API では，次のパラメータを使用します。

- **customerId** :カスタマ ID (非 HP SaaS デプロイメントには **1** を使用)
- **slaId** :SLA RTSM ID(必須ではないパラメータ。システム内のすべての SLA を取得するには空のままとする) 。

次に，API 出力の例を示します。

```
<sla>
    <id>c808d3ddce3d849c1cc0d667bec7f6cc</id>
    <name>TestSLA3</name>
    <description></description>
    <details></details>
    <timezone>Central Standard Time</timezone>
    <type>SLA</type>
    <creator>administrator</creator>
    <url></url>
    <classification>Formal</classification>
    <customer>
        <customerId>e35cded4b3a628a8da4b9cf352007467</customerId>
        <customerName>IT Department</customerName>     </customer>
    <provider>
        <providerId>ae9ea545c5f99adb52d7edd641bbf8ef</providerId>
        <providerName>Customers</providerName>     </provider>
    <state>Running</state>
    <startDate>1290938400</startDate>
    <endDate>1608631200</endDate>
    <targets>
        <target>0</target>
        <target>5</target>
        <target>20</target>
        <target>10</target>
        <target>15</target>
    </targets>
    <trackingPeriods>

     <trackingPeriod>3ef2dfe04fa07c349e72ca9e7b04e2af</trackingPeri-
od>
```

```
     <trackingPeriod>bffe0ea9334ff7f309e969eec3c9c266</trackingPeri-
od>

     <trackingPeriod>a562b09777271425835abadb12f32e73</trackingPeri-
od>

     <trackingPeriod>f9f77b20f986fbb2eb064b0a30ece93d</trackingPeri-
od>
     <trackingPeriod>4ecf36e0eb88816745a8849db029c73f</trackingPeri-
od>     </trackingPeriods>
    <calendars>
        <calendar>ecf840eef788851986195301aba206fd</calendar>
    </calendars>
</sla>
```

出力フィールドは次のとおりです。

フィールド	説明
slaId	SLA ID
name	SLA 名
description	SLA の説明
details	SLA の詳細
type	OLA, SLA, UC
creator	SLA を作成したユーザの名前
timezone	SLA のタイム・ゾーン
url	外部ソースにある契約詳細へのリンク
classification	公式または非公式
customer	SLA のカスタマに指定されている組織の CI ID と名前
provider	SLA のプロバイダに指定されている組織の CI ID と名前
state	準備中, 実行中, 終了
startDate	SLA の開始日 (1970 年 1 月 1 日 00:00:00 GMT からのミリ秒単位で日付を表す long 値)
endDate	SLA の終了日 (1970 年 1 月 1 日からのミリ秒単位で日付を表す long 値)
targets	SLA ターゲットの ID
trackingPeriods	追跡期間の ID
calenders	SLA カレンダーの ID

リターン・コード

API では，次のリターン・コードを返します。

名前	エラー・コード	説明
UNAUTHORIZED	401	選択した SLA に対する閲覧権限がユーザにない
NOT_FOUND	404	SLA が見つからない
INTERNAL_ SERVER_ERROR	500	一般エラー

SLA 計算結果の取得

SLA の計算結果を取得する方法は次のとおりです。この API は，CI，KPI，追跡期間，カレンダーによるフィルタリングをサポートしています。

API 構文

```
http://<ゲートウェイ・サーバ>/topaz/slm/customers/
<CustomerId>/ciSummary/<view>/sla/ <slaID>?
calendar=<calendarId>&startDate=<startDate>&ciIds=<ciIds>&kpiId=<kpiI-
d>
```

API では，次のパラメータを使用します。

- **customerId :**カスタマ ID (非 HP SaaS デプロイメントには **1** を使用)
- **view :** 必須。レポート・ビュー。次のいずれかを使用します。hour，day，week，month，quarter，year，weekToDate，monthToDate，quarterToDate，yearToDate，slaToDate
- **slaId :**SLA ID(すべての SLA を指定するには **"all"** を使用)。
- **calendars :** オプション。SLA のカレンダー ID(SLA で定義されているすべてのカレンダーを指定するときは空のままとする。67ページ「カレンダーの取得」を参照)。
- **startDate :** クローズ期間の場合のみ必須 (秒単位)。
- **endDate :** クローズ期間の場合のみ必須 (秒単位)。
- **ciIds :** オプション。カンマ区切りの CI ID。すべての SLA CI を指定するときは空のままとする(slaId が **"all"** のときは SLA ノード)。
- **kpiId :** オプション。KPI の内部 ID(KPI リポジトリ UI に表示される ID)。すべての KPI を指定するには空のままとする。この API はサービス停止 KPI をサポートしません。

次に，API 出力の例を示します。

```
<slaStatus>
    <slaId>2c4d69f2784021a6a94b7a40100ba31f</slaId>
    <ciId>b8ae7539b0cb338f3bce84b4866647eb</id>
    <startDate>1293832800</startDate>
    <endDate>1294225200</endDate>

    <trackingPeriod>a562b09777271425835abadb12f32e73</trackingPerio-
d>
    <calendar>dd9ef8826c553d3e217fa0b3bf03f0a0</calendar>
    <kpiType>220</kpiType>
    <kpiDisplayName>SLM Status</kpiDisplayName>
    <kpiStatus>0</kpiStatus>
    <statusDisplayName>Failed</statusDisplayName>
    <kpiValue>0</kpiValue> </slaStatus>
```

出力フィールドは次のとおりです。

フィールド	説明
slaId	SLA ID
ciId	CI ID
startDate	期間の開始日(1970 年 1 月 1 日からの秒単位で日付を表す long 値)
endDate	期間の終了日(1970 年 1 月 1 日からの秒単位で日付を表す long 値)
trackingPeriod	サンプルの追跡期間:時間,日,週,月,四半期,年,SLA 期間(69ページ「追跡期間の取得」を参照)
calendar	サンプルのカレンダー(67ページ「カレンダーの取得」を参照)
kpiType	KPI ID(71ページ「KPI の取得」を参照)
kpiDisplayName	KPI の表示名
kpiStatus	KPI のステータス ID(73ページ「インジケータ・ステータスの取得」を参照)
statusDisplayName	KPI ステータスの表示名
kpiValue	KPI の数値

リターン・コード

API では,次のリターン・コードを返します。

名前	エラー・コード	説明
BAD_REQUEST	400	• ビューがサポートされていない • 開始日が終了日より後になっている • クローズ・ビュー(クローズ期間)の startDate,endDate が見つからない
UNAUTHORIZED	401	選択した SLA に対する閲覧権限がユーザにない
INTERNAL_SERVER_ERROR	500	• 結果のサイズが最大クォータを超過した • 一般エラー

カレンダーの取得

システムで定義されているカレンダーを取得する方法は次のとおりです。

API 構文

```
http://<ゲートウェイ・サーバ>/topaz/slm/customers/
<CustomerId>/repositories/calendars/<calendarId>
```

API では，次のパラメータを使用します。

- **customerId :**カスタマ ID (非 HP SaaS デプロイメントには **1** を使用)
- **calendarId:** カレンダー ID (システムのすべてのカレンダーを指定するには空のままとする)

次に，API 出力の例を示します。

```
<calendars>     <calendar>
        <id>5c6c09ec3d61db775333bb5beb5ea863</id>
        <name>testcalc</name>
        <description></description>
        <scheduleDescription>Friday 3:00 AM - 3:30 AM, Wednesday
2:30 PM - 3:00
        PM, Thursday 5:30 PM - 6:00 PM, Wednesday 3:30 AM - 4:30
AM, Tuesday 11:00 AM - 11:30 AM, Friday 10:00 AM - 10:30 AM, Monday
2:00 AM - 2:30 AM, Wednesday 7:00 AM - 7:30 AM, Thursday 10:30 AM -
11:00 AM, Sunday, Monday 6:30 AM - 7:00 AM, Friday 2:00 PM - 2:30
PM, Monday 8:00 PM - 8:30 PM, Thursday 9:00 PM - 9:30 PM
        </scheduleDescription>
    </calendar>
    <calendar>
        <id>ecf840eef788851986195301aba206fd</id>
        <name>24x7</name>
        <description>24 hours, 7 days a week</description>
        <scheduleDescription>Sunday, Monday, Tuesday, Wednesday,
Thursday, Friday, Saturday 12:00 AM - 12:00 AM
        </scheduleDescription>
    </calendar>
    <calendar>
        <id>86594b724cb03e2647b2a86b08103a28</id>
        <name>Business Hours</name>
        <description>8:00-17:00, Monday-Friday</description>
        <scheduleDescription>Monday, Tuesday, Wednesday, Thursday,
Friday 8:00 AM - 5:00 PM
        </scheduleDescription>
    </calendar> </calendars>
```

出力フィールドは次のとおりです。

フィールド	説明
id	カレンダー ID
name	カレンダーの表示名
description	カレンダーの詳細
scheduleDescription	カレンダーのスケジュール設定を表す文字列

リターン・コード

API では，次のリターン・コードを返します。

名前	エラー・コード	説明
NOT_FOUND	404	カレンダーが見つからない
INTERNAL_ SERVER_ERROR	500	一般エラー

追跡期間の取得

システムで定義されている追跡期間を取得する方法は次のとおりです。

API 構文

```
http://<ゲートウェイ・サーバ>/topaz/slm/customers/
<CustomerId>/trackingPeriods/<trackingPeriodId>
```

API では，次のパラメータを使用します。

- **customerId :** カスタマ ID (非 HP SaaS デプロイメントには **1** を使用)
- **trackingPeriodId:** 追跡期間 ID(システムのすべての追跡期間を指定するには空のままとする)。

次に，API 出力の例を示します。

```
<trackingPeriods>     <trackingPeriod>
        <id>a562b09777271425835abadb12f32e73</id>
        <name>Month</name>
    </trackingPeriod>
    <trackingPeriod>
        <id>dd613d616284c14ec848a4fa5d939655</id>
        <name>Hour</name>      </trackingPeriod>
    <trackingPeriod>
        <id>2e57ee4d9c4f6b50a1d01385a861aa4b</id>
        <name>Day</name>
    </trackingPeriod>
    <trackingPeriod>
        <id>4ecf36e0eb88816745a8849db029c73f</id>
        <name>Week</name>
    </trackingPeriod>
    <trackingPeriod>
        <id>3ef2dfe04fa07c349e72ca9e7b04e2af</id>
        <name>Quarter</name>
    </trackingPeriod>
    <trackingPeriod>
        <id>f9f77b20f986fbb2eb064b0a30ece93d</id>
        <name>Year</name>
    </trackingPeriod>
    <trackingPeriod>
        <id>bffe0ea9334ff7f309e969eec3c9c266</id>
        <name>SLA period</name>     </trackingPeriod>
</trackingPeriods>
```

出力フィールドは次のとおりです。

フィールド	説明
id	追跡期間 ID
name	追跡期間の表示名

リターン・コード

API では，次のリターン・コードを返します。

名前	エラー・コード	説明
INTERNAL_ SERVER_ERROR	500	一般エラー

KPI の取得

システムで定義されている KPI を取得する方法は次のとおりです。

API 構文

```
http://<ゲートウェイ・サーバ>/topaz/slm/customers/
<CustomerId>/repositories/indicators/kpis/<kpiId>
```

API では，次のパラメータを使用します。

- **customerId :** カスタマ ID (非 HP SaaS デプロイメントには **1** を使用)
- **kpiId :** KPI ID(システムのすべての KPI を指定するには空のままとする) 。

次に，API 出力の例を示します。

```
<kpis>     <kpi>
        <id>1</id>
        <name>Legacy System</name>     </kpi>      <kpi>
        <id>1311</id>
        <name>Value</name>
    </kpi>     <kpi>
        <id>1310</id>
        <name>Exceptions</name>
    </kpi>     <kpi>
        <id>615</id>
        <name>Operational Status</name>     </kpi>
    <kpi>
        <id>6</id>
        <name>Application Performance</name>     </kpi>
    <kpi>
        <id>7</id>
        <name>Application Availability</name>     </kpi>
    <kpi>
        <id>1500</id>
        <name>Generic</name>
    </kpi> </kpis>
```

出力フィールドは次のとおりです。

フィールド	説明
id	KPI ID
name	KPI の表示名

リターン・コード

API では，次のリターン・コードを返します。

名前	エラー・コード	説明
NOT_FOUND	404	KPI が見つからない
INTERNAL_ SERVER_ERROR	500	一般エラー

インジケータ・ステータスの取得

インジケータ・ステータスを取得する方法は次のとおりです。

API 構文

```
http://<ゲートウェイ・サーバ>/topaz/slm/customers/
<CustomerId>/repositories/indicators/targets
```

API では，次のパラメータを使用します。

customerId : カスタマ ID (非 HP SaaS デプロイメントには **1** を使用)

次に，API 出力の例を示します。

```
<targets>     <target>
        <id>20</id>
        <name>Exceeded</name>
    </target>     <target>
        <id>15</id>
        <name>Met</name>
    </target>     <target>
        <id>10</id>
        <name>Minor Breached</name>
    </target>
    <target>
        <id>5</id>
        <name>Breached</name>
    </target>     <target>
        <id>0</id>
        <name>Failed</name>
    </target>     <target>
        <id>-4</id>
        <name>Downtime</name>
    </target>
    <target>
        <id>-2</id>
        <name>No Data</name>
    </target> </targets>
```

出力フィールドは次のとおりです。

フィールド	説明
id	ターゲット ID/ステータス ID(計算結果に表示される ID)
name	ターゲットの表示名/ステータスの表示名

リターン・コード

API では，次のリターン・コードを返します。

名前	エラー・コード	説明
INTERNAL_ SERVER_ERROR	500	一般エラー

第4章

SLM ルール API

注: BSM バージョン 9.00 以降では，インジケータのステータスおよび値をサンプルに基づいて計算するルール(85ページ「API サンプル・ルール」 および87ページ「API 継続時間ベースのサンプル・ルール」) を使用して，メトリックベースの状況インジケータ(HI) を計算します。

ルール API のドキュメントには，さまざまな KPI の計算方法が記載されています。**BSM バージョン 9.00 以降では，サンプル・ベース値の計算時にこれらのメソッドがメトリックベースの HI の計算に使用されます。**

本章では，ルール API を使用して主要管理指標(KPI) を計算するためのビジネス・ルールを作成する方法について説明します。標準設定のサービス・レベル管理ルールについては，『BSM アプリケーション管理ガイド』の「SLM ビジネス・ルールのリスト」の項に記載されています。

新しいルールの作成には，ルール API を使用することをお勧めします。ルール API により，Groovy 動的スクリプト言語とGroovy Runtime Environment 1.7.3 以前を使用してルールを作成できます。この API のユーザは Groovy，Java 構文，および BSM の管理およびアプリケーションに精通していている必要があります。

ルール API クラスについては，『HP Rules API Reference』に Javadoc 形式で記載されています。これらのファイルは次のフォルダにあります。
\\<ゲートウェイ・サーバのルート・ディレクトリ>\AppServer\webapps\site.war\amdocs\eng\doc_lib\Service_Health\Rules_API\index.html

注: ルール API を使用するルールを多数使用すると，SLA 計算のパフォーマンスに影響します。

API ルールのタイプ

次に，サービス・レベル管理の API ルールを示します。

- **単純平均ルールと単純継続時間ベースの平均ルール:** これらのルールはサンプル・フィールドから取得したデータに基づいて状況インジケータの平均値を計算し，これらの値を操作できるようにします。これらのルールは，計算で使用されるサンプル・フィールドを入力するだけとほかの API ルールよりも定義が簡単です。詳細は，77ページ「API 単純平均ルール」を参照してください。
- **グループと兄弟ルール**:このルールでは，元のサンプル・データでなく，ほかの KPI から受信した集計値に基づいて KPI を計算します。受信したデータの送信元は，子 CI(グループ) の KPI，または同じ CI(兄弟) に関連付けられた別の KPI などになります。詳細は，78ページ「API グループと兄弟ルール」を参照してください。
- **サンプル・ルール**:このルールでは，サンプル・フィールドから取得したデータに基づいて KPI を計算します。詳細は，85ページ「API サンプル・ルール」を参照してください。
- **継続時間ベースのサンプル・ルール**:このルールでは，サンプル・フィールドから取得したデータに基づいて KPI を計算します。ルール計算には，サンプルの値と継続時間が使用されます(経過時間ごとの可用性や経過時間ごとのバックログなど) 。詳細は，87ページ「API 継続時間ベースのサンプル・ルール」を参照してください。

- **サンプルによるサービス停止ルール**:このルールでは，サンプルから取得したデータに基づいてサービス停止を計算します。詳細は，91ページ「API サンプルによるサービス停止ルール」を参照してください。

サンプル・ルールの計算方法の詳細については，81ページ「サンプル・ルールの計算メカニズム - 概要」を参照してください。

API ルールの作成

ルールは，ルール API を使用して，次の 3 つの方法で作成できます。

- ［KPI 定義］ページを使用して，特定の KPI のルールを作成する
- テキスト・ファイルを使用して，複数の KPI の新しいルールを作成する。
- ルール・リポジトリ内の API ルール・テンプレートの複製を使用する

これらの方法については，93ページ「ルール API を使用したルールの作成」を参照してください。

ツールチップおよびログ・ファイル

ルール API を使用する際にツールチップに KPI 情報を表示する方法については，99ページ「ツールチップ・エントリの使用方法」を参照してください。

101ページ「ルール API コードからログ・ファイルに書き込む方法」で説明するように，ルール API コードからログ・ファイルに書き込むことができます。

API 単純平均ルール

API 単純平均ルールおよび API 単純継続時間ベースの平均ルールでは，サンプル・フィールドから取得したデータに基づいて状況インジケータの平均値を計算します。これらの計算はユーザが簡単に操作できるようになっています。

API 単純平均ルール

単純平均ルールではサンプル・フィールド値の平均が計算され，ルール・パラメータの列挙子とデノミネーションが使用されます。

- 計算に使用するサンプル・フィールドの名前を[**列挙子**]領域に**sample.<フィールド名>** の形式で入力します。

  サンプル・フィールド dValue の平均値を計算するには，[列挙子]に「`sample.dValue`」を入力します。このルールでは，計算サイクル中に受信したサンプルからこのフィールドの値を取得して，その平均値が計算されます。この平均が，その計算サイクルにおける状況インジケータの値となります。

  数式を使用してこれらの計算を操作することもできます。たとえば，2 つの値 (送信されたテキスト・メッセージを表す attr1 と失敗したテキスト・メッセージを表す attr2) を含むサンプルを送信する EMS 統合があるとします。成功メッセージの平均数 (attr1-attr2) を計算するには，「列挙子」に「`sample.attr1-sample.attr2`」を入力します。

- [**デノミネーション**]領域を使用すると，上記の値にアクションを実行できます。この領域に数値や値を入力すると，[列挙子]の結果の除算に使用されます。

  たとえば，[列挙子]に「`sample.dValue`」を，[デノミネーション]に「`2 `」を入力した場合，サンプルの[dValue]フィールドの平均は 2 で除算され，状況インジケータの値はサンプル・フィールドの平均の半分となります。

API 単純継続時間ベースの平均ルール

単純継続時間ベースの平均ルールでは，サンプル自体から継続時間値が取得され，サンプル・フィールド値の平均が計算されます。

計算に使用するサンプル・フィールドの名前を[**列挙子**]フィールドに**Sample.<フィールド名>** の形式で入力します。

このルールはサンプルから継続時間を取得し，継続時間で除算して，サンプル・フィールド値の平均を計算します。

たとえば，指定したフィールドの値が 10(40 分間) および 5(20 分間) の場合，(10 × 40) ＋ (5 × 2) /60 ＝ 8.33 と計算されます。したがって，この時間における状況インジケータの平均値は 8.33 となります。

API グループと兄弟ルール

API グループと兄弟ルールでは，元のサンプル・データでなく，ほかの KPI から受信したデータに基づいて KPI を計算します。データは，子 CI の KPI や，同じ CI に関連付けられているほかの KPI または HI から取得されます。

KPI はグループまたは兄弟 KPI または HI の集計値に基づいて計算されます。KPI の計算結果は，集計結果を表します。

注: 兄弟ルールを作成する場合は，KPI の[計算順序]フィールドで定義されたように，KPI がその兄弟 KPI の後で計算されるようにする必要があります。詳細は，「KPI リポジトリ・ページ」を参照してください。

グループと兄弟ルールのメソッドおよびフィールド

グループと兄弟ルールは，次のガイドラインに従って，ルール API インタフェース **GroupAndSiblingCalculator** を実装します。

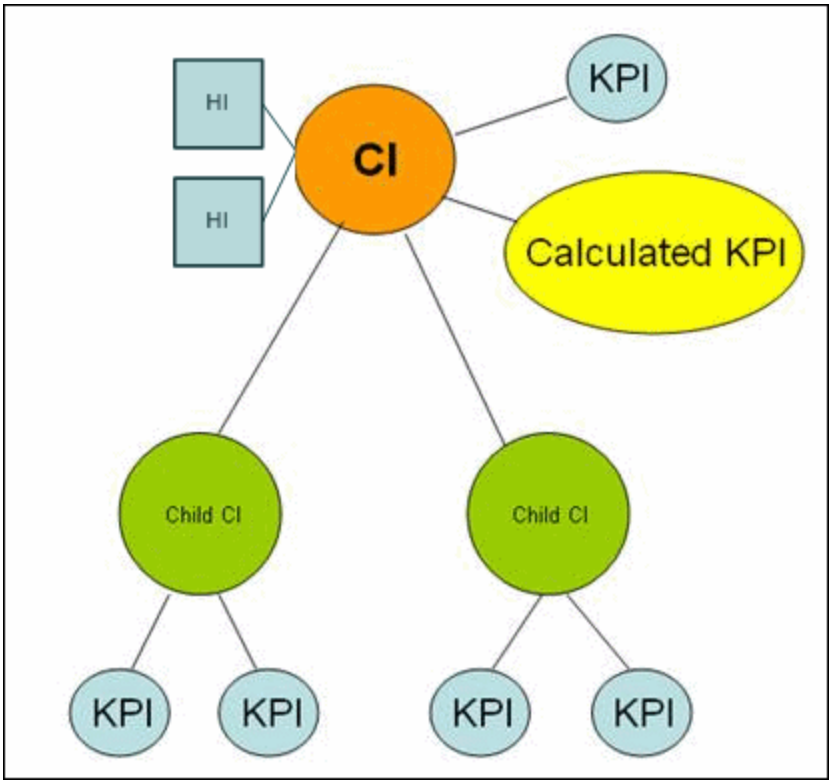
- このインタフェースでは，唯一のメソッドは **calculateKPI** です。メソッド・シグネチャは次のとおりです。

  ```
  public void calculateKPI(CI ci, KPI kpi)
  ```

- **calculateKPI** メソッドには，現在の CI を表す **ci** と，API ルールで値が計算される KPI や HI を表す **kpi** のパラメータがあります。
- **ci** パラメータのタイプは[**CI**]で，子 CI の KPI または兄弟 KPI，あるいは CI の HI へのアクセサとして使用されます。**CI** によって返される **KPI** オブジェクトを使用すると，これらの KPI や HI の集計値を取得できます。
- **kpi** パラメータ・タイプは **KPI** で，計算結果の設定に使用されます。

次の図では，計算された KPI が兄弟 KPI または子 KPI に基づいて計算され，**kpi** パラメータによって表されます。

計算された KPI が割り当てられる CI は，**ci** パラメータで表され，ほかの KPI や HI のアクセス機構となります。



ルール API クラスについては,『HP Rules API Reference』に Javadoc 形式で記載されています。これらのファイルは次のフォルダにあります。
\\<ゲートウェイ・サーバのルート・ディレクトリ>\AppServer\webapps\site.war\amdocs\eng\doc_lib\Service_Health\Rules_API\index.html

グループと兄弟ルールの例については,104ページ「例 - API グループと兄弟ルール」を参照してください。

API ルールは,93ページ「ルール API を使用したルールの作成」の説明に従って,[KPI 定義]ページまたはルール・リポジトリから,あるいはテキスト・ファイル・テンプレートを使用して定義できます。

KPI 定義ページまたはルール・リポジトリでのグループと兄弟ルールの定義

[KPI 定義]ページまたはルール・リポジトリから,グループと兄弟ルールを定義するには,[**KPI 計算スクリプト**]領域に **calculateKPI** メソッドを実装します。

calculateKPI メソッドのパラメータ **ci** と **kpi** がこのスクリプトで使用するために利用可能です。

詳細については,94ページ「特定の KPI またはサービス停止用 API ルールの定義方法」または98ページ「ルール・リポジトリでの API ルールの定義方法」を参照してください。

[KPI 定義]ページ内での特定の子 KPI へのアクセス

[KPI 定義]ページで特定の KPI にグループ・ルールを作成する場合,特定の子 KPI にアクセスするためのコードを簡素化するメカニズムが API には組み込まれています。KPI 計算スクリプトを定義する場合は,**"<CI 名>"."<KPI 名>"** という形式で入力します。

この例については,104ページ「例 - API グループと兄弟ルール」を参照してください。

テキスト・ファイルを使用した，グループと兄弟ルールの定義

テキスト・ファイルを使用してグループと兄弟ルールを定義するには，95ページ「テキスト・ファイル・ベースの API ルールの作成方法」の説明に従って **SlmGroupAndSiblingTemplate.groovy** テンプレートを使用します。

テキスト・ファイルに **calculateKPI** メソッド本体を入力します。

サンプル・ルールの計算メカニズム - 概要

API サンプル・ルールおよび継続時間ベースのサンプル・ルールでは，カレンダーや追跡期間の各組み合わせごとに，サンプル値に基づいて KPI を計算します。この計算プロセスは複数の計算プロセスで構成されます。

計算サイクルごとに，次の手順が発生します。

1. プロファイル・データベースに格納されている計算サイクル時間のサンプルが取得されます。たとえば，10:00～11:00 の KPI を計算する場合，サイクル継続時間が 5 分であれば，10:00～10:05 のタイムスタンプを持つデータベース内のサンプルが，最初のサイクル計算で使用されます。
2. サンプルにフィルタリング・メカニズムが適用され，計算に含めるサンプルが判別されます。
3. フィルタリング・メカニズムを通過したサンプルに基づいて，サイクルの KPI が計算されます。詳細は，82ページ「サンプル・ルール:サンプルに基づく KPI の計算」を参照してください。
4. サイクルの結果，および全計算期間の以前の集計結果に基づいて，KPI の集計結果が計算されます。これらの KPI 集計結果は，サービス・レベル管理レポートに表示されます。詳細は，83ページ「サンプル・ルール:KPI の集計結果の計算」を参照してください。

サンプル・ルール :サンプルに基づく KPI の計算

KPI 結果は，計算サイクルごとに **samples** パラメータに基づいて **calculateKPI** メソッドによって計算されます。**samples** パラメータは **Sample** オブジェクトの **List** で，サンプル・フィールド値を保持します。

ルール API クラスについては，『HP Rules API Reference』に Javadoc 形式で記載されています。これらのファイルは次のフォルダにあります。
\\<ゲートウェイ・サーバのルート・ディレクトリ>\AppServer\webapps\site.war\amdocs\eng\doc_lib\Service_Health\Rules_API\index.html

calculateKPI メソッドは，KPI 値，キー，ツールチップの設定に使用できます。

calculateKPI メソッドで設定される値，キー，およびツールチップは，83ページ「サンプル・ルール :KPI の集計結果の計算」で説明するとおり，**calculateAggregatedKPI** メソッドで利用できます。

KPI 値の設定

サンプル・フィールドから直接 KPI 値を計算する場合は，KPI 値を使用する必要があります。この値の設定の詳細については，**setValue** メソッドのドキュメントを参照してください。

KPI 値を使用する例については，108ページ「例 - サンプルベースの最大応答時間ルール」を参照してください。

KPI 計算キーの設定

各 KPI には，値やツールチップを計算する場合に役立つキーを保持できます。たとえば平均応答時間(合計応答時間/合計サンプル数)の計算時に，**setKey** メソッドが現在のサイクルに対する 2 つのキー(合計応答時間，合計サンプル数)の設定に使用されます。これらのキーが集計されて，KPI の集計値の計算に使用されます。

KPI 計算キーを使用する例については，105ページ「例 - サンプルベースの平均応答時間ルール」を参照してください。

KPI ツールチップの設定

サンプル・フィールドから直接 KPI ツールチップを計算する場合は，KPI ツールチップを使用する必要があります。ツールチップの設定の詳細については，**setTooltip** メソッドのドキュメントを参照してください。

サンプル・ルール :KPI の集計結果の計算

calculateAggregatedKPI メソッドは **calculateKPI** メソッドで設定された値，ツールチップ，およびキーを使用して，集計値，集計キー，および集計ツールチップを計算します。

集計結果はそのサイクルの KPI 計算結果と，以前のサイクルの集計計算結果に基づいています。

ルール API クラスについては，『HP Rules API Reference』に Javadoc 形式で記載されています。これらのファイルは次のフォルダにあります。
\\<ゲートウェイ・サーバのルート・ディレクトリ>\AppServer\webapps\site.war\amdocs\eng\doc_lib\Service_Health\Rules_API\index.html

calculateAggregatedKPI メソッドは，KPI 集計値，集計キー，集計ツールチップの設定に使用できます。

KPI 集計値の設定

KPI 集計値は KPI 値(最小の計算)または集計キー(平均応答時間など)に基づいて計算できます。KPI 集計値はサービス・レベル管理レポートに表示されます。集計値の設定の詳細については，**setAggregatedValue** メソッドのドキュメントを参照してください。

KPI 値に基づいて KPI 集計値を計算する例については，108ページ「例 - サンプルベースの最大応答時間ルール」を参照してください。

KPI 集計キーに基づいて KPI 集計値を計算する例については，105ページ「例 - サンプルベースの平均応答時間ルール」を参照してください。

KPI 集計計算キーの設定

集計キーは，集計値または集計ツールチップを計算する場合に役立ちます。集計キーは，集計キーと現在のサイクルのキーに基づいて計算されます。

たとえば平均応答時間(合計応答時間/合計サンプル数)の計算時に，**setAggregatedKey** メソッドが2つの集計キー(合計応答時間，合計サンプル数)の設定に使用されます。これらの値は，その後，集計 KPI 値を計算するときに使用します。

集計キーの設定の詳細については，**setAggregatedKey** メソッドのドキュメントを参照してください。

KPI 集計計算キーを使用する例については，105ページ「例 - サンプルベースの平均応答時間ルール」を参照してください。

KPI ステータスの設定

ルールで KPI ステータスを設定する場合は，**calculateAggregatedKPI** メソッド内で **setStatus** メソッドを使用します。

KPI 集計ツールチップの設定

KPI 集計ツールチップは，KPI ツールチップまたは集計キーに基づいて計算できます。KPI 集計ツールチップはサービス・レベル管理レポートに表示されます。

集計ツールチップの設定の詳細については，**setAggregatedTooltip** メソッドのドキュメントを参照してください。

サンプル・ルールまたは継続時間ベースのサンプル・ルールを使用する場合

計算内でサンプル継続時間を使用せず，(必要に応じて) サンプル値およびサンプル数に基づいて KPI を計算するには，サンプル・ルールを使用します。

計算内でサンプル数を使用しないで，サンプル値およびサンプル継続時間に基づいて KPI を計算するには，継続時間ベースのサンプル・ルールを使用します。

平均応答時間の計算例

平均応答時間を計算するには，サンプル・ルールまたは継続時間ベースのサンプル・ルールを使用します。

10:00 にサンプルが取得され，応答時間は 100 秒でした。10:55 にもサンプルが取得され，応答時間は 50 秒でした。ルールは 10:00～11:00 の期間で計算されます。

API サンプル・ルールでは，平均応答時間 (合計応答時間/サンプル数) ＝ 75 秒が計算されます。

API 継続時間ベースのサンプル・ルールでは，サンプルの値および継続時間に基づいて加重応答時間を計算します。最初のサンプルの継続時間は 55 分，2 番目のサンプルの継続時間は 5 分です。この期間の加重平均応答時間は(55 × 100 ＋ 5 × 50) /60 ＝ 95.83 秒になります。

API サンプル・ルール

サンプル・ルールでは，サンプル・フィールドから取得した集計データに基づいて KPI を計算します。サンプル・ルールを使用する場合については，84ページ「サンプル・ルールまたは継続時間ベースのサンプル・ルールを使用する場合」を参照してください。

ルールの計算方法の詳細については，81ページ「サンプル・ルールの計算メカニズム - 概要」を参照してください

サンプル・ルールのメソッドおよびフィールド

サンプル・ルールでは，次のメソッドを含むルール API インタフェース **SlmSamplesAggregatedCalculator** を実装します。

```
public void calculateKPI(CI ci, SlmKPI kpi, List<サンプル> samples)

public void calculateAggregatedKPI(CI ci, SlmKPI kpi)

public boolean isSampleValid(CI ci, SlmKPI kpi, Sample sample)
```

- **calculateKPI** メソッドは，計算サイクルごとに KPI を計算します。このメソッドにはパラメータ **ci**，**kpi**，および **samples** が含まれています。これらは現在の CI，ルールで値を計算する KPI，およびルール計算に使用するサンプルを表します。
- **kpi** パラメータのタイプは **SlmKPI** であり，計算結果を設定するために使用できます。
- **samples** パラメータは **Sample** オブジェクトの **List** で，サンプル・フィールド値を保持します。
- **calculateAggregatedKPI** メソッドは集計 KPI を計算します。このメソッドにはパラメータ **ci**，および **kpi** が含まれています。
- **isSampleValid** メソッドはサンプルをフィルタリングする場合に使用します。無効なサンプルは，計算には含まれません。有効なサンプルは **calculateKPI** メソッドの **samples** パラメータに含まれています。
- また，ルールでは **Sample** オブジェクトによって保持されるサンプル・フィールドを定義するための **sampleFields** フィールドを設定する必要があります。これらの値は，ルールによって使用される値です。

ルール API クラスについては，『HP Rules API Reference』に Javadoc 形式で記載されています。これらのファイルは次のフォルダにあります。
\\<ゲートウェイ・サーバのルート・ディレクトリ>\AppServer\webapps\site.war\amdocs\eng\doc_lib\Service_Health\Rules_API\index.html

サンプル・ルールの詳細な例については，105ページ「例 - API サンプル・ルール」を参照してください。

API ルールは，93ページ「ルール API を使用したルールの作成」の説明に従って，[KPI 定義]ページまたはルール・リポジトリから，あるいはテキスト・ファイル・テンプレートを使用して定義できます。

KPI 定義ページまたはルール・リポジトリでのサンプル・ルールの定義

[KPI 定義]ページまたはルール・リポジトリからサンプル・ルールを定義するには，フィールドに次のとおりに入力します。

- **サンプル・フィールド**：計算に値を含めることができるサンプル・フィールドを示します。サンプル名はカンマで区切ります(たとえば，`"u_iStatus", "dResponseTime"`)。

- **KPI 計算スクリプト** :**calculateKPI** メソッド実装を入力します。メソッド・シグネチャを入力しないでください。このスクリプトでは，**calculateKPI** メソッドのパラメータ **ci**，**kpi**，および **samples** を利用できます。
- **集計計算スクリプト** :**calculateAggregatedKPI** メソッド実装を入力します。メソッド・シグネチャを入力しないでください。このスクリプトでは，**calculateAggregatedKPI** メソッドのパラメータ **ci** および **kpi** を利用できます。
- **サンプル フィルタ スクリプト** :このフィールドには標準設定の **isSampleValid** メソッドの実装が含まれています(標準設定では，すべてのサンプルが計算に含まれます) 。計算からサンプルを除外するには，このフィールドを編集します。

詳細については，94ページ「特定の KPI またはサービス停止用 API ルールの定義方法」または98ページ「ルール・リポジトリでの API ルールの定義方法」を参照してください。

テキスト・ファイルを使用した，サンプル・ルールの定義

テキスト・ファイル・テンプレートを使用してサンプル・ルールを定義するには，**SlmSampleRuleTemplate.groovy** テンプレート・ファイルを使用し，95ページ「テキスト・ファイル・ベースの API ルールの作成方法」の説明に従います。

テキスト・ファイルに **sampleFields** フィールド，**calculateKPI** メソッド本文，および **calculateAggregatedKPI** メソッド本文を定義します。

isSampleValid メソッドには true を返す標準設定の実装が含まれています(すべてのサンプルが計算に含まれます) 。上書きするには，メソッドのコメントアウトを解除して，実装を入力します。

API 継続時間ベースのサンプル・ルール

継続時間ベースのサンプル・ルールでは，サンプル・フィールドから取得した集計データに基づいて KPI を計算します。継続時間ベースのルールでは，ルールの計算時に各サンプルの継続時間を使用します(経過時間ごとの可用性や経過時間ごとのバックログなど)。

継続時間ベースのサンプル・ルールを使用する場合の詳細については，84ページ「サンプル・ルールまたは継続時間ベースのサンプル・ルールを使用する場合」を参照してください。ルールの計算方法の詳細については，81ページ「サンプル・ルールの計算メカニズム - 概要」を参照してください

継続時間ベースのサンプル・ルールのメソッドおよびフィールド

継続時間ベースのサンプル・ルールでは，次のメソッドを含むルール API インタフェース **SlmSamplesTimeBasedAggregatedCalculator** を実装します。

```
public void calculateKPI(CI ci, SlmKPI kpi, List<サンプル> samples)

public void calculateAggregatedKPI(CI ci, SlmKPI kpi)

public boolean isSampleValid(CI ci, SlmKPI kpi, Sample sample)

public boolean isSampleAndDurationValid(CI ci, SlmKPI kpi, Sample
sample)
```

- **calculateKPI** メソッドは，計算サイクルごとに KPI を計算します。このメソッドにはパラメータ **ci**，**kpi**，および **samples** が含まれています。これらは現在の CI，ルールで値を計算する KPI，およびルール計算に使用するサンプルを表します。
  - **kpi** パラメータのタイプは **SlmKPI** であり，計算結果を設定するために使用できます。
  - **samples** パラメータは **Sample** オブジェクトの **List** で，サンプル・フィールド値とサンプル継続時間を保持します。**samples** パラメータには，90ページ「継続時間ベースのサンプル・ルールによるフィルタリング」の説明のとおり，フィルタ・メカニズムを通過するサンプルが含まれています。前回の計算サイクルで使用した最後のサンプルも，89ページ「継続時間ベースのサンプルの連続性」の説明に従って含めることができます。
- サンプルの継続時間は，サンプルのタイムスタンプから次のいずれかのイベント(先に発生したもの)までの間隔と定義されています。
  - 次のサンプルが **isSampleAndDurationValid** メソッドを使用してフィルタリングされた場合は，次のサンプルのタイムスタンプ
  - サイクル内の次のサンプルのタイムスタンプ
  - サイクルの終わり
- **calculateAggregatedKPI** メソッドは集計 KPI を計算します。このメソッドにはパラメータ **ci**，および **kpi** が含まれています。
- フィルタリングには，90ページ「継続時間ベースのサンプル・ルールによるフィルタリング」で説明のとおり，**isSampleValid** および **isSampleAndDurationValid** メソッドが使用されます。

ルール API クラスについては，『HP Rules API Reference』に Javadoc 形式で記載されています。これらのファイルは次のフォルダにあります。
\\<ゲートウェイ・サーバのルート・ディレクトリ>\AppServer\webapps\site.war\amdocs\eng\doc_lib\Service_Health\Rules_API\index.html

- また，ルールでは **Sample** オブジェクトによって保持されるサンプル・フィールドを定義するための **sampleFields** フィールドを設定する必要があります。これらの値は，ルールによって使用される値です。

継続時間ベースのサンプル・ルールの詳細な例については，110ページ「例 - API 継続時間ベースのサンプル・ルール」を参照してください。

API ルールは，93ページ「ルール API を使用したルールの作成」で説明のとおり，[KPI 定義]ページまたはルール・リポジトリから，あるいはテキスト・ファイル・テンプレートを使用して定義できます。

KPI 定義ページまたはルール・リポジトリでの継続時間ベースのサンプル・ルールの定義

[KPI 定義]ページまたはルール・リポジトリから継続時間ベースのサンプル・ルールを定義するには，フィールドに次のとおりに入力します。

- **サンプル・フィールド**:計算に値を含めることができるサンプル・フィールドを示します。サンプル名はカンマで区切ります(たとえば，`"u_iStatus", "dResponseTime"`)。
- **KPI 計算スクリプト**:**calculateKPI** メソッド実装を入力します。メソッド・シグネチャを入力しないでください。このスクリプトでは，**calculateKPI** メソッドのパラメータ **ci**，**kpi**，および **samples** を利用できます。
- **集計計算スクリプト**:**calculateAggregatedKPI** メソッド実装を入力します。メソッド・シグネチャを入力しないでください。このスクリプトでは，**calculateAggregatedKPI** メソッドのパラメータ **ci** および **kpi** を利用できます。
- [**サンプル フィルタ スクリプト**]と[**サンプルおよび期間フィルタ スクリプト**]:これらのフィールドには標準設定の **isSampleValid** メソッドと **isSampleAndDurationValid** メソッドの実装が含まれています(標準設定では，すべてのサンプルが計算に含まれます)。計算からサンプルを除外するには，これらのフィールドを編集します。

詳細については，94ページ「特定の KPI またはサービス停止用 API ルールの定義方法」または98ページ「ルール・リポジトリでの API ルールの定義方法」を参照してください。

テキスト・ファイルを使用した，継続時間ベースのサンプル・ルールの定義

テキスト・ファイル・テンプレートを使用して継続時間ベースのサンプル・ルールを定義するには，**SlmDurationBasedSampleRuleTemplate.groovy** テンプレート・ファイルを使用し，95ページ「テキスト・ファイル・ベースの API ルールの作成方法」の説明に従います。

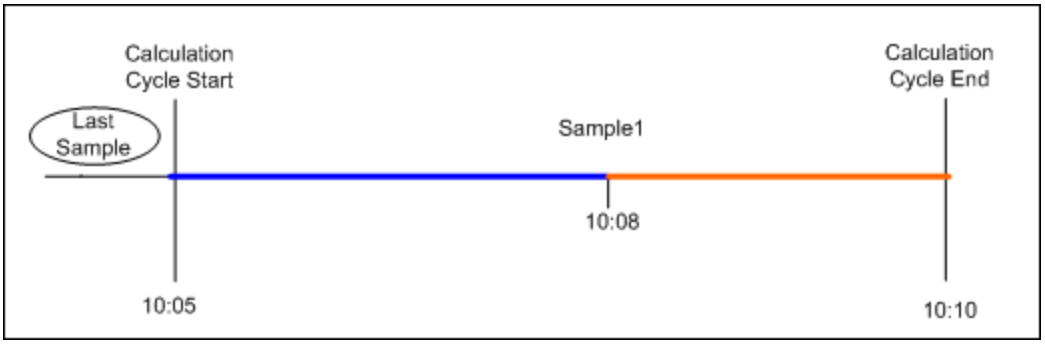
テキスト・ファイルに **sampleFields** フィールド，**calculateKPI** メソッド本文，および **calculateAggregatedKPI** メソッド本文を定義します。

isSampleValid および **isSampleAndDurationValid** メソッドには true を返す標準設定の実装が含まれています(すべてのサンプルが計算に含まれます)。上書きするには，メソッドのコメントアウトを解除して，実装を入力します。

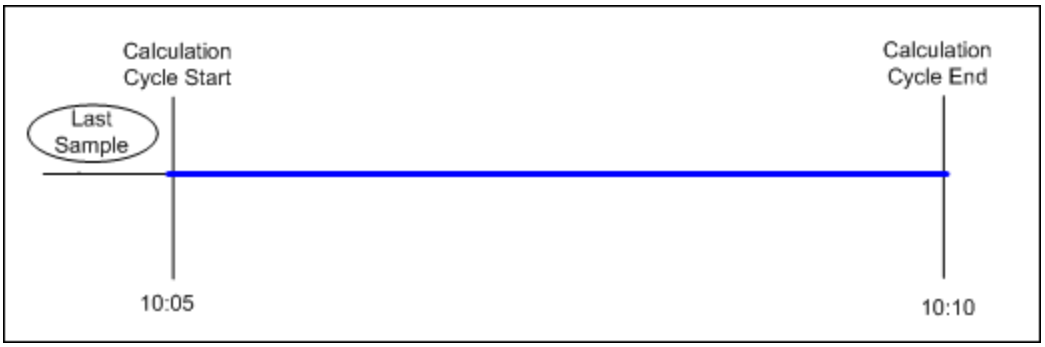
継続時間ベースのサンプルの連続性

継続時間ベースのサンプル・ルールで，(**calculateKPI** メソッドの) **samples** パラメータに含まれる最初のサンプルは，前回のサイクル計算の最後のサンプルになります。これにより，サンプルの値が2つ以上の計算サイクルにわたって利用可能になります。

たとえば，10:05～10:10の計算サイクルに，タイムスタンプが10:08のサンプルがデータベースに1つ格納されています(Sample1)。**samples** パラメータには，次の2つのサンプルが含まれています。前のサイクル(継続時間 = 3分)からの最後のサンプルと現在のサイクル(継続時間 = 2分)からのサンプル。



現在のサイクルにサンプルが含まれていない場合，**samples** パラメータには前回のサイクルの最後のサンプルが含まれます(継続時間 ＝ 5分)。



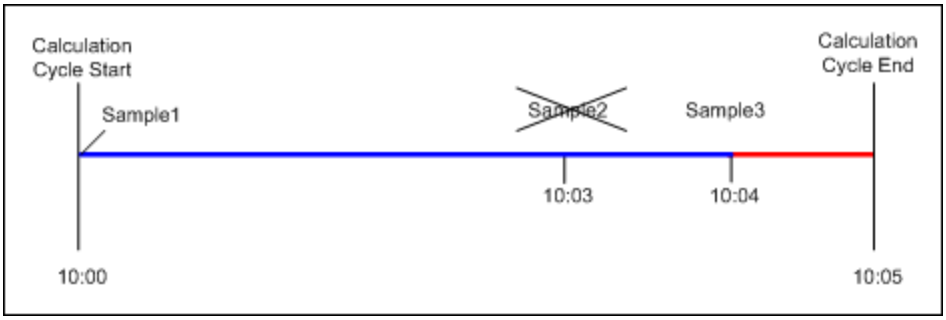
次のいずれか条件が満たされている場合，最後のサンプルは次の計算サイクルには含まれません。

- **データ・タイムアウトなし**：最後のサンプルのタイムスタンプが，データのタイムアウトなしの制限に達した。たとえば，最後のサンプルのタイムスタンプが09:00で，データのタイムアウトなしが1時間に設定されている場合，10:00～10:05の計算サイクル，および以降のすべての計算サイクルにはサンプルは含まれません。
- **ダウンタイム・イベント**：前回のサンプルのタイムスタンプと現在の計算サイクルの間に，ダウンタイム・イベントが発生しました。たとえば，最後のサンプルのタイムスタンプが09:00で，ダウンタイムが10:00～10:30に設定されている場合，このサンプルは10:30～10:35の計算サイクル，および以降のすべての計算サイクルに含まれません。

継続時間ベースのサンプル・ルールによるフィルタリング

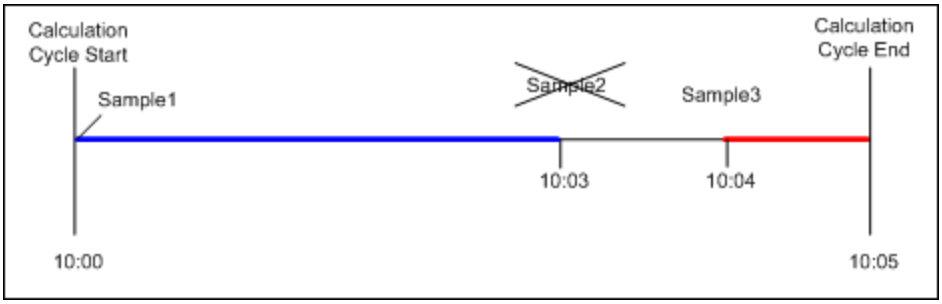
標準設定では，API 継続時間ベースのサンプル・ルールのルール計算では，すべてのサンプルが対象となります。このルールを使用するとき，**isSampleValid** および **isSampleAndDurationValid** メソッドでサンプルをフィルタリングできます。

- **isSampleValid** メソッドを使用すると，前回の有効サンプルの継続時間に，フィルタリングしたサンプルの継続時間が加算されます。



上記の例で，Sample2 は **isSampleValid** メソッドを使用してフィルタリングされます。**samples** パラメータには，継続時間が4分の Sample1 と，継続時間が1分の Sample3 が含まれます。このサイクル内のすべてのサンプルの総継続時間は，5分です。

- **isSampleAndDurationValid** メソッドを使用すると，フィルタリングしたサンプルの継続時間は前回の有効サンプルの継続時間に加算されないため，計算に含まれません。



上記の例で，Sample2 は **isSampleAndDurationValid** メソッドを使用してフィルタリングされます。**samples** パラメータには，継続時間が3分の Sample1 と，継続時間が1分の Sample3 が含まれます。このサイクル内のすべてのサンプルの総継続時間は，4分です。

API サンプルによるサービス停止ルール

API サンプルによるサービス停止ルールでは，サンプルから取得したデータに基づいてサービス停止を計算します。サービス停止は，連続するサンプルで特定数の失敗が連続して算出されると開始され，失敗がないサンプルが計算に使用されると終了します。

『BSM アプリケーション管理ガイド』の「SLM ビジネス・ルール・パラメータのリスト」に記載されているように，ルール・パラメータ(失敗の最小数，最短継続時間，最長継続時間) によりサービス停止の開始時間と終了時間が定義されます。サンプルによるサービス停止ルール・パラメータの例については，116ページ「例 - API サンプルによるサービス停止ルール」を参照してください。

各サンプルには複数の失敗 (ゼロ以上) があることがあります。API サンプルによるサービス停止ルールを使用すると，サンプルが表す失敗数をサンプル値に基づいて計算できます。その後，サービス停止ルール・パラメータを使用してサービス停止が計算されます。

サンプルによるサービス停止ルールのメソッドとフィールド

サンプルによるサービス停止ルールでは，次のメソッドが含まれるルール API インタフェース **OutageBySamplesCalculator** が実装されます。

```
public void calculateOutage(Outage outage, Sample sample)

public boolean isSampleValid(Sample sample)
```

- **calculateOutage** メソッドは **Sample** で表される失敗数を計算します。このメソッドにはパラメータ **outage** および **sample** が含まれます。
  - **outage** パラメータのタイプは**サービス停止**であり，特定の **sample** の失敗数を設定する場合に使用します。
  - **sample** パラメータのタイプは，サンプル・フィールド値を保持する**サンプル**です。
- **isSampleValid** メソッドはサンプルをフィルタリングする場合に使用します。無効なサンプル値は **calculateOutage** メソッドの計算には含まれません。

  ルール API クラスについては，『HP Rules API Reference』に Javadoc 形式で記載されています。これらのファイルは次のフォルダにあります。
  \\<ゲートウェイ・サーバのルート・ディレクトリ>\AppServer\webapps\site.war\amdocs\eng\doc_lib\Service_Health\Rules_API\index.html
- また，ルールでは **Sample** オブジェクトによって保持されるサンプル・フィールドを定義するための **sampleFields** フィールドを設定する必要があります。これらの値は，ルールによって使用される値です。

サンプルによるサービス停止ルールの例については，116ページ「例 - API サンプルによるサービス停止ルール」を参照してください。

API ルールは，93ページ「ルール API を使用したルールの作成」で説明のとおり，[KPI 定義] ページまたはルール・リポジトリから，あるいはテキスト・ファイル・テンプレートを使用して定義できます。

KPI 定義ページまたはルール・リポジトリでのサンプルによるサービス停止ルールの定義

[KPI 定義] ページまたはルール・リポジトリからサンプルによるサービス停止ルールを定義するには，フィールドに次のとおりに入力します。

- **サンプル・フィールド** :計算に値を含めることができるサンプル・フィールドを示します。サンプル名はカンマで区切ります(`"u_iStatus"` など) 。
- **サービス停止計算スクリプト** :**calculateOutage** メソッド実装を入力します。メソッド・シグネチャを入力しないでください。このスクリプトでは，**calculateOutage** メソッドのパラメータ **outage** と **sample** を利用できます。
- **サンプル フィルタ スクリプト** :このフィールドには標準設定の **isSampleValid** メソッドの実装が含まれています(標準設定では，すべてのサンプルが計算に含まれます) 。計算からサンプルを除外するには，このフィールドを編集します。

詳細については，94ページ「特定の KPI またはサービス停止用 API ルールの定義方法」または98ページ「ルール・リポジトリでの API ルールの定義方法」を参照してください。

テキスト・ファイルを使用したサンプルによるサービス停止ルールの定義

テキスト・ファイル・テンプレートを使用してサンプルによるサービス停止ルールを定義するには，**SlmOutageBySampleTemplate.groovy** テンプレート・ファイルを使用し，95ページ「テキスト・ファイル・ベースの API ルールの作成方法」の説明に従います。

テキスト・ファイルに **sampleFields** フィールドおよび **calculateOutage** メソッドの本文を定義します。

isSampleValid メソッドには true を返す標準設定の実装が含まれています(すべてのサンプルが計算に含まれます) 。上書きするには，メソッドのコメントアウトを解除して，実装を入力します。

ルール API を使用したルールの作成

次の項で説明するように，ルール API を使用してルールを作成するには，いくつかの方法があります。

注: 次の3つの方法では，API サンプルによるサービス停止ルールを使用し，サービス停止の計算を定義することもできます。

特定の KPI のルール定義

サービス・レベル管理 KPI には，次の API ルールを含めることができます。API グループと兄弟ルール，API サンプル・ルール，API 継続時間ベースのサンプル・ルール。

[**新規 SLA ウィザード**/**SLA の編集ウィザード**]＞[**SLA インジケータの設定**]ページを使用すると，KPI に API ルールを1つ割り当て，ルールの詳細を入力して KPI のルール・ロジックを定義できます。

その後はいつでも[**SLA インジケータの設定**]ページでルールの詳細を編集して，KPI のルール・ロジックを変更できます。

詳細は，94ページ「特定の KPI またはサービス停止用 API ルールの定義方法」を参照してください。

テキスト・ファイルを使用した複数の KPI 用のルールの作成

各 API ルールには，**<データ処理サーバのルート・ディレクトリ>\BLE\rules\groovy\templates** ディレクトリに対応するテンプレート・ファイルがあります。いずれかのテンプレート・ファイルを使用して，新しいルールを定義するテキスト・ファイルを作成できます。作成したルールをルール・リポジトリに追加すると，標準で含まれているルールと同じように適用できます。

API コードはテキスト・ファイル内でだけ表示および変更できます。サービス・レベル管理内では表示も変更もできません。テキスト・ファイル内でコードに変更を加えた場合は，オフライン・エンジンを再起動した後で，ルールが割り当てられているすべてのインスタンスに変更内容が適用されます。

詳細は，95ページ「テキスト・ファイル・ベースの API ルールの作成方法」を参照してください。

ルール・リポジトリへのルールの定義

ルール・リポジトリには，次の API ルールが含まれています。API グループと兄弟ルール，API サンプル・ルール，API 継続時間ベースのサンプル・ルール。ルール・リポジトリから，ある API ルールを複製し，ルールの詳細を入力してルール・ロジックを定義できます。

KPI にルールを適用したら，いつでも[**SLA インジケータの設定**]ページでルールの詳細を編集し，特定の KPI のルール・ロジックを変更できます。

詳細は，98ページ「ルール・リポジトリでの API ルールの定義方法」を参照してください。

特定の KPI またはサービス停止用 API ルールの定義方法

各 KPI またはサービス停止には適用可能な API ルールがあります。[KPI 定義]ページを使用すると，それらの API ルールの 1 つを KPI またはサービス停止に割り当てて，ルールの詳細を入力してルール・ロジックを定義できます。

1. **KPI またはサービス停止への API ルールの割り当て**

   - CI に割り当てられた特定の KPI に API ルールを割り当てるには，[**アグリーメント マネージャ**] > [**新規 SLA ウィザード/SLA の編集ウィザード**] > [**SLA インジケータの設定**]ページから SLA を編集します。[**KPI の追加**]を選択して，新しい KPI を CI に割り当てるか，または既存の KPI を変更します。詳細については，『BSM アプリケーション管理ガイド』の「[CI の KPI を追加]/[CI の KPI を編集]ダイアログ・ボックス」を参照してください。

     適用可能なビジネス・ルールのリストで，次の API ルールを 1 つ選択します。API グループと兄弟ルール，API サンプル・ルール，または API 継続時間ベースのサンプル・ルール。(API サンプル・ルールおよび API 継続時間ベースのサンプル・ルールは，モニタ CI にのみ適用できます)。ルール・タイプの説明については，75ページ「SLM ルール API」を参照してください。

   - CI のサービス停止に API サービス停止ルールを割り当てるには，[**アグリーメント マネージャ**] > [**新規 SLA ウィザード/SLA の編集ウィザード**] > [**SLA インジケータの設定**]ページから SLA を編集します。詳細については，『BSM アプリケーション管理ガイド』の「[サービス停止 KPI の追加]/[サービス停止 KPI の編集]ダイアログ・ボックス」を参照してください。

     [編集]ボタンをクリックして，サービス停止を編集します。適用可能なビジネス・ルール一覧から，API サンプルによるサービス停止ルールを選択します(このルールはモニタ CI にのみ適用できます)。

2. **KPI またはサービス停止のルール・ロジックの定義**

   作成しているルールのタイプに応じて，次の説明に従って，ルールのメソッドとフィールドを定義します。

   - 78ページ「API グループと兄弟ルール」
   - 85ページ「API サンプル・ルール」
   - 87ページ「API 継続時間ベースのサンプル・ルール」
   - 91ページ「API サンプルによるサービス停止ルール」

テキスト・ファイル・ベースの API ルールの作成方法

API ルールに対応するルール・テンプレート・ファイルがあります。各テンプレートによりルールのインタフェースが実装されます。

いずれかのテンプレートを使用して新しいルールを定義するテキスト・ファイルを作成し，その新しいルールをビジネス・ルール・リポジトリに追加します。追加したルールは，標準設定で含まれているルールと同じように適用可能になります。

API コードはテキスト・ファイル内でだけ表示および変更できます。サービス・レベル管理内では表示も変更もできません。テキスト・ファイル内でコードに変更を加えた場合は，オフライン・エンジンを再起動した後で，ルールが割り当てられているすべてのインスタンスに変更内容が適用されます。

1. ## ルールのテキスト・ファイルの作成

   作成するルールのタイプに基づいて，**<データ処理サーバのルート・ディレクトリ>\BLE\rules\groovy\templates** ディレクトリに置かれているテンプレート・ファイルのいずれかをコピーし，その名前を変更します。

   テンプレートのコピー内で，次の説明に従って，ルールのメソッドとフィールドを定義します。

   - 78ページ「API グループと兄弟ルール」
   - 85ページ「API サンプル・ルール」
   - 87ページ「API 継続時間ベースのサンプル・ルール」
   - 91ページ「API サンプルによるサービス停止ルール」

   ファイルを **<データ処理サーバのルート・ディレクトリ>\BLE\rules\groovy\rules** ディレクトリに保存します。

   ここで，テキスト・ファイルのルール・ロジックを使用するルールを，ルール・リポジトリに追加する必要があります。

   - サービス停止以外のルールの場合は，95ページ「ルール・リポジトリにルールの追加 (グループと兄弟ルール，サンプル・ルールまたは継続時間ベースのサンプル・ルールの場合) 。」の手順に従ってください。
   - サービス停止ルールの場合は，96ページ「ルール・リポジトリへのサービス停止ルールの追加」の手順に従ってください。

2. ## ルール・リポジトリにルールの追加 (グループと兄弟ルール，サンプル・ルールまたは継続時間ベースのサンプル・ルールの場合) 。

   a. [**管理**] > [**サービス レベル管理**] > [**リポジトリ**] > [**ビジネス ルール**] > [**新規ルール**]を選択します。ルール追加の詳細については，『BSM アプリケーション管理ガイド』の「リポジトリのビジネス・ルール・テンプレートのカスタマイズ方法」を参照してください。

   b. [**名前**]フィールドに，作成するルールの名前を入力します(必須) 。

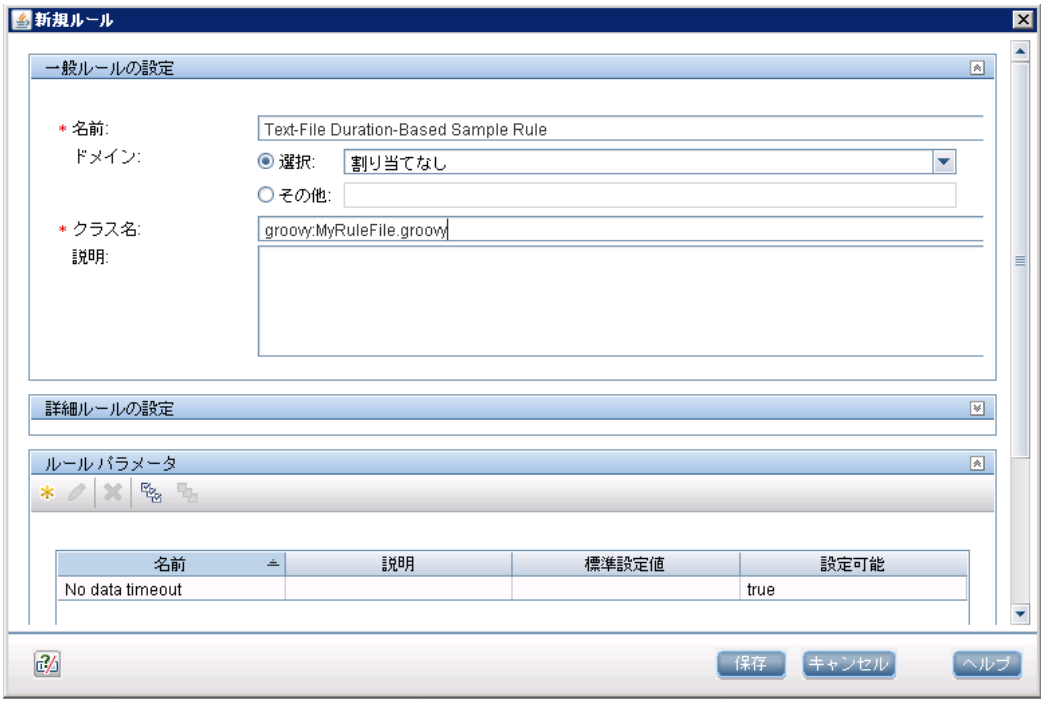
   c. [**クラス名**]フィールドに，「**groovy:**<ファイル名 >」と入力します。ファイル名は **<データ処理サーバのルート・ディレクトリ>\BLE\rules\groovy\rules** ディレクトリの名前と同一 (大文字と小文字を区別) にする必要があります。

   d. API 継続時間ベースのサンプル・ルールには，次のルール・パラメータを作成します。

- [**ルール パラメータ**]領域で，[**新規ルール パラメータ**]をクリックします。
- [**名前**]フィールドに「**データ タイムアウトなし**」と入力します。[**タイプ**]フィールドで[**Long**]を選択します。[**標準設定値**]フィールドに「**3600**」と入力します。

e. [**OK**]をクリックして保存します。

次の図に，ルール・パラメータの追加後の継続時間ベースのサンプル・ルールを示します。



3. ルール・リポジトリへのサービス停止ルールの追加

a. [**管理**]＞[**サービス レベル管理**]＞[**リポジトリ**]＞[**ビジネス ルール**]で，[**合成 WS 停止**]ルール(316)を選択してルールを複製します。

b. [**名前**]フィールドで，ルールの名前を変更します(必須)。

c. [**クラス名**]フィールドに，「**groovy:**<ファイル名>」と入力します。ファイル名は **<データ処理サーバのルート・ディレクトリ>\BLE\rules\groovy\rules** ディレクトリの名前と同一(大文字と小文字を区別)にする必要があります。

d. [**説明**]は編集できますが，ほかのフィールドは変更しないでください。

e. [OK]をクリックして保存します。

4. KPI またはサービス停止に適用可能なルール一覧へのルールの追加

関連する KPI にすでにアタッチされている適用可能なルールのリストに，新しいルールを追加します。サービス停止ルールの該当の KPI は **サービス停止** KPI(200) です。

詳細については，『BSM アプリケーション管理ガイド』の「[新規 KPI]/[KPI の編集]ダイアログ・ボックス」の「[メイン設定]領域」に記載されている「適用可能なルールのパラメータ」を参照してください。

5. テキスト・ファイル編集後のオフライン・エンジンの再起動

ルールを作成した後でテキスト・ファイルに変更を加えた場合は，次の手順を実行して，変更内容を適用します。

a. ブラウザで，次のように入力します。

```
http://<Business Service Management データ処理サーバ名>:11021
```

b. [**ファウンデーション: NannyManager**]を選択します。

c. **showServiceInfoAsHTML** を起動し，オフライン・エンジンを再起動します。

ルール・リポジトリでの API ルールの定義方法

ビジネス・ルール・リポジトリに，複数の KPI またはサービス停止に適用できる API ルールを作成します。このためには，4 つの API ルールのいずれかを複製し，特定のルール・パラメータに標準設定のルール値を設定します。KPI またはサービス停止にルールを適用したら，特定の KPI またはサービス停止のルール・ロジックを[KPI 定義]ページからスクリプトを編集することによって，いつでも変更できます。

1. **API ルールの複製**

   [**管理**]＞[**サービス レベル管理**]＞[**リポジトリ**]＞[**ビジネス ルール**]を選択します。[ビジネス ルール リポジトリ]ページで，次のルールのうちいずれかを複製します。API グループと兄弟ルール，API サンプル・ルール，API 継続時間ベースのサンプル・ルール，API サンプルによるサービス停止ルール。

   ルールを複製する方法の詳細については，『BSM アプリケーション管理ガイド』の「ビジネス・ルール・テンプレートのカスタマイズ方法」を参照してください。

2. **ルールの詳細の編集**

   a. 編集する新しいルールを開きます。

   b. [**名前**]フィールドで，複製したルールの名前を変更します。

   c. [**ルール パラメータ**]で，次の説明に従って，各ルール・パラメータに[**標準設定値**]を設定してルール・ロジックを定義します。

      - 78ページ「API グループと兄弟ルール」
      - 85ページ「API サンプル・ルール」
      - 87ページ「API 継続時間ベースのサンプル・ルール」
      - 91ページ「API サンプルによるサービス停止ルール」

      たとえば，[**KPI 計算スクリプト**]を定義するには，[**KPI 計算スクリプト**]ルール・パラメータを編集します。[**標準設定値**]フィールドに，ルール計算スクリプトを入力します。入力したコードがこのルールの標準設定のコードになり，このルールが割り当てられたすべての KPI の[KPI 定義]ページに表示されます(ほかのフィールドは変更しないでください) 。

3. **KPI またはサービス停止に適用可能なルール一覧へのルールの追加**

   関連する KPI にすでにアタッチされている適用可能なルールのリストに，新しいルールを追加します。サービス停止ルールの該当の KPI は **サービス停止** KPI(200) です。

   詳細については，『BSM アプリケーション管理ガイド』の「[新規 KPI]/[KPI の編集]ダイアログ・ボックス」の「[適用可能なルール]GUI パラメータ」を参照してください。

ツールチップ・エントリの使用方法

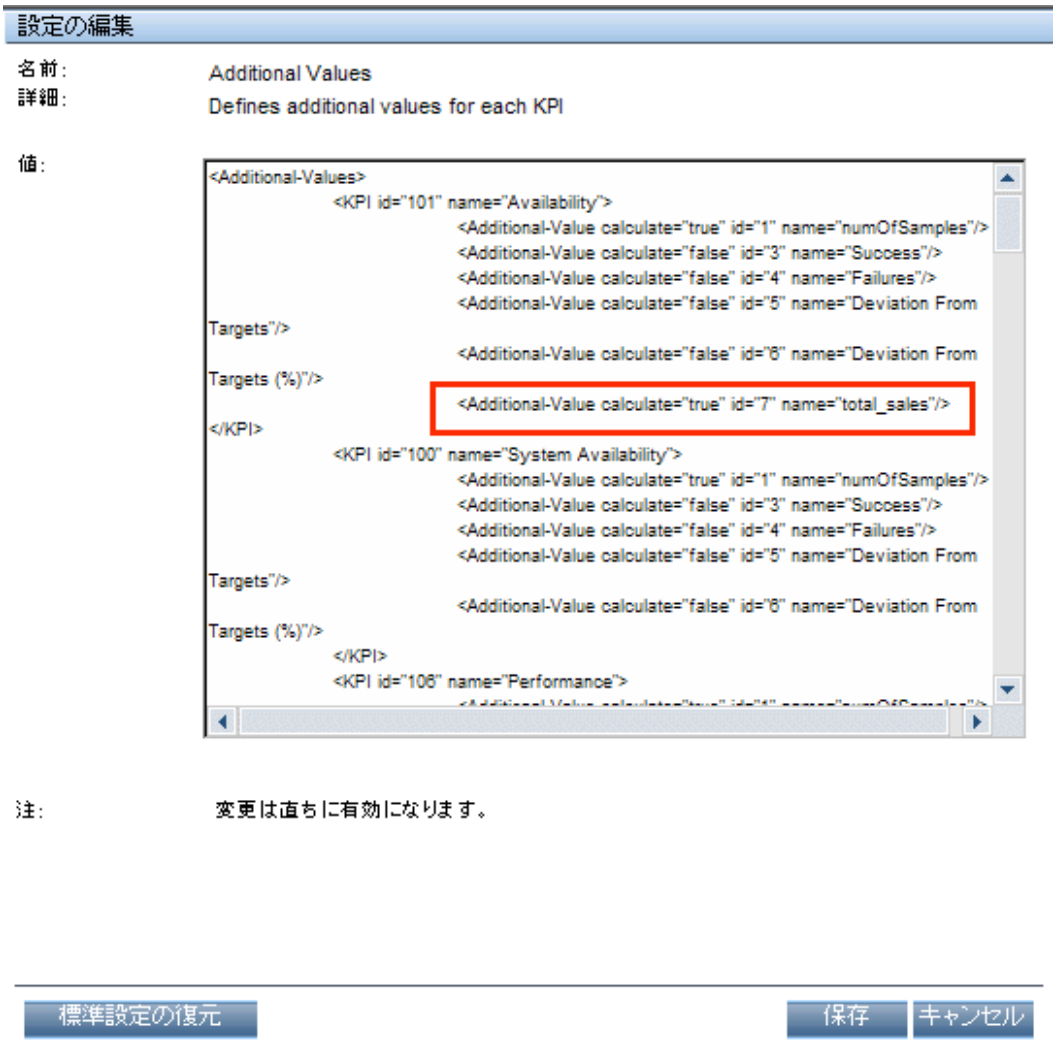
kpi.setTooltip メソッドを使用している場合，インフラストラクチャ設定で対応するツールチップ・エントリを設定する必要があります。

1. [**管理**]＞[**プラットフォーム**]＞[**セットアップと保守**]＞[**インフラストラクチャ設定**]＞[**サービス・レベル管理**]を選択します。
2. 次の手順に従って[**追加値**]を編集します。
   a. ツールチップを追加する KPI にスクロール・ダウンします。
   b. KPI の最後のエントリの後に，次の形式の行を追加します。

      <Additional-Value calculate="true" id="<ID>" name="<名前>"/>

      上記の形式では <ID> は現在の最大 ID よりも 1 大きい数で(最大 ID が 5 の場合は 6 と入力)，<名前> はコードで使用しているのと同一のツールチップ名です。

      たとえば，コードにメソッド呼び出し **kpi.setTooltip("total_sales", value)** が含まれていて，KPI が**可用性**である場合は，次のように入力します。



c. [**保存**]をクリックします。

ルール API コードからログ・ファイルに書き込む方法

API ルールで，**logger** オブジェクトを使用してルール・メソッドからログ・ファイルに書き込むことができます。ログには，debug，info，warn，error，fatal という 5 つのレベルがあります。これらのレベルそれぞれで，特定の logger メソッドを使用します。

標準設定では，重大度が「error」と「fatal」のログ・メソッド呼び出しだけが，ログ・ファイルに書き込まれます。この設定は，ログ設定ファイル内で変更できます。

ルール API を使用してログ・ファイルに書き込むには，次の手順を実行します。

1. ルール・メソッド内で，次のいずれかのメソッドを実装します(メソッドは重大度の順にリストされています) 。
   - **logger.debug("<API ルール名> :log message");**
   - **logger.info("<API ルール名> :log message");**
   - **logger.warn("<API ルール名> :log message");**
   - **logger.error("<API ルール名> :log message");**
   - **logger.fatal("<API ルール名> :log message");**

   API ルールの名前をログ・メッセージ内に入力して，各ログ・メッセージをそのソース・ルールで識別します。
2. ルール API ログ・ファイルは，**<データ処理サーバのルート・ディレクトリ>\HPBSM\log\offline_engine\RulesAPI** ディレクトリにあります。

   ルール・タイプに応じて，次のいずれかのファイルを開いて，ログ・メッセージを表示します。
   - **groupAndSiblingRule.log**(API グループと兄弟ルールの場合)
   - **sampleRule.log**(API サンプル・ルールと API 継続時間ベースのサンプル・ルールの場合)
   - **OutageRule.log** (API サンプルによるサービス停止ルールの場合)

ログ・ファイルに書き込まれる重大度レベルを変更するには，次の手順を実行します。

1. 標準設定では，重大度が「error」と「fatal」のログ・メソッド呼び出しだけが，ログ・ファイルに書き込まれます。この設定を変更するには，**<データ処理サーバのルート・ディレクトリ> HPBSM\conf\core\Tools\log4j\offline_engine\slm_rules.properties** にあるログ設定ファイルを開きます。
2. ルール・タイプに対応する行で，文字列 **${loglevel}** をログ記録する重大度レベルに置き換えます(**DEBUG**，**INFO**，**WARN**，**ERROR**，**FATAL** のうちいずれかを入力する) 。ルール・タイプに応じて，次のいずれかの行を編集します。
   - グループと兄弟ルールの場合
     :log4j.category.com.mercury.am.bac.slm.rules.group.common.SlmGroupAndSiblingRule = **${loglevel}**, slm.rules.api.group.appender
   - サンプル・ルールと継続時間ベースのサンプル・ルール
     :log4j.category.com.mercury.am.bac.slm.rules.leaf.simplified.SlmSimplifiedLeafRule = **${loglevel}**, slm.rules.api.sample.appender

- サンプルによるサービス停止ルール
  :log4j.category.com.mercury.am.bac.slm.rules.outages.simplified.SimplifiedOutageRule = **${loglevel}**, slm.rules.api.outage.appender

ルール API 計算に CI プロパティを含める方法

API ルール内で，CI クラス **getPropertyValue** メソッドと KPI クラス **getCiProperty** メソッドを使用して CI プロパティを含めることができます。SLM ルールでは，修飾子 **BLE_ATTRIBUTE** を持つ CI プロパティにのみアクセス可能です。

この属性を CI クラスに追加するには，クラスをエクスポートしてから定義を編集し，サーバにインポートし直す必要があります。エクスポートしたクラスを開いて編集するには，次の xml を必要な属性に追加します。

```
<Attribute name="<attribute-name>" type="double" display-name="<attribute-display-name>">
  <Attribute-Qualifiers>
    <Attribute-Qualifier name="BLE_ATTRIBUTE"/>
  </Attribute-Qualifiers>
</Attribute>
```

例 - API グループと兄弟ルール

サービス・レベル管理とサービス状況は，両方とも API グループと兄弟ルールを計算する同じインタフェースを実装しています。

API グループと兄弟ルールを示す詳細な例については，45ページ「例 - API グループと兄弟ルール」を参照してください。

Status クラスで使用するステータスは，サービス・レベル管理およびサービス状況では異なります。たとえば，サービス状況のコードの `Status.OK` は，サービス・レベル管理のコードの `Status.EXCEEDED` と同じです。次の表ではステータスの対応を示します。

サービス状況のステータス	サービス・レベル管理のステータス
OK	超過
警告	適合
軽微	軽微な違反
重大	違反
致命的	失敗

例 - API サンプル・ルール

この項では，API サンプル・ルールの例を示します。次の例について説明します。

例 - サンプルベースの平均応答時間ルール

次のルールでは，dResponseTime サンプル・フィールドに基づいて平均応答時間を計算します。ルールの結果(集計値)は，該当の計算期間のすべてのサンプルに基づいて計算された平均応答時間です。

ルールのロジックは，(合計応答時間(秒数)/合計サンプル数)です。

このルールでは，**SlmKPI** キーおよび集計キーを使用して，合計応答時間と合計サンプル数を集計し，ルールの結果を計算します。

```
// Define the sample fields that will be used in the rule
calculation. def sampleFields = ["dResponseTime"];

/* * Implementation of the SlmSamplesAggregatedCalculator interface
method.* For more information refer to the Rules API
documentation.*/ public void calculateKPI(CI ci, SlmKPI kpi,
List<Sample> samples) { /* * Sum the field dResponseTime from all
given samples.This represents the total * response time for all the
samples in the current calculation cycle.*/ def totalTime =
Utils.sumField(samples, "dResponseTime"); /* * Keep the total
response time, converted to seconds, on a kpi key named *
totalResponseTime.This key is used by the calculateAggregatedKPI
method.    */ kpi.key.totalResponseTime = Utils.divide(totalTime,
1000.0); /* * Keep the number of samples for the current
calculation cycle on a kpi key named * totalSamples.This key is
used by the calculateAggregatedKPI method.    */
kpi.key.totalSamples = samples.size(); }

/* * Implementation of the SlmSamplesAggregatedCalculator interface
method.*/ public void calculateAggregatedKPI(CI ci, SlmKPI kpi) {
/* * Keep the aggregated response time on a kpi aggregated key
named * totalResponseTime, by adding the current aggregated
totalResponseTime from the * kpi aggregated key, and the current
calculation cycle response time taken from the * kpi key named
totalResponseTime (calculated in the calculateKPI method).*/
kpi.aggregatedKey.totalResponseTime = Utils.sum
(kpi.key.totalResponseTime, kpi.aggregatedKey.totalResponseTime)./*
* Keep the aggregated total samples on a kpi aggregated key named
totalSamples, * by adding the current aggregated total samples from
the kpi aggregated key, and * the current calculation cycle total
```

```
samples, taken from the kpi key named * totalSamples (calculated in
the calculateKPI method).*/ kpi.aggregatedKey.totalSamples =
Utils.sum(kpi.key.totalSamples, kpi.aggregatedKey.totalSamples) /*
* Set the aggregated value of the KPI by dividing the two
aggregated values.* This aggregated value will be displayed in the
SLA reports.*/ kpi.aggregatedValue = Utils.divide
(kpi.aggregatedKey.totalResponseTime,
kpi.aggregatedKey.totalSamples) }
```

計算 - サンプルベースの平均応答時間ルール

次では，サンプルベースの平均応答時間ルールの計算例を示します。10:00～11:00 の間に，9 つのサンプルを受信しました。

10:00 {サンプル・フィールド :dResponseTime = 60}
10:04:00 {サンプル・フィールド :dResponseTime = 30}
10:11:00 {サンプル・フィールド :dResponseTime = 30}
10:25:00 {サンプル・フィールド :dResponseTime = 25}
10:27:00 {サンプル・フィールド :dResponseTime = 35}
10:35:00 {サンプル・フィールド :dResponseTime = 10}
10:36:00 {サンプル・フィールド :dResponseTime = 20}
10:38:00 {サンプル・フィールド :dResponseTime = 30}
10:52:00 {サンプル・フィールド :dResponseTime = 75}

計算サイクルはそれぞれ 5 分長です。ルール計算は次のとおりです。

サイクル	calculateKPI で設定されるキーと値		calculateAggregatedKPI で設定される集計キーと値		
	totalResponseTime	totalSamples	totalResponseTime	totalSamples	Aggregated Value
10:00-10:05	90	2	90	2	45
10:05-10:10	Null	0	90	2	45
10:10-10:15	30	1	120	3	40
10:15-10:20	Null	0	120	3	40
10:20-10:25	Null	0	120	3	40
10:25-10:30	60	2	180	5	36

サイクル	calculateKPI で設定されるキーと値		calculateAggregatedKPI で設定される集計キーと値		
10:30-10:35	Null	0	180	5	36
10:35-10:40	60	3	240	8	30
10:40-10:45	Null	0	240	8	30
10:45-10:50	Null	0	240	8	30
10:50-10:55	75	1	315	9	35
10:55-11:00	null	0	315	9	35
					最終結果: 35

例 - フィルタを使用したサンプルベースの平均応答時間ルール

このルールは前のルール(105ページ「例 - サンプルベースの平均応答時間ルール」) と同じですが，サンプル・フィルタが追加されています。

サンプル・フィールドのコードは，次のとおりです。

```
// This rule uses the dResponseTime sample field and u_iStatus
sample field. def sampleFields = ["dResponseTime", "u_iStatus"];
```

このルールには，次のメソッドの追加もあります。

```
/* * Override the default implementation of the
SlmSamplesAggregatedCalculator
* interface method.* * Filter samples that hold u_iStatus field
value which is not 0 or 2. */ public boolean isSampleValid(Sample
sample) { //Get the value of the sample's u_iStatus field. def
avialFieldValueObj = sample.u_iStatus; return (avialFieldValueObj
== 0 || avialFieldValueObj == 2) }
```

計算 - フィルタを使用したサンプルベースの平均応答時間ルール

次では，フィルタを使用したサンプルベースの平均応答時間ルールの計算例を示します。

10:00～11:00 の間，プロファイル・データベースには次の9つのサンプルがあります。

10:00 {サンプル・フィールド :dResponseTime = 60, u_iStatus = 0}
10:04 {サンプル・フィールド :dResponseTime = 30, u_iStatus = 1}
10:11 {サンプル・フィールド :dResponseTime = 30, u_iStatus = 2}
10:25 {サンプル・フィールド :dResponseTime = 25, u_iStatus = 1}
10:27 {サンプル・フィールド :dResponseTime = 35, u_iStatus = 0}
10:35 {サンプル・フィールド :dResponseTime = 10, u_iStatus = 1}
10:36:00 {サンプル・フィールド :dResponseTime = 20, u_iStatus = 0}
10:38:00 {サンプル・フィールド :dResponseTime = 30, u_iStatus = 2}
10:52 {サンプル・フィールド :dResponseTime = 75, u_iStatus = 1}

太字のサンプルは，フィルタを通過しません。次の5つのサンプルが計算に含まれます。

10:00 {サンプル・フィールド :dResponseTime = 60, u_iStatus = 0}
10:11:00 {サンプル・フィールド :dResponseTime = 30, u_iStatus = 2}
10:27:00 {サンプル・フィールド :dResponseTime = 35, u_iStatus = 0}
10:36:00 {サンプル・フィールド :dResponseTime = 20, u_iStatus = 0}
10:38:00 {サンプル・フィールド :dResponseTime = 30, u_iStatus = 2}

計算結果は(6 + 30 + 35 + 20 + 30)/5 = 35となります。

例 - サンプルベースの最大応答時間ルール

次のルールでは，dResponseTime サンプル・フィールドに基づいて最大応答時間を計算します。ルールの結果(集計値)は，該当の計算期間のすべてのサンプルに基づいて計算された最大応答時間です。このルールでは **SlmKPI** 値を使用して，各計算サイクルの最大値を保持します。

```
// Define the sample fields that will be used in the rule
calculation. def sampleFields = ["dResponseTime"];

/* * Implementation of the SlmSamplesAggregatedCalculator interface
method.For more information refer to the Rules API documentation.*/
public void calculateKPI(CI ci, SlmKPI kpi, List<Sample> samples)
{         /**         * Find the maximum value of the dResponseTime
field from all given samples,         * and set it as the KPI value
for the current calculation cycle.        */         kpi.value =
Utils.getMax(samples, "dResponseTime"); }     /**      *
Implementation of the SlmSamplesAggregatedCalculator interface
method.    */     public void calculateAggregatedKPI(CI ci, SlmKPI
kpi) {         /**         * Keep the aggregated maximum response
time on the KPI aggregated value,         * by replacing the
current aggregated value with the maximum between the KPI         *
aggregated value and the KPI value (calculated in the calculateKPI
method).        * This aggregated value will be displayed in the
SLA reports.        */         kpi.aggregatedValue = Utils.max
(kpi.value, kpi.aggregatedValue) }
```

計算 - サンプルベースの最大応答時間ルール

次では，サンプルベースの最大応答時間ルールの計算例を示します。

10:00～11:00 の間に，9 つのサンプルを受信しました。

10:00 {サンプル・フィールド :dResponseTime = 60}
10:04:00 {サンプル・フィールド :dResponseTime = 30}
10:11:00 {サンプル・フィールド :dResponseTime = 30}
10:25:00 {サンプル・フィールド :dResponseTime = 25}
10:27:00 {サンプル・フィールド :dResponseTime = 35}
10:35:00 {サンプル・フィールド :dResponseTime = 10}
10:36:00 {サンプル・フィールド :dResponseTime = 20}
10:38:00 {サンプル・フィールド :dResponseTime = 30}
10:52:00 {サンプル・フィールド :dResponseTime = 75}

ルールの計算結果は 75 です。

例 - API 継続時間ベースのサンプル・ルール

このセクションでは，API 継続時間ベースのサンプル・ルールの例を示します。次の例について説明します。

- 110ページ「例 - 継続時間ベースの平均応答時間ルール」
- 112ページ「例 - isSampleValid メソッド・フィルタを使用した継続時間ベースの平均応答時間ルール」
- 113ページ「例 - isSampleAndDurationValid メソッド・フィルタを使用した継続時間ベースの平均応答時間ルール」
- 114ページ「例 - isSampleAndDurationValid とisSampleValid メソッド・フィルタを使用した継続時間ベースの平均応答時間ルール」

例 - 継続時間ベースの平均応答時間ルール

次のルールでは，該当の計算期間のすべてのサンプルの，dResponseTime サンプル・フィールドとサンプル継続時間に基づく加重平均応答時間を計算します。

ルールのロジックは，(サンプル応答時間 * サンプル継続時間) の合計/サンプル継続時間の合計です。

このルールでは **SlmKPI** キーおよび集計キーを使用して，加重応答時間の合計とサンプル継続時間の合計を集計して，ルールの結果を計算します。

```
// Define the sample fields that will be used in the rule
calculation. def sampleFields = ["dResponseTime"];

/* * Implementation of the SlmSamplesTimeBasedAggregatedCalculator
interface method.* For more information refer to the Rules API
documentation.*/ public void calculateKPI(CI ci, SlmKPI kpi,
List<Sample> samples) {     /**     * Set the KPI key totalDuration
to sum duration of all given samples.     * This key is used by the
calculateAggregatedKPI method.     */     kpi.key.totalDuration =
Utils.sumDuration(samples)     // Iterate over all samples that
arrived in the current calculation cycle.     samples.each {Sample
sample ->         /**         * Calculate weighted response time
for each sample by multiplying         * sample duration with
sample dResponseTime field value.         */         def
weightedResponseTime = Utils.multiply(sample.duration,
sample.dResponseTime);         /**         * Keep the total
weighted response time for all given samples         * on a KPI key
named totalResponseTime.         * This key is used by the
calculateAggregatedKPI method.         */
        kpi.key.totalResponseTime = Utils.sum
(kpi.key.totalResponseTime, weightedResponseTime)         } }

/* * Implementation of the SlmSamplesTimeBasedAggregatedCalculator
interface method.*/ public void calculateAggregatedKPI(CI ci,
SlmKPI kpi) {     /**      * Keep the aggregated response time on a
```

```
kpi aggregated key named      * totalResponseTime, by adding the
current aggregated totalResponseTime, and the      * current
calculation cycle response time taken from the totalResponseTime
kpi key      * (calculated by the calculateKPI method).     */
    kpi.aggregatedKey.totalResponseTime = Utils.sum
(kpi.key.totalResponseTime, kpi.aggregatedKey.totalResponseTime)
    /**      * Keep the aggregated total duration on a kpi
aggregated key named totalDuration,      * by adding the current
aggregated total duration from the kpi aggregated key,      * and
the current calculation cycle total duration.     */
    kpi.aggregatedKey.totalDuration = Utils.sum
(kpi.key.totalDuration, kpi.aggregatedKey.totalDuration)     /**
 * Set the aggregated value of the KPI by dividing the two
aggregated values.     * This aggregated value will be displayed in
the SLA reports.     */     kpi.aggregatedValue = Utils.divide
(kpi.aggregatedKey.totalResponseTime,
kpi.aggregatedKey.totalDuration) }
```

計算 - 継続時間ベースの平均応答時間ルール

次では，継続時間ベースの平均応答時間ルールの計算例を示します。

10:00～11:00 の間，プロファイル・データベースには次の 5 つのサンプルがあります。

10:00:00 Sample1 {サンプル・フィールド :dResponseTime = 60}
10:04 Sample2 {サンプル・フィールド :dResponseTime = 30}
10:25 Sample3 {サンプル・フィールド :dResponseTime = 25}
10:38 Sample4 {サンプル・フィールド :dResponseTime = 30}
10:52 Sample5 {サンプル・フィールド :dResponseTime = 75}

計算サイクルはそれぞれ 5 分長です。ルール計算は次のとおりです。

	samples parameter		Keys set by calculateKPI		Aggregated keys and value set by calculateAggregatedKPI		
Cycle	Sample	Sample Duration	totalResponseTime	totalDuration	totalResponseTime	totalDuration	Aggregated Value
10:00-10:05	Sample1	240	16200	300			
	Sample2	60			16200	300	54.000
10:05-10:10	Sample2	300	9000	300	25200	600	42.000
10:10-10:15	Sample2	300	9000	300	34200	900	38.000
10:15-10:20	Sample2	300	9000	300	43200	1200	36.000
10:20-10:25	Sample2	300	9000	300	52200	1500	34.800
10:25-10:30	Sample3	300	7500	300	59700	1800	33.167
10:30-10:35	Sample3	300	7500	300	67200	2100	32.000
10:35-10:40	Sample3	180	8100	300			
	Sample4	120			75300	2400	31.375
10:40-10:45	Sample4	300	9000	300	84300	2700	31.222
10:45-10:50	Sample4	300	9000	300	93300	3000	31.100
10:50-10:55	Sample4	120	17100	300			
	Sample5	180			110400	3300	33.455
10:55-11:00	Sample5	300	22500	300	132900	3600	36.917
							Result: 36.917

例 - isSampleValid メソッド・フィルタを使用した継続時間ベースの平均応答時間ルール

このルールは110ページ「例 - 継続時間ベースの平均応答時間ルール」と同じですが，**isSampleValid** メソッド・フィルタが追加されています。

サンプル・フィールドのコードは，次のとおりです。

```
// This rule uses the dResponseTime sample field and u_iStatus
sample field. def sampleFields = ["dResponseTime", "u_iStatus"];
```

このルールには，次のメソッドの追加もあります。

```
/* * Override default implementation of the
SlmSamplesTimeBasedAggregatedCalculator
* interface method.* * Filter samples that hold u_iStatus field
with value of 6. */ public boolean isSampleValid(Sample sample) {
//Get the value of the sample's u_iStatus field. def
avialFieldValueObj = sample.u_iStatus; return (avialFieldValueObj
!= 6) }
```

計算 - isSampleValid メソッド・フィルタを使用した継続時間ベースの平均応答時間ルール

次では，**isSampleValid** メソッド・フィルタを使用した継続時間ベースの平均応答時間ルールの計算例を示します。

10:00～11:00 の間，プロファイル・データベースには次の 5 つのサンプルがあります。

10:00:00 Sample1 {サンプル・フィールド :dResponseTime = 60, u_iStatus = 0}
10:04 Sample2 {サンプル・フィールド :dResponseTime = 30, u_iStatus = 2}
10:25 Sample3 {サンプル・フィールド :dResponseTime = 25, u_iStatus = 6}
10:38 Sample4 {サンプル・フィールド :dResponseTime = 30, u_iStatus = 0}
10:52 Sample5 {サンプル・フィールド :dResponseTime = 75, u_iStatus = 2}

Sample3 は **isSampleValid** メソッド・フィルタを通過しなかったため，次の 4 つのサンプルが計算に含まれます。

10:00:00 Sample1 {サンプル・フィールド :dResponseTime = 60, u_iStatus = 0}
10:04 Sample2 {サンプル・フィールド :dResponseTime = 30, u_iStatus = 2}
10:38 Sample4 {サンプル・フィールド :dResponseTime = 30, u_iStatus = 0}
10:52 Sample5 {サンプル・フィールド :dResponseTime = 75, u_iStatus = 2}

フィルタリング後，Sample3 と Sample4 の間隔は Sample2 の継続時間に含まれるとみなされます。

ルールによる計算は次のようになります。

	samples parameter		Keys set by calculateKPI		Aggregated keys and value set by calculateAggregatedKPI		
Cycle	Sample	Sample Duration	totalResponseTime	totalDuration	totalResponseTime	totalDuration	Aggregated Value
10:00-10:05	Sample1 Sample2	240 60	16200	300	16200	300	54.000
10:05-10:10	Sample2	300	9000	300	25200	600	42.000
10:10-10:15	Sample2	300	9000	300	34200	900	38.000
10:15-10:20	Sample2	300	9000	300	43200	1200	36.000
10:20-10:25	Sample2	300	9000	300	52200	1500	34.800
10:25-10:30	Sample2	300	9000	300	61200	1800	34
10:30-10:35	Sample2	300	9000	300	70200	2100	33.428
10:35-10:40	Sample2 Sample4	180 120	9000	300	79200	2400	33
10:40-10:45	Sample4	300	9000	300	88200	2700	32.666
10:45-10:50	Sample4	300	9000	300	97200	3000	32.4
10:50-10:55	Sample4 Sample5	120 180	17100	300	114300	3300	34.636
10:55-11:00	Sample5	300	22500	300	136800	3600	38
							Result: 38

例 - isSampleAndDurationValid メソッド・フィルタを使用した継続時間ベースの平均応答時間ルール

このルールは110ページ「例 - 継続時間ベースの平均応答時間ルール」と同じですが，**isSampleAndDurationValid** メソッド・フィルタが追加されています。

サンプル・フィールドのコードは，次のとおりです。

```
// This rule uses the dResponseTime sample field and u_iStatus
sample field. def sampleFields = ["dResponseTime", "u_iStatus"];
```

このルールには，次のメソッドの追加もあります。

```
/* * Override default implementation of the
SlmSamplesTimeBasedAggregatedCalculator
* interface method.* * Filter samples that hold u_iStatus field
value which is not 0 or 2. */ public boolean
isSampleAndDurationValid(CI ci, SlmKPI kpi, Sample sample) { //Get
the value of the sample's u_iStatus field. def avialFieldValueObj =
sample.u_iStatus; return (avialFieldValueObj == 0 ||
avialFieldValueObj == 2) }
```

計算 - isSampleAndDurationValid メソッド・フィルタを使用した継続時間ベースの平均応答時間ルール

次では，**isSampleAndDurationValid** メソッド・フィルタを使用した継続時間ベースの平均応答時間ルールの計算例を示します。

10:00～11:00 の間，プロファイル・データベースには次の 5 つのサンプルがあります。

10:00:00 Sample1 {サンプル・フィールド :dResponseTime = 60, u_iStatus = 0}
10:04 Sample2 {サンプル・フィールド :dResponseTime = 30, u_iStatus = 2}

10:25 Sample3 {サンプル・フィールド :dResponseTime = 25, u_iStatus = 2}
10:38 Sample4 {サンプル・フィールド :dResponseTime = 30, u_iStatus = 1}
10:52 Sample5 {サンプル・フィールド :dResponseTime = 75, u_iStatus = 2}

Sample4 は **isSampleAndDurationValid** メソッド・フィルタを通過しなかったため，次の 4 つのサンプルが計算に含まれます。

10:00:00 Sample1 {サンプル・フィールド :dResponseTime = 60, u_iStatus = 0}
10:04 Sample2 {サンプル・フィールド :dResponseTime = 30, u_iStatus = 2}
10:25 Sample3 {サンプル・フィールド :dResponseTime = 25, u_iStatus = 2}
10:52 Sample5 {サンプル・フィールド :dResponseTime = 75, u_iStatus = 2}

フィルタリング後，Sample4 と Sample5 の間の間隔は Sample3 の一部とみなされないため，この時間の総継続時間は 46 分になります。

ルールによる計算は次のようになります。

	samples parameter		Keys set by calculateKPI		Aggregated keys and value set by calculateAggregatedKPI		
Cycle	Sample	Sample Duration	totalResponseTime	totalDuration	totalResponseTime	totalDuration	Aggregated Value
10:00-10:05	Sample1 Sample2	240 60	16200	300	16200	300	54.000
10:05-10:10	Sample2	300	9000	300	25200	600	42.000
10:10-10:15	Sample2	300	9000	300	34200	900	38.000
10:15-10:20	Sample2	300	9000	300	43200	1200	36.000
10:20-10:25	Sample2	300	9000	300	52200	1500	34.800
10:25-10:30	Sample3	300	7500	300	59700	1800	33.167
10:30-10:35	Sample3	300	7500	300	67200	2100	32.000
10:35-10:40	Sample3	180	4500	180	71700	2280	31.477
10:40-10:45	empty			0	71700	2280	31.477
10:45-10:50	empty			0	71700	2280	31.477
10:50-10:55	Sample5	180	13500	180	85200	2460	34.634
10:55-11:00	Sample5	300	22500	300	107700	2760	36.917
							Result: 39.021

例 - isSampleAndDurationValid と isSampleValid メソッド・フィルタを使用した継続時間ベースの平均応答時間ルール

このルールでは，**isSampleAndDurationValid** メソッド・フィルタと **isSampleValid** メソッド・フィルタが使用されます。これらについては，次の項で説明します。

- 112ページ「例 - isSampleValid メソッド・フィルタを使用した継続時間ベースの平均応答時間ルール」
- 113ページ「例 - isSampleAndDurationValid メソッド・フィルタを使用した継続時間ベースの平均応答時間ルール」

10:00～11:00 の間，プロファイル・データベースには次の 5 つのサンプルがあります。

10:00:00 Sample1 {サンプル・フィールド :dResponseTime = 60, u_iStatus = 0}
10:04 Sample2 {サンプル・フィールド :dResponseTime = 30, u_iStatus = 2}
10:25 Sample3 {サンプル・フィールド :dResponseTime = 25, u_iStatus = 6}
10:38 Sample4 {サンプル・フィールド :dResponseTime = 30, u_iStatus = 1}
10:52 Sample5 {サンプル・フィールド :dResponseTime = 75, u_iStatus = 2}

Sample3 は **isSampleValid** メソッド・フィルタを通過せず，Sample4 は **isSampleAndDurationValid** メソッド・フィルタを通過しませんでした。次の 3 つのサンプルが計算に含まれます。

10:00:00 Sample1 {サンプル・フィールド :dResponseTime = 60, u_iStatus = 0}
10:04 Sample2 {サンプル・フィールド :dResponseTime = 30, u_iStatus = 2}
10:52 Sample5 {サンプル・フィールド :dResponseTime = 75, u_iStatus = 2}

フィルタリング後，Sample3 とSample4 の間隔は Sample2 の継続時間に含まれるとみなされますが，Sample4 とSample5 の間隔は Sample4 の継続時間に含まれるとみなされません (**isSampleAndDurationValid** によってフィルタリングされます)。

ルールによる計算は次のようになります。

	samples parameter		Keys set by calculateKPI		Aggregated keys and value set by calculateAggregatedKPI		
Cycle	Sample	Sample Duration	totalResponseTime	totalDuration	totalResponseTime	totalDuration	Aggregated Value
10:00-10:05	Sample1	240	16200	300			
	Sample2	60			16200	300	54
10:05-10:10	Sample2	300	9000	300	25200	600	42
10:10-10:15	Sample2	300	9000	300	34200	900	38
10:15-10:20	Sample2	300	9000	300	43200	1200	36
10:20-10:25	Sample2	300	9000	300	52200	1500	34.8
10:25-10:30	Sample2	300	9000	300	61200	1800	34
10:30-10:35	Sample2	300	9000	300	70200	2100	33.429
10:35-10:40	Sample2	180	5400	180	75600	2280	33.157
10:40-10:45					75600	2280	33.157
10:45-10:50					75600	2280	33.157
10:50-10:55	Sample5	180	13500	180	89100	2460	36.219
10:55-11:00	Sample5	300	22500	300	111600	2760	40.434
							Result: 40.434

例 - API サンプルによるサービス停止ルール

このセクションでは，サンプルによるサービス停止ルールの例を示します。次の例について説明します。

例 - サンプルによるサービス停止ルールと標準設定ルール・パラメータを使用した計算

次のセクションでは，標準設定サービス停止ルール・パラメータに基づく，サンプルによるサービス停止ルールを示します。

最低失敗数 :2
最短継続時間 :0
最長継続時間 :未定義

サンプルの u_iStatus フィールド値が 0 または 2 でない場合，サンプルは 1 つの失敗を表します。u_iStatus フィールド値が 6 のサンプルも除外されます。

```
// Define the sample fields that will be used in the rule
calculation. def sampleFields = ["u_iStatus"];

/* * Implementation of the OutageBySamplesCalculator interface
method.* If the sample's u_iStatus field value is not 0 or 2, the
sample represents 1 failure.* それ以外のすべての場合，サンプルは失敗なしを表
す。* * For more information refer to the Rules API documentation.*/
public void calculateOutage(Outage outage, Sample sample) {     //
Take the sample field's u_iStatus value.     def statusFieldValue =
sample.u_iStatus;     if(statusFieldValue != 0 && statusFieldValue
!= 2){         outage.setNumberOfFailures 1;     } }

/* * Override default implementation of the
OutageBySamplesCalculator interface method.* * If the sample's u_
iStatus field value is not 0 or 2, the sample represents 1
failure.*/ public boolean isSampleValid(Sample sample) {     def
statusFieldValue = sample.u_iStatus;     return (statusFieldValue
!= 6) }
```

次の計算では，上記のサンプルによるサービス停止ルールを示します。

10:00～11:00 の間，プロファイル・データベースには次の 6 つのサンプルがあります。

10:10 Sample1 {サンプル・フィールド :u_iStatus = 1}
10:20 Sample2 {サンプル・フィールド :u_iStatus = 1}
10:25 Sample3 {サンプル・フィールド :u_iStatus = 6}
10:30 Sample4 {サンプル・フィールド :u_iStatus = 0}

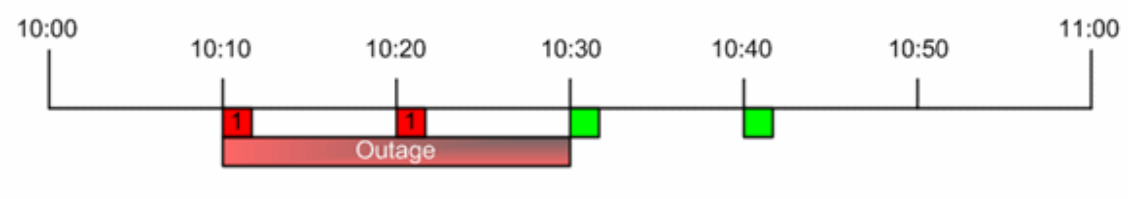
10:35 Sample5 {サンプル・フィールド :u_iStatus = 6}
10:40 Sample6 {サンプル・フィールド :u_iStatus = 2}

Sample3 および Sample5 は **isSampleValid** メソッドを通過しませんでした。次のサンプルが **calculateOutage** メソッドに渡されます。

10:10 Sample1 {サンプル・フィールド :u_iStatus = 1}
10:20 Sample2 {サンプル・フィールド :u_iStatus = 1}
10:30 Sample4 {サンプル・フィールド :u_iStatus = 0}
10:40 Sample6 {サンプル・フィールド :u_iStatus = 2}

Sample1 と Sample2 の場合，次の行が **calculateOutage** メソッド内で呼び出されます。**outage.setNumberOfFailures 1;**

計算結果は次のとおりです。



継続時間が 20 分 (10:10～ 10:30) のサービス停止が報告されます。

例 - 最短継続時間が 900 秒のサンプルによるサービス停止の計算

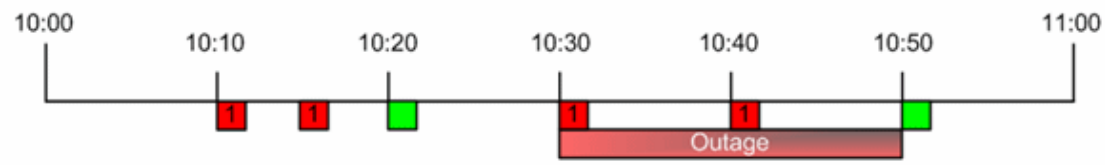
次の計算では，次のサービス停止ルール・パラメータに基づく，サンプルによるサービス停止ルールを示します。

最低失敗数 :2
最短継続時間 :900 (秒)
最長継続時間 :未定義

10:00～ 11:00 の間，プロファイル・データベースには次の 6 つのサンプルがあります。

- 10:10:00 - 1 つの失敗を表すサンプル
- 10:15:00 - 1 つの失敗を表すサンプル
- 10:20:00 - 失敗なしを表すサンプル
- 10:30:00 - 1 つの失敗を表すサンプル
- 10:40 - 1 つの失敗を表すサンプル
- 10:50 - 失敗なしを表すサンプル

計算結果は次のとおりです。



10:30～10:50 の間の継続時間が 20 分のサービス停止が報告されます。サービス停止継続時間が最短サービス停止継続時間パラメータに達しなかったため，10:10～10:20 の期間にはサービス停止はありません。

例 - 最長継続時間が 1 時間のサンプルによるサービス停止の計算

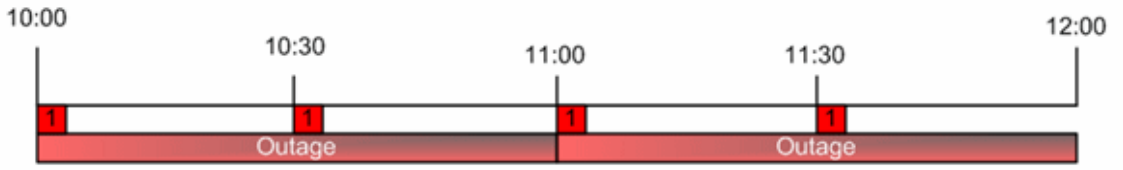
次の計算では，次のサービス停止ルール・パラメータに基づく，サンプルによるサービス停止ルールを示します。

最低失敗数 :2
最短継続時間 :0(標準設定)
最長継続時間 :1(時間)

10:00～12:00:00 の間，プロファイル・データベースには次の 4 つのサンプルがあります。

- 10:00:00 - 1 つの失敗を表すサンプル
- 10:30:00 - 1 つの失敗を表すサンプル
- 11:00 - 1 つの失敗を表すサンプル
- 11:30:00 - 1 つの失敗を表すサンプル

計算結果は次のとおりです。



継続時間が 1 時間の 2 つのサービス停止が報告されます。最初のサービス停止は 10:00～11:00，2 番目のサービス停止は 11:00～12:00 です。

例 - 2 つの失敗を表すサンプルを使用したサンプルによるサービス停止の計算

次の計算では，次のサービス停止ルール・パラメータに基づく，サンプルによるサービス停止ルールを示します。

最低失敗数 :2
最短継続時間 :0(標準設定)
最長継続時間 :未定義

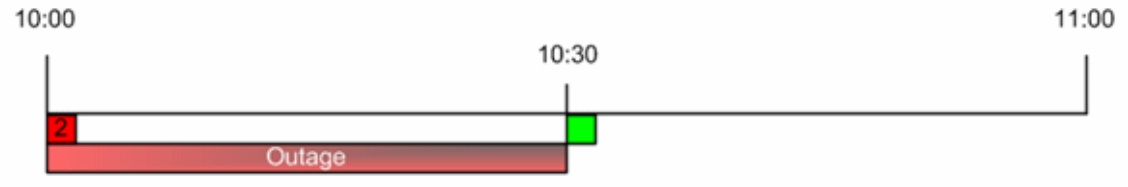
10:00～11:00 の間，プロファイル・データベースには次の 2 つのサンプルがあります。

- 10:00 - 2 つの失敗を表すサンプル

コードの次の行が，**calculateOutage** メソッドの **outage** パラメータで呼び出されます。`outage.setNumberOfFailures 2;`

- 10:30:00 - 失敗なしを表すサンプル

計算結果は次のとおりです。



継続時間が30分(10:00～10:30)のサービス停止が報告されます。

第5章

SLM Web サービス API

SLM Web サービス API とは，BSM 内外のアプリケーションから SLA を管理する統合ツールです。SLM Web サービスでは，HP Software-as-a-Service(HP SaaS) デプロイメントからも含む，SLA の作成と管理をサポートしています。

詳細については，次を参照してください。

- 121ページ「SLM Web サービスの使用」
- 122ページ「SLM Web サービスの操作 」

前提条件

API のユーザは，サービス・レベル管理とSOAP の概念に精通している必要があります。

権限

Web サービスに接続するためのログイン資格情報は，管理者が用意します。ユーザは，管理者権限またはSLA 所有者の資格情報が必要です。

権限マネージャで権限を設定する方法については，『BSM プラットフォーム管理ガイド』の「権限の概要」を参照してください。

サポートされている操作

次の Web サービス操作がサポートされています。

- SLA の作成(CI なし)
- SLA の一般的なプロパティの取得
- SLA の一般的なプロパティの更新
- SLA の削除
- SLA へのサービスの追加
- SLA からのサービスの削除
- SLA に含まれているサービスの取得

SLM Web サービスの使用

SLM Web サービス API では，サービス要求を送信できます。エンジンでステートメントを解析できなかった場合，またはステートメントを正常に実行できなかった場合は，エラーの説明が返されます。エラーがない場合は，要求の結果が返されます。

Web サービスの説明は，次の SOAP WSDL ファイルにあります。

`http://<サーバ>:8080/slm_ws/services/SlmServices?wsdl`

ポートの指定は，標準以外のインストールの場合のみ必要です。正しいポート番号については，システム管理者にお問い合わせください。

開発者は，Web サービスを呼び出すコードを，WSDL で開発環境から生成できます。WSDL ではインタフェース操作および操作パラメータを記述できます。

SLM Web サービスの操作

このセクションでは，SLM Web サービスの操作について説明します。

createSLA

新しい SLA を，指定した SLA プロパティで作成します。

操作の署名：

```
String createSLA(String customerId, SlaPropertiesDTO properties)
```

操作の引数：

引数	説明
customerId	通常は 1(HP SaaS 環境を除く)。
properties	新しい SLA を初期化するために使用する，次のプロパティ。 name,description,agreementDetails,type(OLA, SLA,UC), classification (internal, external); startDate,endDate,timeZoneId,customerId, providerId,trackingPeriods

戻り値 :RTSM の SLA の ID を返します。

例外 :ユーザが新しい SLA を作成できなかった場合 (たとえば，ユーザが SLA の作成に必要な権限を持っていない場合，または許可されている SLA の最大数に達している場合) は，`SlmWebServiceException` がスローされます。

updateSLA

新しいプロパティを使用して SLA を更新します。

操作の署名：

```
updateSLA(String customerId, String SLAId, SlaPropertiesDTO
properties)
```

操作の引数：

引数	説明
customerId	通常は 1(HP SaaS 環境を除く)。
SLAId	更新する SLA の RTSM ID。
properties	更新するプロパティ。

例外:

- 指定した ID の SLA が存在しない場合は, `SlaDoesNotExistsException` がスローされます。
- 操作の正常な実行を妨げるシステム上の問題が発生した場合は, `SlmWebServiceException` がスローされます。

deleteSLA

指定した SLA を削除します。

操作の署名:

```
deleteSLA(String customerId, String SLAId)
```

操作の引数:

引数	説明
customerId	通常は 1(HP SaaS 環境を除く)。
SLAId	削除する SLA の RTSM ID。

例外:エラーが発生した場合, 種類にかかわらず `SlmWebServiceException` がスローされます。

getSLAProperties

指定した SLA のプロパティを取得します。

操作の署名:

```
SlaPropertiesDTO getSLAProperties(String customerId, String SLAId)
```

操作の引数:

引数	説明
customerId	通常は 1(HP SaaS 環境を除く)。
SLAId	取得するプロパティを持つ SLA の RTSM ID。

例外:エラーが発生した場合, 種類にかかわらず `SlmWebServiceException` がスローされます。

addServicesForSLAWithOfferings

指定したサービスおよび影響サブツリーを SLA に追加します。サービスごとに, 一致する提供サービスが使用されます。

操作の署名:
`String[] addServicesForSLAWithOfferings(String customerId, String slaId, ServiceWithOffering[] servicesWithOffering)`

操作の引数:

引数	説明
customerId	通常は 1(HP SaaS 環境を除く)。
SLAId	サービスを追加する SLA の RTSM ID。
servicesWithOffering	一致する提供サービス名の SLA に追加するサービス。 `ServiceWithOffering` にはそれぞれserviceId, serviceName, offeringName が含まれています。 **注:** • offeringName を指定する必要があります。 • 少なくとも 1 つの serviceId または serviceName を指定する必要があります。`serviceId` が存在しない場合, サーバは `serviceName` を使用して取得を試行します。

戻り値:検証エラー・メッセージの配列が返されます(エラーがない場合は空の配列)。

例外:操作の正常な実行を妨げるシステム上の問題が発生した場合は, `SlmWebServiceException` がスローされます。

deleteServiceFromSLA

指定したサービスとそのパスを指定した SLA から削除します。

操作の署名:
`deleteServiceFromSLA(String customerId, String SLAId, Service[] services)`

操作の引数:

引数	説明
customerId	通常は 1(HP SaaS 環境を除く)。
SLAId	サービスを削除する SLA の RTSM ID。
services	SLA から削除するサービス。 サービスにはそれぞれ`serviceId`, `serviceName` が含まれています。 `serviceId` が存在しない場合, サーバは serviceName を使用して取得を試行します。

例外:エラーが発生した場合, 種類にかかわらず `SlmWebServiceException` がスローされます。

getSLAServicesFullPath

指定した SLA のサービスを取得します。

操作の署名：
```
ServiceFullPath[] getSLAServicesFullPath(String customerId, String
SLAId)
```

例： サービス4，5があるSLAの例：

```
{id1, id2, id3, id4}
{id1, id2, id5}
```

操作の引数：

引数	説明
customerId	通常は1(HP SaaS環境を除く)。
SLAId	SLAのRTSM ID。

戻り値：SLA内のサービスの完全パスが返されます。

例外：エラーが発生した場合，種類にかかわらず `SlmWebServiceException` がスローされます。

getServiceSLAs

指定したサービスを含むSLAが返されます。

操作の署名：
```
String[] getServiceSLAs(String customerId, String serviceId)
```

操作の引数：

引数	説明
customerId	通常は1(HP SaaS環境を除く)。
serviceId	サービスのRTSM ID。

戻り値：指定したサービスを含むSLA IDの配列が返されます。

例外：エラーが発生した場合，種類にかかわらず `SlmWebServiceException` がスローされます。

第3部分

エンド・ユーザ管理

第6章

EUM Admin の Open API

EUM 管理の Open API を使用すると，Business Service Management の EUM 管理ユーザ・インタフェースを使用しなくても，EUM 設定に関する操作を実行できます。この API は，Business Process Monitor および Real User Monitor の設定の取得，更新，および作成をサポートしています。

EUM 管理の Open API は RESTful Web サービスです。これには，JAX-RS v1.0 標準に準拠する統合サーバ側モジュールが含まれています。さらに，Apache Wink フレームワークに基づく Java クライアントも提供されます。

EUM Admin の Open API クラスに関する説明は，Javadoc 形式で『BSM EUM Administration API Reference』(英語版) に記載されています。これらのファイルは次のフォルダにあります。

\\<ゲートウェイ・サーバのルート・ディレクトリ>\AppServer\webapps\site.war\amdocs\eng\doc_lib\API_docs\EUM_API\EUM_Administration_API_CSH.htm

第4部分

オペレーション管理

第7章

オペレーション管理の拡張

この『Operations Manager i 拡張性ガイド』では，Operations Manager i(OMi) ライセンスによって提供される HP Business Service Management (BSM) オペレーション管理の機能のカスタマイズおよび拡張について説明します。

本書は，次のセクションで構成されます。

- セクション I: 133ページ「コンテンツの展開」
- セクション II: 167ページ「実行時サービス・モデルへのデータの追加」
- セクション III: 226ページ「イベント処理インタフェース」
- セクション IV: 240ページ「オペレーション管理 UI とほかのアプリケーションとの統合」
- セクション V: 251ページ「オペレータ機能およびイベント変更検出の自動化」
- セクション VI: 303ページ「外部イベント・プロセスの統合」

セクション I: 133ページ「コンテンツの展開」では，簡単な例として架空の ACME 環境を使用し，コンテンツ開発者が新規アプリケーションの管理機能を追加するために必要な手順を示します。この例では，新規アプリケーションの管理情報をオペレーション管理 で使用できるようにするために実行する必要のあるさまざまな統合手順について説明します。

セクション II: 167ページ「実行時サービス・モデルへのデータの追加」では，開発者が独自のトポロジ同期マッピング・ルールを作成し，標準設定のマッピング・ルールを拡充して Run-Time Service Model (RTSM) に HP Operations Manager(HPOM) のノードやサービスから構成アイテム(CI) および CI 関係を追加する方法について説明します。

セクション I の133ページ「コンテンツの展開」で紹介した ACME 環境の例をさらに発展させ，特定のサービス・モデルに固有のトポロジ同期ルールを作成する方法を説明します。

セクション III: 226ページ「イベント処理インタフェース」では，イベント処理においてイベントを変更および強化するイベント処理スクリプトとカスタム・アクションの役割について説明します。

セクション IV: 240ページ「オペレーション管理 UI とほかのアプリケーションとの統合」では，ドリルダウン URL 起動を使用してオペレーション管理ユーザ・インタフェースの要素に外部アプリケーションを統合する方法について説明します。

セクション V: 251ページ「オペレータ機能およびイベント変更検出の自動化」では，インテグレータを対象として，プログラムを使用してオペレータ機能を自動化し，イベント Web サービスを使用してイベントの変更を検出する方法について説明します。イベントの作業時にオペレータがコンソールで行うほとんどの操作をプログラミングでき，効率性の向上を実現できます。

セクション VI: 303ページ「外部イベント・プロセスの統合」では，イベント処理に外部プロセスを統合することを目的とする外部アプリケーションとの統合を実現するイベント同期 Web サービス・インタフェースについて説明します。このインタフェースでは，転送されたイベントおよびそれ以降のイベント変更に関する通知をプログラムで受信できます。

イベント同期 Web サービス・インタフェースの主目的は，イベントおよびイベントへの変更を ITIL インシデント・マネージャ(HP Service Manager や BMC Remedy Service Desk など) のような外部マネージャと同期させることです。

前提条件

次に，OMi を拡張およびカスタマイズするための前提条件を示します。

- 関連する HP ソフトウェア製品およびコンポーネントについての知識があり，それらのマニュアルを読んでいることが必要です。次に，HPOM 間の統合を拡張するために必要な手順の例をいくつか示します。
  - **実行時サービス・モデル**(RTSM) 。管理および運用に関する詳細な知識を備えていることが前提となります。『HP Business Service Management Modeling Guide』は，CI タイプ，CI 属性，ビューなどの RTSM の概念を把握するための必須の資料です。Excel ブックなどの外部ソースからのデータ・インポートの詳細については，『HP Universal CMDB Discovery and Integration Content Guide』の「Import From Excel Workbook Discovery」および「Importing Data from External Sources」を参照してください。これらのガイドは，HP ソフトウェア製品マニュアルの Web サイト(http://support.openview.hp.com/selfsolve/manuals) の Universal CMDB (Application Mapping) 製品のページにあります。
  - Windows および UNIX 用の **HP Operations Manager**(HPOM) 。特に，HPOM のサービス・ツリーを参照し，RTSM と同期させる必要のあるサービスを特定してください。

    注 :RTSM はグラフを使用し，ツリー構造を必要としないため，ツリーの親 (アプリケーションやシステム・インフラストラクチャなど) の作成のみに使用されるサービスは，ほとんどの場合，同期する必要はありません。

  - **Operations Manager i**(OMi) 。たとえば，次の手順は OMi で実行する必要があります。
    i. HPOM サービス・ツリーを参照しながら `contextmapping.xml` ファイル内のコンテキストと同期させるサービスをマークする。
    ii. 各 HPOM サービスの CI タイプを選択する。HPOM サービスから CI タイプへのマッピングを `typemapping.xml` ファイルに入力します。
    iii. `attributemapping.xml` ファイルで，必要なすべての CI 属性 (キー属性) が HPOM サービス属性から取得されていることを確認する。
    iv. RTSM モデルに従ってすべての関係を `relationmapping.xml` ファイルに作成する。

    注 : CI タイプには，単純な CI 属性と関係からなるキーを持つものがあります(実行中のソフトウェアなど) 。このような CI タイプについては，`attributemapping.xml` ですべてのキー属性を設定し，`relationmapping.xml` に適切な関係を作成することによって，CI を正しくインスタンス化する必要があります。

    同期パッケージの使用例については，`default`，`operations-agent`，`nodegroups` などの標準トポロジ同期パッケージを参照してください。これらのトポロジ同期パッケージは次の場所にあります。

    <HPBSM ルート・ディレクトリ>`/conf/opr/topology-sync/sync-packages`

    この情報は『BSM 拡張性ガイド』に含まれています。

  - **HP Business Service Management** (BSM) 。
- Groovy スクリプトを作成するためには，Groovy スクリプティングと構文に関する知識が必要です。

スクリプティングには Groovy がサポートされ，Groovy スクリプトは，トポロジ同期プロセス，イベント処理インタフェース(EPI) およびカスタム・アクション・スクリプト，オペレータ機能の自動化，および外部イベント・プロセスの統合に使用されます。

- マッピング・エンジンの CI データ構造内を移動するためには，XPath クエリ言語の知識が必要です。

第8章

コンテンツの展開

本項では，次の操作に必要な手順について説明します。

- 標準で用意されている既存の監視設定データをカスタマの要求に合わせてカスタマイズする。
- IT 環境の新しいアプリケーションおよびエレメントに監視機能を追加する。

これらの手順は，ACME 環境の簡単な例を使用して説明します。.

本項の内容

統合コンテンツ

新しいアプリケーション領域を監視ソリューションに統合する場合は，その統合にとって次の点が重要となります。

- 134ページ「トポロジ」
- 134ページ「イベント・タイプ・インジケータと状況インジケータ」
- 135ページ「相関処理ルール」
- 135ページ「追加のイベント処理」
- 135ページ「ツール」
- 135ページ「ビュー・マッピング」
- 135ページ「グラフ」

トポロジ

トポロジ・データは実行時サービス・モデル(RTSM)に含まれています。RTSMには，構成アイテム・タイプ(CIタイプ)の定義と，CIタイプ間において可能な関連付けが含まれています。構成アイテム(CI)はCIタイプのインスタンスです。

新規アプリケーションをBSM オペレーション管理 監視ソリューションに統合し，新規アプリケーションのトポロジ・ビューを作成するためには，次の手順の実行が必要となる場合があります。

- 新規アプリケーション用に新しいCIタイプを作成します。
- 新しいCIタイプのキー属性値を特定します。
- 新規アプリケーションに対して関係(メンバシップ，依存関係，構成関係など)を確立します。
- RTSMにCIおよびCI関係を作成します。

新規アプリケーションのトポロジ・データを統合するために必要な労力は，どのような既存のデータがあるかによって異なります。たとえば，既存のRTSMオブジェクトを再利用できるアプリケーションを統合する方が，すべてのRTSM CIタイプとその関係の定義を含め，一から始める必要があるアプリケーションを統合する場合より，少ない労力で済みます。

新規アプリケーションの統合におけるトポロジ・データの役割の詳細については，136ページ「トポロジ」を参照してください。

イベント・タイプ・インジケータと状況インジケータ

オペレーション管理を実行するBSM監視ソリューションが提供する高度な状況ベースの監視機能を新しいアプリケーション領域で利用するには，次の手順を実行します。

- RTSMにCIを追加し，オペレーション管理 イベントをRTSM内の適切なCIにマップします。
- 受信したイベントをCIのサービス状況に関するデータに変換します。つまり，さまざまなCIタイプの受信イベントを分析し，意味のあるイベント・タイプ・インジケータ(ETI)および状況インジケータ(HI)を作成します。
- HIを状況ベースの主要管理指標(KPI)に割り当てます。

詳細については，147ページ「イベント・タイプ・インジケータと状況インジケータ」を参照してください。

相関処理ルール

あらゆるソースからのイベントを中央のコンソールに統合し，さらにトポロジベースのイベント相関処理(TBEC)を使用してイベントを相関させることによって，イベント管理プロセスを簡略化できます。

TBEC ルールは，既知の根本原因イベントとそれに関連する症状イベントを関連付けます。症状イベントは，根本原因イベントの結果として発生するイベントです。TBEC によって，重複や過負荷が回避されるため，ブラウザに表示されるイベントの数が大幅に削減されます。これにより，大規模なネットワークで発生する大量の類似(関連)した症状イベントを管理できます。

TBEC ルールに指定された構成アイテム・タイプに影響するイベントを表すには，HI 値とETI 値を使用します。これらの値を使用して相関処理ルールを作成します。

相関処理ルールの詳細については，158ページ「相関処理ルールとマッピング」を参照してください。

追加のイベント処理

追加のイベント処理を実行することにより，Groovy スクリプトを使用してイベントを変更および強化できます。

イベント処理インタフェース(EPI)を使用して，イベント処理において多くのユーザ定義の Groovy スクリプトを実行できます。

イベントに適用するカスタム・アクションを設定することもできます。

追加のイベント処理の詳細については，159ページ「追加のイベント処理」を参照してください。

ツール

個々のイベントの管理および監視や新しいアプリケーション領域に関するイベント関連の問題の解決に役立つツールを設定できます。

新規アプリケーションに設定されたツールの例については，160ページ「ツール」を参照してください。

ビュー・マッピング

RTSM Modeling Studio を使用して構成アイテム・タイプを RTSM ビューにマップすることにより，ビューを[ヘルストップ ビュー]ペインで選択および使用できます。

CI タイプの RTSM ビューへのマッピングの詳細については，161ページ「ビュー・マッピング」を参照してください。

グラフ

グラフやチャートによって，新しいアプリケーション領域のイベントが影響する構成アイテムに影響を与えるパフォーマンス関連の問題や傾向の視覚化と分析に役立つ追加のデータが得られます。

グラフの詳細については，162ページ「グラフ」を参照してください。

トポロジ

本項では，「ACME」と呼ばれるサンプル・アプリケーション環境を使用して，新規アプリケーションを統合して新しい環境のトポロジ・ビューを作成する方法について説明します。

本項の内容

新規アプリケーションの統合

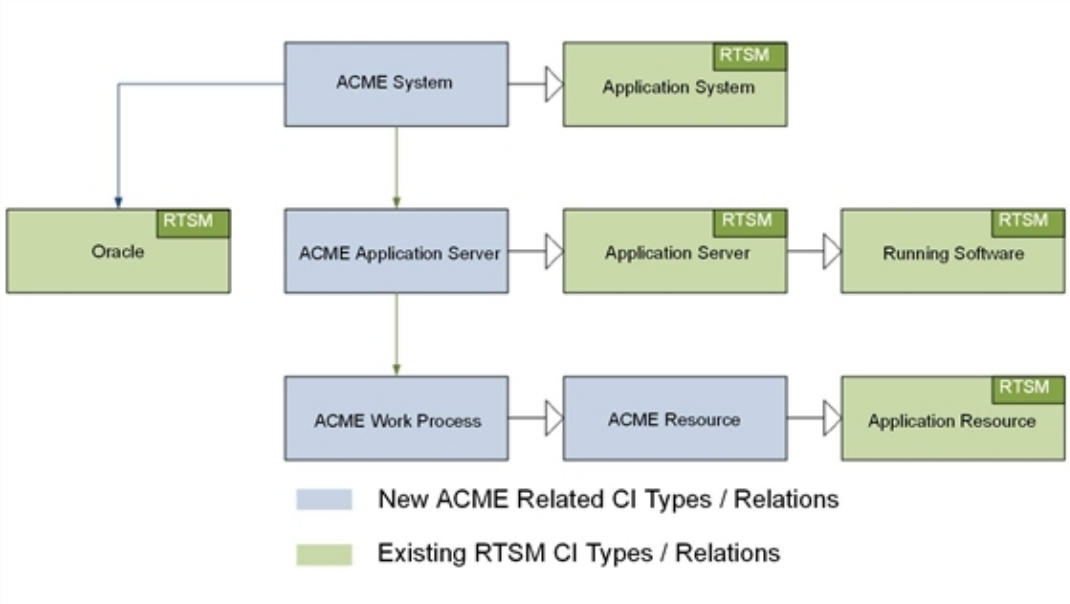
オペレーション管理を実行するBSM監視ソリューションに新規アプリケーションを統合し，新規アプリケーションのトポロジ・ビューを作成するには，次の手順を実行します。

- 新規アプリケーションの新しいCIタイプを作成します(既存のCIタイプを再利用できない場合)。
- 新しいCIタイプのキー属性値を設定します。
- 新規アプリケーションの関係を確立します。
- RTSMにCIおよびCI関係を作成します。

新規アプリケーションの新しいCIタイプの作成

新規アプリケーションを統合するには，まずアプリケーションの新しいCIタイプを作成します。

次の図に“ACME”環境のトポロジ・モデルを示します。



このACME環境の例では，**ACME システム**内にさまざまな **ACME アプリケーション・サーバ**があります。これらのアプリケーション・サーバでは，**ACME ワーク・プロセス**を使用してユーザの要求を実行します。

ACME環境では，**Oracle データベース**に情報を保存します。上の図では，次に挙げる4つの新しいCIタイプが青で示されています。

- ACME System
- ACME Application Server
- ACME Work Process
- ACME Resource

これらの新しいCIタイプは，RTSM内の既存のCIタイプ(緑色)の子要素です。

新しいCIタイプの作成方法とRTSMの概念の使用方法の詳細については，BSMの『Modeling Guide』を参照してください。

新しいCIタイプを作成するには，次の手順を実行します。

1. CIタイプ・マネージャを開きます。

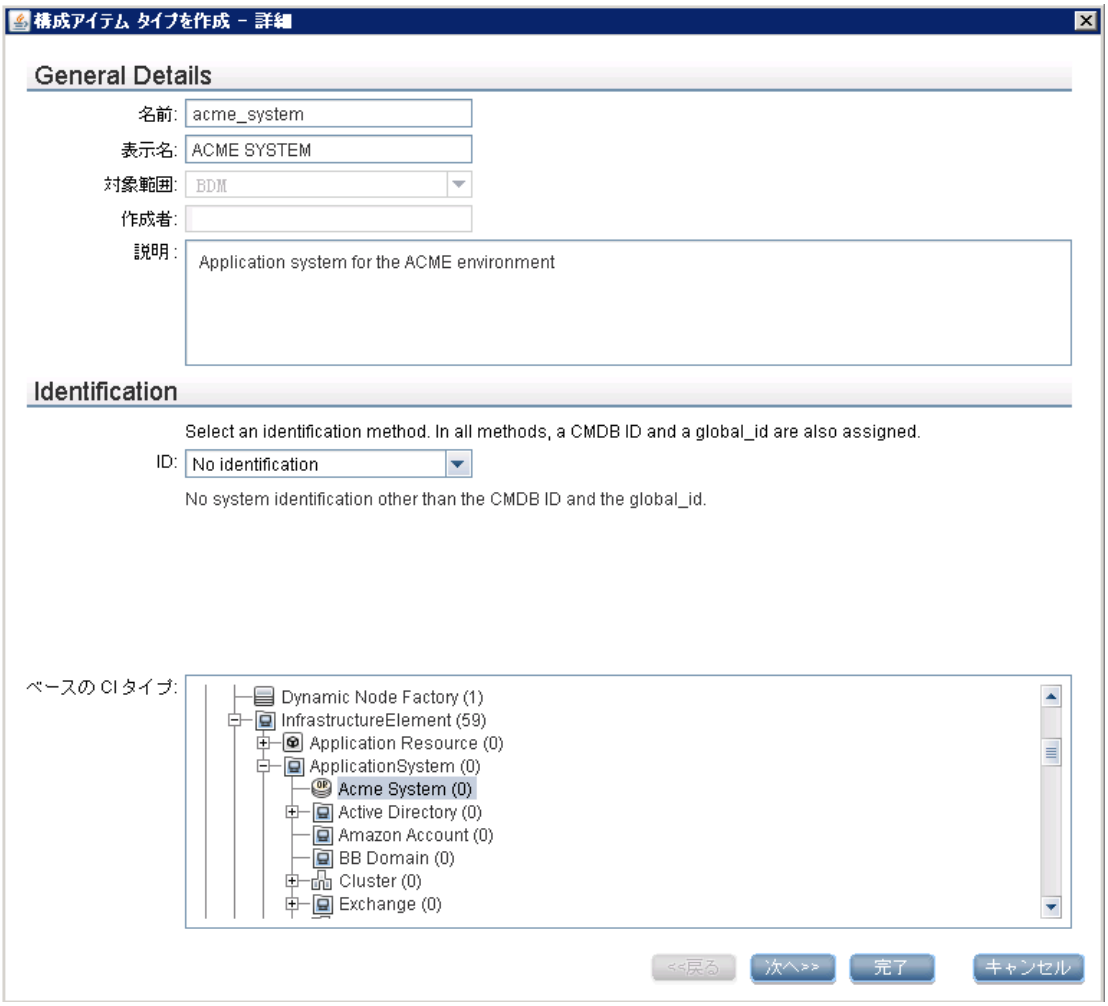
   [RTSM 管理] > [モデリング] > [CI タイプ・マネージャ]

2. [CIタイプ]ペインで，ドロップダウン・メニューから[**CI タイプ**]を選択してCIタイプ・ツリーをアクティブにします。

3. CIタイプ・ツリーで，新規アプリケーションを追加するフォルダへ移動します。次に，例を示します。

   [構成アイテム] > [インフラストラクチャ要素] > [Application System]

4. 右クリックして，[新規]( ) ボタンをクリックします。[Create Configuration Item Type]ダイアログ・ボックスが開きます。



5. [名 前]フィールドに，作 成 する CI タイプの名 前 を「**acme_system**」と入 力します。

   [表 示 名]フィールドに，CI タイプの表 示 名 を「**ACME System**」と入 力します。

   オプション。[説 明]フィールドに，作 成 している CI タイプの説 明 を入 力します。

   [**次 へ**]をクリックして[Create Configuration Item Type]ダイアログの次 のページへ進 みます。このページでは，次 の138ページ「新 しい CI タイプのキー属 性 値 の設 定 」の説 明 に従 って，新 たに作 成 した CI のキー属 性 を設 定 します。

新しい CI タイプのキー属 性 値 の設 定

新 しい CI タイプを一 意 のキー属 性 で識 別 する必 要 があります。一 意 のキー属 性 を設 定 することにより，異 なるディスカバリ・ソースなどによって重 複 する CI が作 成 されるおそれがなくなります。

次 の表 に ACME 環 境 のキー属 性 の候 補 を示 します。

ACME 環境の CI タイプ属性のリスト

CI タイプ表示名	属性	説明	値
ACME System	name	ACME システムの名前	システム ID
ACME Application Server	name	ACME システム・ランドスケープ内の ACME サーバを識別する一意の名前	ACME サーバ名
	root_container	コンテナ・ホスト	ホストの CI 参照 (CI ID)
ACME Work Process	name	特定のタイプのワーク・プロセスを表す論理的なシングル インスタンス表現	ワーク・プロセス・カテゴリ，バッチ，ダイアログ，またはスプール

1. ［Create Configuration Item Type-Attributes］ダイアログで，CI タイプのキー属性 (ACME System など) を設定します。

   新しい CI タイプを既存の属性で識別するには，キー属性として設定する属性と同じ行の［**キー**］カラムをクリックします。小さなキー・アイコンが表示されます(その属性を設定しない場合は，もう一度クリックします) 。

表示名	名前	タイプ	詳細	標準設定値	可視	編集可能	キー	比
Description	description	string	Description		✔	✔		
Digest	digest	string				✔		
Display Label	display_label	string	Used as ...		✔			
Documents	document_list	string	Documents			✔		
Enable Aging	root_enableageing	boolean	Is aging e...	false	✔	✔		
External ID	data_externalid	string	external s...			✔		
FAMILY_ICON	FAMILY_ICON	string				✔		
Global Id	global_id	string	A globally ...		✔			
global_id_scope	global_id_scope	string		Default				
Is Candidate For ...	root_iscandidatef...	boolean	Is candid...	false	✔			
Language	language	string	System s...			✔		
Last Access Time	root_lastaccessti...	date	When wa...		✔			
LastModifiedTime	last_modified_time	date	When wa...		✔			
layer	layer	layer_enu...		software	✔	✔		
MENU	MENU	xml		<MENU>...		✔		
Monitored By	monitored_by	string_list			✔	✔		
Name	name	string			✔	✔		
Note	data_note	string			✔	✔		
Operation Corr St...	data_operationcor...	operation...	Operation...	Normal				
Operation Is New	data_operationisn...	boolean	Operation...	false				
Operation State	data_operationsta...	operation...	Operation...	Normal				
Origin	data_origin	string			✔			
Owner Tenant	TenantOwner	string	The Tena...		✔	✔		
Reference to the c...	credentials_id	string	Referenc...			✔		
root_iconproperties	root_iconproperties	string			✔	✔		
~~State~~	~~state~~	~~string~~	~~State loca...~~			✔		
Store KPI History ...	is_save_persiste...	boolean	Store KPI ...	false		✔		
System	root_system	string						

2. 新しい CI タイプのキー属 性を設 定するだけでなく，その CI タイプ専 用 の独 自 の属 性を作 成 することもできます。

   新しい属 性を作 成 するには，次 の手 順を実 行します。

   a. [Create Configuration Item Type-Attributes]ダイアログの[追 加]( ) ボタンをクリックします。[Add Attribute]ダイアログが開きます。

b. [属性名]フィールドに新しい属性の名前を「**acme_instance_number**」と入力します。

[表示名]フィールドに新しい属性の表示名を「**ACME Instance Number**」と入力します。

[スコープ]フィールドで，スコープとして「**BDM**」を選択します。

オプション。[説明]フィールドに新しい属性の説明を入力します。

c. 属性の型を設定し，必要に応じて[Value Size]フィールドと[標準設定値]フィールドに値を入力します。

d. [**OK**]をクリックします。

3. 手順 1 で説明したように，新たに作成した属性を CI タイプ ACME System のキー属性として設定します。

4. [**終了**]をクリックします。

新規アプリケーションの関係の確立

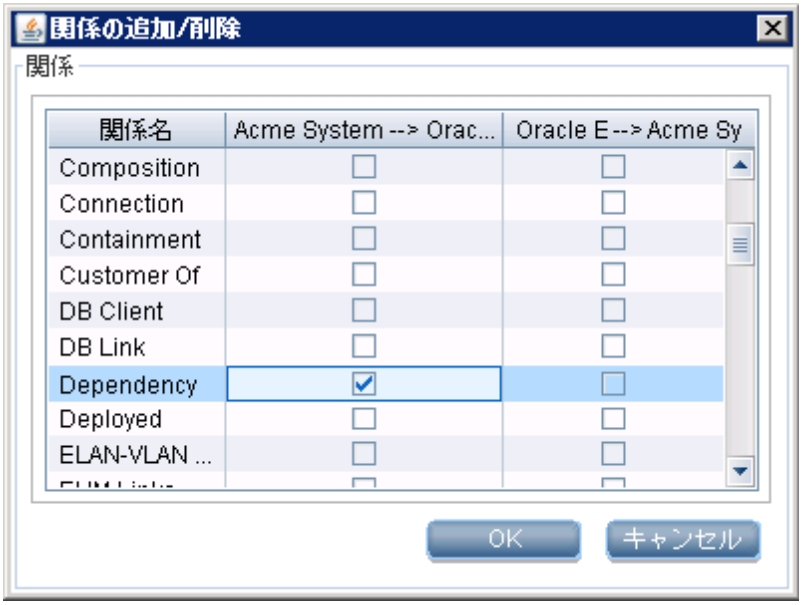
新規アプリケーションを統合するには，次にアプリケーションの関係を確立します。

標準設定の RTSM モデルには，ACME システムと ACME アプリケーション・サーバの間に**メンバシップ**関係があります。この例の ACME システムは Oracle データベースに依存しています。ただし，この**依存**関係は標準設定の RTSM モデルには存在しません。CI タイプ・マネージャを使用して，この関係を作成する必要があります。

[RTSM 管理] > [モデリング] [CI タイプ・マネージャ]

ACME システムと Oracle データベースの間に依存関係を作成するには，次の手順を実行します。

1. CI タイプ・マネージャで，[**ACME システム**]と[**Oracle**]を選択します。
2. 右クリックし，[**Add/Remove Relationship**]を選択します。
3. [Add/Remove Relationship]ダイアログで，[**ACME システム > Oracle**]カラムの[依存関係]のボックスにチェックを入れ，ACME システムが Oracle データベースに依存するという関係を確立します。



4. [**OK**]をクリックします。

RTSM での CI および CI 関係の作成

新規アプリケーションを統合するには，次に RTSM で CI および CI 関係を作成します。

CI および CI 関係を作成するには，次の 3 つの方法があります。

- RTSM のディスカバリ機能と HP データ・フロー管理(DFM)を使用して CI と CI 関係を作成する。
- トポロジ同期を使用して CI と CI 関係を作成する。トポロジ同期では，HPOM サービス・モデルを再利用して，対応する CI および CI 関係を作成します。もちろん，そのためには HPOM に適切なサービス・モデルが作成されている必要があります。

  トポロジ同期の詳細については，167ページ「実行時サービス・モデルへのデータの追加」を参照してください。
- CI を RTSM 内に手動で作成する。この方法が実際的であるのは，少数の CI しか作成する必要がなく，それらの CI が本質的に安定していて，変更される可能性が低い場合に限られます。

RTSM/HP データ・フロー管理 ディスカバリによる CI の作成

HP データ・フロー管理 は，開放型システム間相互接続(OSI)モデルのレイヤ 2 ～ 7 で論理アプリケーション・アセットを自動的に検出し，マップします。このディスカバリ・テクノロジは，ディスカバリ・パターンを基礎としています。

データ・フロー管理 のライセンス構造は次のとおりです。

- UCMDB Foundation ライセンス。Foundation ライセンスには，BTO 製品のバックボーン・コンポーネントとして UCMDB が含まれています。このバージョンでは，UCMDB の複数のインスタンス間でデータ・フローを実現でき，BTO 製品との統合によってソリューションを展開できます。
- UCMDB Integration ライセンス。Integration ライセンスは，UCMDB Foundation ライセンスにサード・パーティ統合を追加したものです。
- UCMDB DDM Advanced ライセンス。DDM Advanced ライセンスには，IT インフラストラクチャ要素を検出し，その情報を CI および関係として RTSM に送るためのディスカバリ機能がすべて含まれています。DDM Advanced ライセンスでは，ユーザが RTSM モデルを拡張し，独自のディスカバリ・パターンを作成することもできます。

データ・フロー管理を使用すると，以下の外部データ・ソースへのクエリを実行することもできます。

- カンマ区切り(CSV) ファイル
- プロパティ・ファイル
- データベース

詳細については，HP Business Service Management 文書ライブラリの『Data Flow Management Guide』および『RTSM Developer Reference Guide』を参照してください。

トポロジ同期による CI の作成

多くの HPOM カスタマは，HPOM サービス・ビューまたはサービス・ナビゲータを使用して，IT リソースと IT サービスとの関係をオペレータに対して視覚化します。

この方法では，HP Operations スマート・プラグインのサービス・ディスカバリ機能または独自のディスカバリ・メカニズムを使用してサービス・ツリーを作成します。

この場合は，トポロジ同期機能を使用し，HPOM サービス・モデルとトポロジ同期マッピング・ルールに基づいて CI と CI 関係を作成できます。標準設定のマッピング・ルールが用意されており，それによって，オペレーション管理と連携できる以下の HP Operations スマート・プラグインで作成されたサービス・モデルをマップできます。

- HP Operations Smart Plug-in for Databases(Oracle および MS SQL Server のみ)
- HP Operations Smart Plug-in for IBM WebSphere Application Server
- HP Operations Smart Plug-in for BEA WebLogic Application Server
- HP Operations Smart Plug-in for Microsoft Active Directory
- HP Operations Smart Plug-in for Microsoft Exchange Server
- HP Operations Smart Plug-in for Virtualization Infrastructure
- HP Operations Smart Plug-in for Systems Infrastructure
- HP Operations Smart Plug-in for Cluster Infrastructure

多大な労力を注ぎ込み，独自のカスタム・サービス・モデルとそのマニュアルや自動サービス・モデル作成プロセスを作成した場合は，そのモデルを再利用することにより，対応する RTSM 構成アイテムを自動的に作成できます。その場合，必要となるのは，対応するトポロジ同期マッピング・ルールを作成することだけです。

トポロジ同期の詳細については，167ページ「実行時サービス・モデルへのデータの追加」を参照してください。

ディスカバリ方法を選択する際の考慮事項

RTSM にデータを追加するためのディスカバリ方法を選択するときには，次の考慮事項が役立ちます。

- 『Data Flow Management Guide』の使用が適している場合

  次の場合に『Data Flow Management Guide』を使用します。

  - RTSM にデータを追加するために，すでに『Data Flow Management Guide』を使用しているか，使用する計画がある場合。
  - HPOM 内にまだサービス・モデルが**ない**場合。『Data Flow Management Guide』を使用することをお勧めします。これがオペレーション管理 の RTSM にデータを追加するのに望ましいディスカバリ方法です。

- トポロジ同期の使用が適している場合

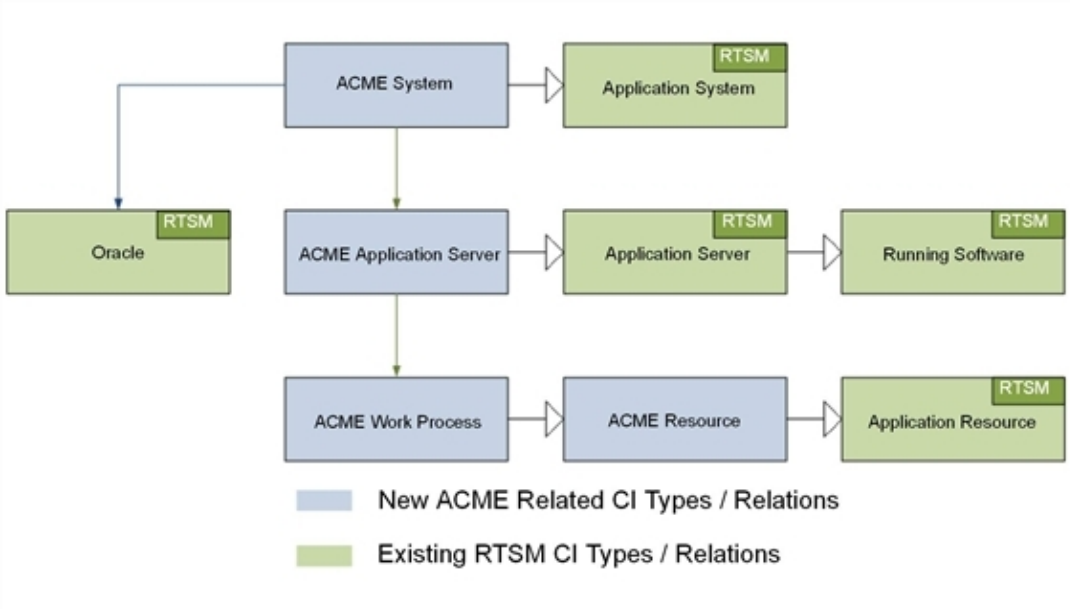
  次の場合は，同期パッケージに用意されている標準設定のトポロジ同期ルールを使用します。

  - RTSM へのデータの追加に『Data Flow Management Guide』を使用しておらず(使用する計画もなく)，かつ
  - ACME トポロジに対応するサービスが含まれる HPOM に既存のサービス・モデルがある場合。

- CI を手動で作成することが適している場合

  少数の CI しか作成する必要がなく，それらの CI が本質的に安定していて，変更される可能性が低い場合は，CI を手動で作成できます。

強化ルール

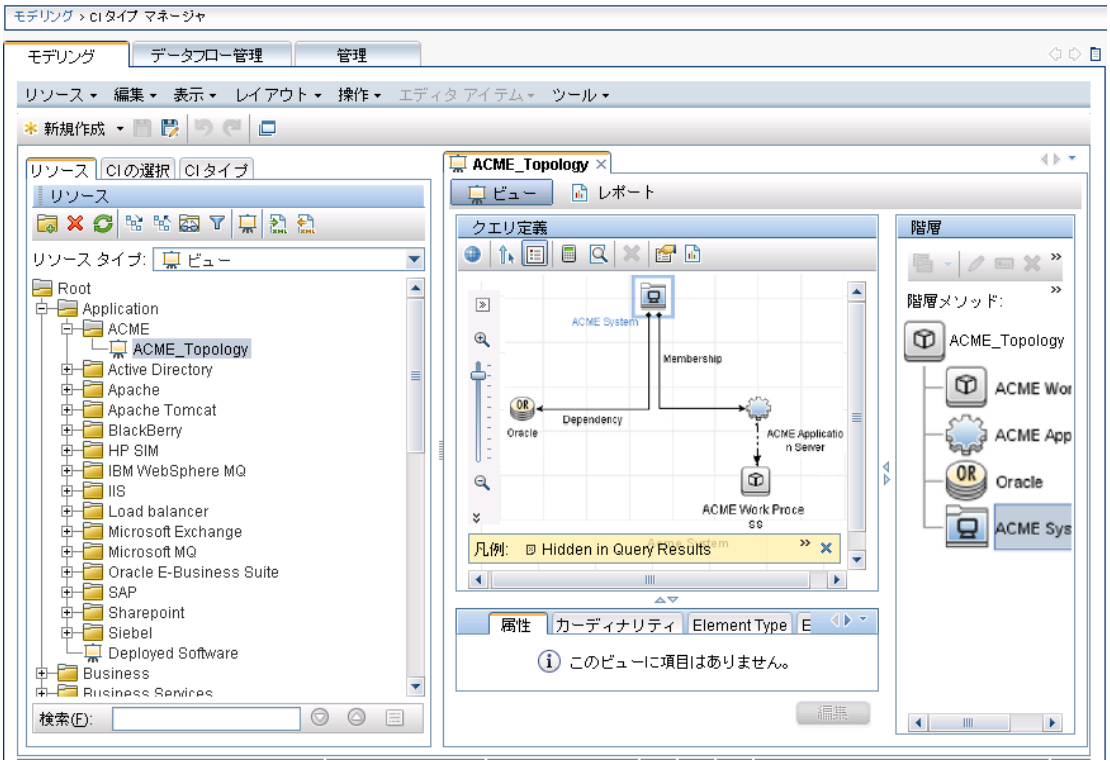


この図は，ACME システムと Oracle データベースの間にクロスドメイン関係がある ACME モデルを示しています。ACME ディスカバリが Oracle データベース CI を作成するために十分なキー属性がない場合，この依存関係を強化ルールによって生成できます。

強化ルールの作成および管理の詳細については，『Modeling Guide』の「Enrichment Manager」セクションを参照してください。

新規アプリケーションのトポロジ・ビュー

ACME トポロジを表示するビューを作成するには，RTSM Modeling Studio を使用します。次のスクリーンショットは，RTSM Modeling Studio で表示した ACME トポロジ・ビューを示しています。



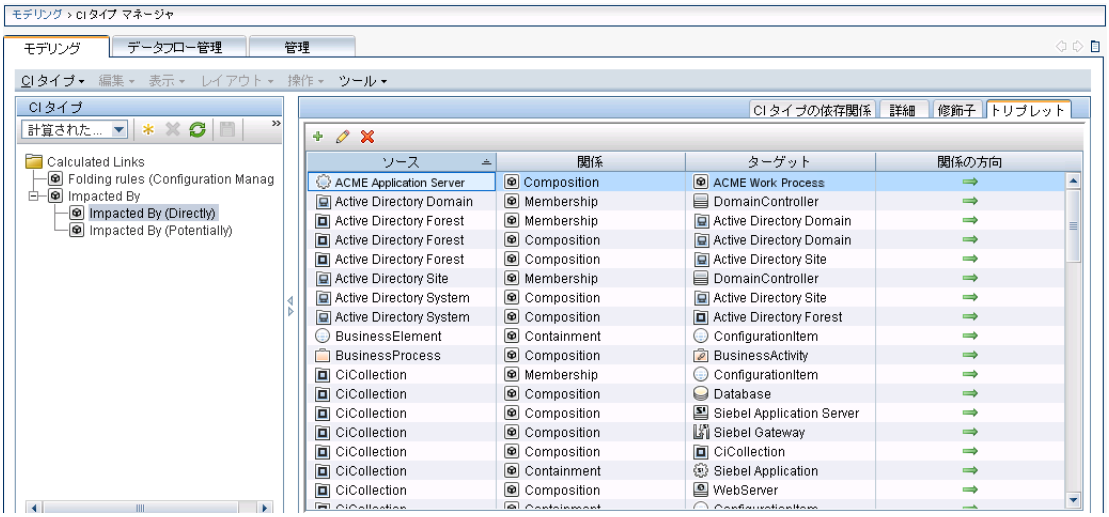
RTSM Modeling Studio の ACME トポロジ・ビュー

影響の伝搬

影響モデリングは計算された関係を使用して実行されます。KPI の計算では影響関係が重要となります。RTSM での影響モデリングの概念を含む詳細については，『Modeling Guide』を参照してください。

注 : トポロジ内の影響の伝搬を検証するビューを作成できます。新しいビューを作成し，CI タイプ間の関係タイプとして `impacted by` を選択します。

次の図は，伝搬ルールを使用して ACME アプリケーション・サーバと ACME ワーク・プロセスの間の影響関係を作成する計算された関係 (トリプレット) を示しています。



ACME アプリケーション・サーバとACME ワーク・プロセス間 の影 響 関 係 の作 成

イベント・タイプ・インジケータと状況インジケータ

本項では，BSM ソリューションに含まれる高度なイベント相関処理機能および状況ベースの監視機能を利用できるように新しいアプリケーションのイベントを強化する方法について説明します。

ACME の例では，HPOM が ACME ランドスケープのスマート・プラグインを提供することを前提としています。HP Operations スマート・プラグインがなければ，ユーザが ACME アプリケーションを分析してそのインタフェースを特定し，そのようなランドスケープを監視するためのポリシーを作成しなければなりません。

高度なイベント相関処理機能および状況ベースの監視機能を使用するには，次の手順を実行します。

- RTSM に CI を追加し，オペレーション管理 イベントを RTSM 内の適切な CI にマップします。
- さまざまな CI タイプの受信イベントを分析し，意味のあるイベント・タイプ・インジケータ(ETI) および状況インジケータ(HI) を作成します。
- HI を状況ベースの主要管理指標(KPI) に割り当てます。

イベントの CI へのマッピング

オペレーション管理 が提供する高度なイベント監視機能および状況監視機能の基本的な要件は，RTSM でイベントが適切な CI にマップされていることです。

イベントが適切な CI にマップされていなければ，次のような結果になります。

- HPOM メッセージにイベント・タイプ・インジケータまたは状況インジケータを設定しても何の効果もありません。
- 相関処理ルールがトリガしません。
- 状況パースペクティブ・ビューに誤った状況とKPI データが表示されるか，何も表示されません。

そのため，次のことが必要となります。

- RTSM に CI が追加されている。
- それらの CI にイベントをマップできるように，イベント統合を(必要に応じて) 調整する。

イベントを CI にマップするには，スマート・マッピング・テクノロジを使用します。特定のイベント属性でヒントが検索され，それらのヒントが CI 属性と比較され，イベントが最も一致する CI にマップされます。

ほとんどのイベントには，影響を受けるホストの DNS 名が必ず含まれています(HPOM メッセージ・ノード名フィールド)。この DNS 名が 1 つのヒントとして使用されるため，スマート・マッピングでは，ほぼ必ず受信イベントを最低でもホスト CI にだけはマップできます(もちろん，ホスト CI が RTSM に存在していることが前提となります)。

ただし，複雑な IT トポロジ・モデルを活用するためには，次のようにマップすることが重要となります。

- データベース・イベントを対応するデータベース CI にマップします。
- ほかのアプリケーションに関連するイベントを各アプリケーションの CI にマップします。

このようなマッピングを実現するには，次に挙げる追加のイベント属性を評価します。

- アプリケーション
- オブジェクト
- HPOM サービス ID 属性または CI 解決ヒント属性

  CI 解決ヒント属性を設定した場合，HPOM サービス ID 属性は無視されます。

これらの属性に設定できる識別ヒントが多いほど，イベントが適切な CI にマップされる可能性が高まります。

スマート・マッピングにおいて何が属性に属しているかを特定できるように，ヒントを特定の形式で指定する必要があります。

標準設定の形式は次のとおりです。

- `<hint 1>:<hint 2>:...:<hint n>`(アプリケーション属性およびオブジェクト属性の場合)
- `<hint 1>:<hint 2>:...:<hint n>@@<hostname>`(HPOM サービス ID 属性および CI 解決ヒント属性の場合)

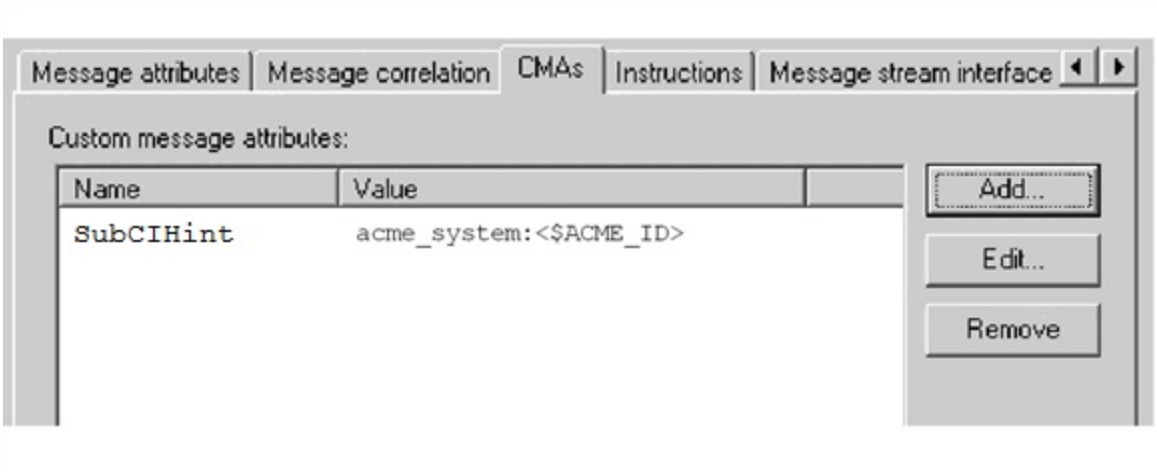
区切り文字(:)は，BSM インフラストラクチャ設定で設定できます。詳細については，HP Business Service Management のオンライン・ヘルプを参照してください。

指定したすべてのヒントが評価され，RTSM 内で一致する CI が検出されます。アプリケーション属性，オブジェクト属性，および HPOM サービス ID 属性のヒントは，下位互換性確保のために評価されます。以前は，多くの HP Operations スマート・プラグインがこれらのフィールドをイベントがどのオブジェクト(RTSM では構成アイテム)に属するかに関する情報の伝送に使用していました。この情報が適切な CI を特定するのに十分であれば，何も変更する必要はありません。

ただし，イベントが不適切な CI に関連付けられていることがわかった場合は，CI 解決ヒント属性に必要なヒントを設定する必要があります。これを行うには，HPOM で HPOM メッセージに `RelatedCiHint` と呼ばれるカスタム・メッセージ属性(CMA)を設定します。

カスタム・メッセージ属性 RelatedCiHint の設定

次の図に，HPOM メッセージに CMA `RelatedCiHint` を設定する例を示します。



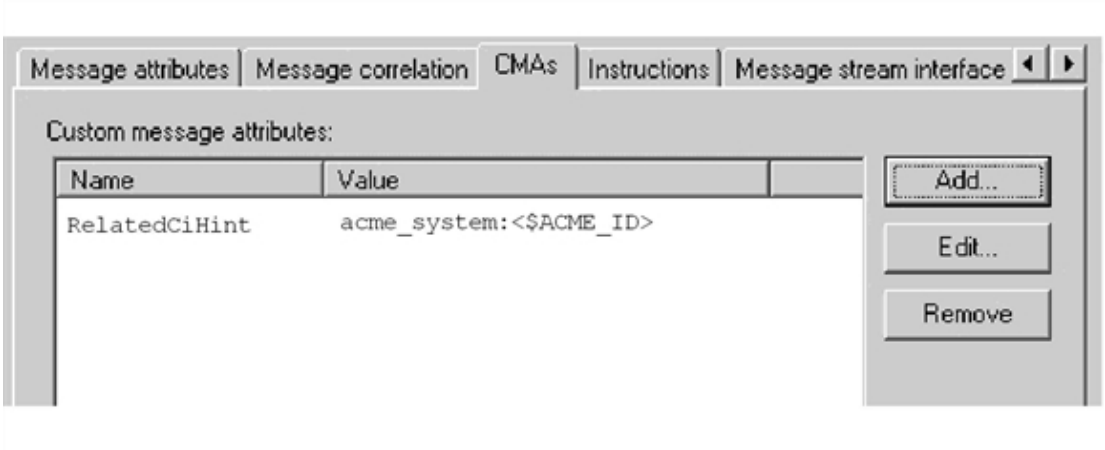
HPOM メッセージでのカスタム・メッセージ属性 RelatedCiHint の設定

RelatedCiHint 変数を設定するときには，次のベスト・プラクティスに関する考慮事項に留意することが重要です。

- CMA RelatedCiHint には，対応する CI を見つけるために十分なヒントが含まれている必要があります。
- ホストとの構成関係を持つ CI とそのような関係を持たない CI を区別する必要があります。

カスタム・メッセージ属性 SubCiHint の設定

次の図に，HPOM メッセージに CMA RelatedCiHint を設定する例を示します。



HPOM メッセージでのカスタム・メッセージ属性 SubCiHint の設定

SubCiHint 変数を設定するときには，次のベスト・プラクティスに関する考慮事項に留意することが重要です。

- CMA SubCiHint には，対応する CI を見つけるために十分なヒントが含まれている必要があります。
- ホストとの構成関係を持つ CI とそのような関係を持たない CI を区別する必要があります。

RelatedCiHint の値

通常，RelatedCiHint 変数には次の値を設定する必要があります。

- **“ホスト元” CI の場合**

  <CI-Typename>:<key-attribute-1>:<key-attribute-2>:<key-attribute-n>@@hostname

  通常，“ホスト元” CI は “ソフトウェア・エレメント” のサブタイプです。たとえば，websphereas タイプの CI には，ホストとの間に container-link 関係があります。

  もう 1 つの例として，exchangeclientaccessserver タイプの Exchange Server ロール CI があります。この CI タイプの root-container はソフトウェア・エレメントであり，その CI タイプでは root-container はホストです。

- **仮想 CI の場合**

  <CI-Typename>:<key-attribute-1>:<key-attribute-2>:<key-attribute-n>

仮想 CI には，ホストとの間に強い包含関係 (container-link または root-container) はありません。

一般的な CI タイプの例として，`cluster` があります。この CI タイプには，ホストとの間に強い包含関係はありません。

イベント・タイプ・インジケータおよび状況インジケータの作成

監視対象の HPOM イベントを RTSM の適切な CI にマップしたら，次にイベント・タイプ・インジケータと状況インジケータを作成します。

イベントの分析とインジケータの定義

意味のあるイベント・タイプ・インジケータ(ETI) および状況インジケータ(HI) を作成するためには，さまざまな CI タイプの受信イベントを分析する必要があります。

ETI はイベントの属性です(自体がインスタンスとしては存在しません) 。ETI を使用して，受信イベントを管理対象 IT 環境での発生のタイプに従って分類します。たとえば，HPOM のメッセージにイベント・タイプ属性を設定するカスタム・メッセージ属性 (CMA) の `EtiHint` を追加することもできます。CMA を設定しない場合は，適用可能なマッピング・ルールを使用して ETI を設定できます。ETI には，環境内でのイベントの発生を説明する 1 つ以上の値が必要です。たとえば，`Lost database Connection:Occurred` のように値を設定します。

監視対象のシステムにおける Operations Management イベントの原因となる特定のタイプの発生には，同じ ETI を割り当てる必要があります。適切な相関処理ルールの定義後，イベントは ETI に基づいて相関処理されます。相関処理ルールでは，CI で生じるイベントのタイプを関連付けます。

HI は，監視対象の CI の指定されたアスペクトの状況を特定し，表示します。HI は，ハードウェア・リソースが使用可能かどうかを示し，1 つの値によって CI の正常な状態を表します。たとえば，`ACME System Status:Available` のように示されます。CI の異常は，`ACME System Status:Unavailable` などの 1 つ以上の値で示されます。

HI では，特定のプロセスへの負荷が正常，高，または超過である場合などに，ソフトウェア・アプリケーションの状態を示すこともできます。異常な状態の例として，`ACME Work Process` CI タイプの `Job Queue Length:Too Long` があります。

ヘルス・インジケータは，CI の状況の情報を提供するイベントにのみ設定できます。状況インジケータは，関連付けられている ETI を通して特定の構成アイテム・タイプに割り当てられています。

HI は，監視対象のリソースの可用性およびパフォーマンスを計算して主要管理指標 (KPI) を導き出すために必要なデータも提供します。156ページ「HI の KPI への割り当て」に，ACME 社の例の HI を状況ベースの KPI に割り当てる方法を示します。

例として挙げている ACME 社の環境の新しい CI タイプについては，136ページ「新規アプリケーションの統合」で説明しました。また，136ページ「新規アプリケーションの統合」の図も参照してください。

これらの CI タイプには，具体的な ETI および HI があります。151ページ「ETI および HI の概要」に，ACME 社の環境においてどの ETI/HI が重要であるかについて検討した結果を示します。

ETI および HI の概要

CI タイプ表示名	カテゴリ	名前	値	重大度	ポリシー
ACME System	HI	ACME System Status	Available	正常域	ACME_SystemStatus001
ACME System	HI	ACME System Status	Unavailable	危険域	ACME_SystemStatus001
ACME Application Server	HI	ACME Application Server Status	Available	正常域	ACME_AppServerStatus_001
ACME Application Server	HI	ACME Application Server Status	Unavailable	重要警戒域	ACME_AppServerStatus_001
ACME Work Process	ETI	Job Aborted	Occurred		ACME_opcmsg_001
ACME Work Process	ETI	Job Start Passed	Occurred		ACME_opcmsg_002
ACME Work Process	HI	Job Queue Length	Normal	正常域	ACME_JobQueue001
ACME Work Process	HI	Job Queue Length	Too Long	重要警戒域	ACME_JobQueue001
ACME Work Process	ETI	Lost Database Connection	Occurred		ACME_LogFile001

注: 状況インジケータは，監視対象オブジェクトの健全性の現在の状態を示す必要があります。そのため，イベントは監視対象オブジェクトの健全性が継続的に監視されている場合にのみ HI を設定する必要があります。

HI と ETI は，BSM ユーザ・インタフェースの次の領域で作成します。

[管理] > [オペレーション管理] > [監視] > [インジケータ]

HI と ETI の作成方法の詳細については，HP Business Service Management のオンライン・ヘルプを参照してください。

次に，CI タイプ ACME System の HI の作成例を示します。

1. 次の場所まで移動します。

   [管理] > [オペレーション管理] > [監視] > [インジケータ]

2. [CI タイプ] 表示枠で，インジケータを設定する CI タイプ(この例では ACME System) を右クリックします。

3. [新規]( ) ボタンをクリックし，作成するインジケータの種類として，[状況インジケータ]または[イベント タイプ インジケータ]を選択します。ここでは，[**状況インジケータ**]を選択します。[新規状況インジケータ]ダイアログが開きます。

4. [新規状況インジケータ]ダイアログの[一般]領域に次の情報を入力します。

   [表示名]フィールドに，作成するHIの表示名を「**`ACME System Status`**」と入力します。

   [名前]フィールドには，標準設定の名前が自動的に入力されます。たとえば，ターゲット HP Service Manager サーバに対する表示名を「**`ACME System Status`**」と入力すると，[名前]フィールドには `ACME_Service_Status` が自動的に表示されます。もちろん，この標準設定の名前を変更する場合は，[名前]フィールドに別の名前を入力できます。

   オプション。[説明]フィールドに，作成しているCIタイプの説明を入力します。

   [アプリケーション]フィールドには，HIの対象となるアプリケーションとして，[サービス状況]，[SLM]，[サービス状況とSLMの両方]のいずれかを選択します。

5. [新規状況インジケータ]ダイアログの[状態]領域では，[新規]( ) ボタンをクリックしてインジケータの状態を追加するか，既存の状態を編集します。

この例では，新しいインジケータ状態を追加します。重大度が正常域で値がAVAILABLEである標準の状態と，重大度が危険域で値がUNAVAILABLEの状態を追加します。

値がAVAILABLEで重大度が正常域であるインジケータの標準の状態を追加するには，次の手順を実行します。

a. [新規]( ) ボタンをクリックします。[インジケータ状態の編集]ダイアログが開きます。

b. [**表示名**]フィールドにHIインジケータの状態の表示名を「AVAILABLE」と入力します。

c. [**名前**]フィールドにHIインジケータの状態のシステム名を「AVAILABLE」と入力します。

d. [**標準設定**]チェックボックスを選択して，この状態を標準の状態にします。

e. [ステータス]フィールドで[正常域]を選択します。

f. [**アイコン**]フィールドで，この正常域状態のアイコンを選択します。

g. [**保存**]をクリックします。

6. 値がUNAVAILABLEで重大度が危険域であるインジケータの状態を追加するには，次の手順を実行します。

a. [新規]( ) ボタンをクリックします。

b. [**表示名**]フィールドにHIインジケータの状態の表示名を「UNAVAILABLE」と入力します。

c. [**名前**]フィールドにHIインジケータの状態のシステム名を「UNAVAILABLE」と入力します。

d. [**標準設定**]チェックボックスの選択を解除します。

e. [**ステータス**]フィールドで[**危険域**]を選択します。

f. [**アイコン**]フィールドで，この危険域状態のアイコンを選択します。

g. [**保存**]をクリックします。

7. CIタイプに対して作成するほかのHIおよびETIについて上記の手順を繰り返します。

イベントへのイベント・タイプ・インジケータの割り当て

イベントにETIを割り当てるには，次の2つの方法があります。

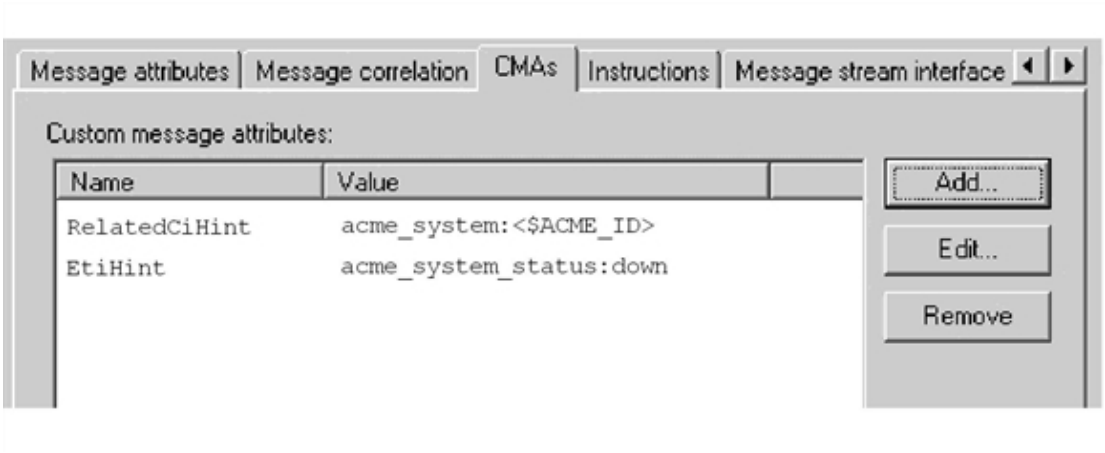
- ETIマッピング・ルールを使用する
- HPOMポリシー内にカスタム・メッセージ属性 EtiHint を設定する

どちらの方法を使用するかは，受信 HPOM メッセージの変更が必要となる可能性に大きく依存します。受信 HPOM messages/オペレーション管理 イベントの ETI を設定する場合は，CMA を使用することをお勧めします。ただし，ETI マッピング・ルールを使用して ETI をイベントに割り当てることもできます。

CMA “EtiHint” による ETI の設定

151ページ「ETI および HI の概要」に，ACME 監視システムの分析された ETI および HI のリストがあります。右端の列に，イベントを作成した HPOM ポリシーの情報が示されています。これらの HPOM ポリシー条件に CMA `EtiHint` を追加する必要があります。

次の図に CMA `EtiHint` の設定例を示します。

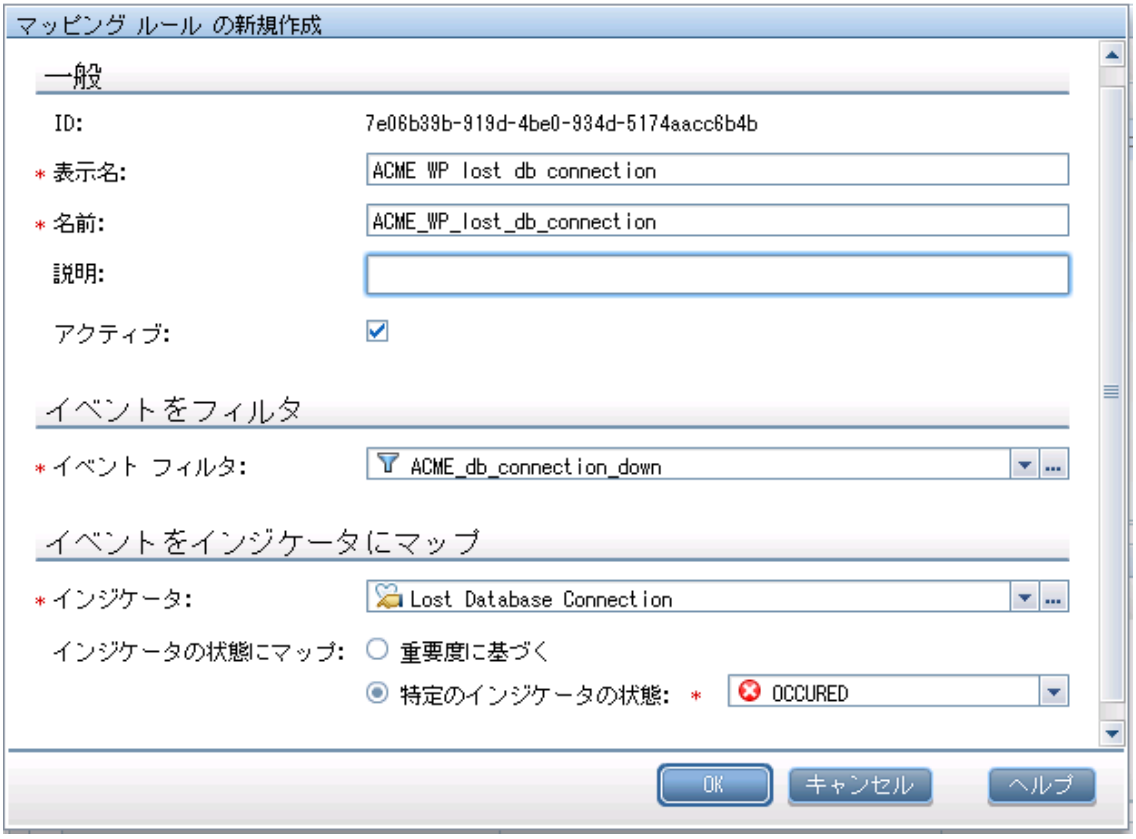


ETI マッピング・ルールによる ETI の設定

ACME 社の例では，すべてのメッセージが HPOM ポリシーを使用して送信されます。したがって，CMA を使用して ETI を設定します。ただし，マッピング・ルールを使用して受信 HPOM メッセージの ETI を設定することもできます。

次の図に `db connection down` 関連メッセージのフィルタ設定の例を示します。

次の図に，上記のフィルタ設定を使用して設定した，`Lost Database Connection` ETI のマッピング・ルールを示します。



このマッピング・ルールは，`Category=ACME`，`Sub Category=Work Process`，および `Title=db connection lost` である個々の受信メッセージを CI タイプ `ACME Work Process` の ETI `Lost Database Connection` に一致させます。

HI の KPI への割り当て

次に，HI を状況ベースの KPI に割り当てます。HI は，KPI が CI で表される監視対象リソースの可用性とパフォーマンスを計算するために必要なデータを提供します。BSM の次の領域で HI を KPI に割り当てます。

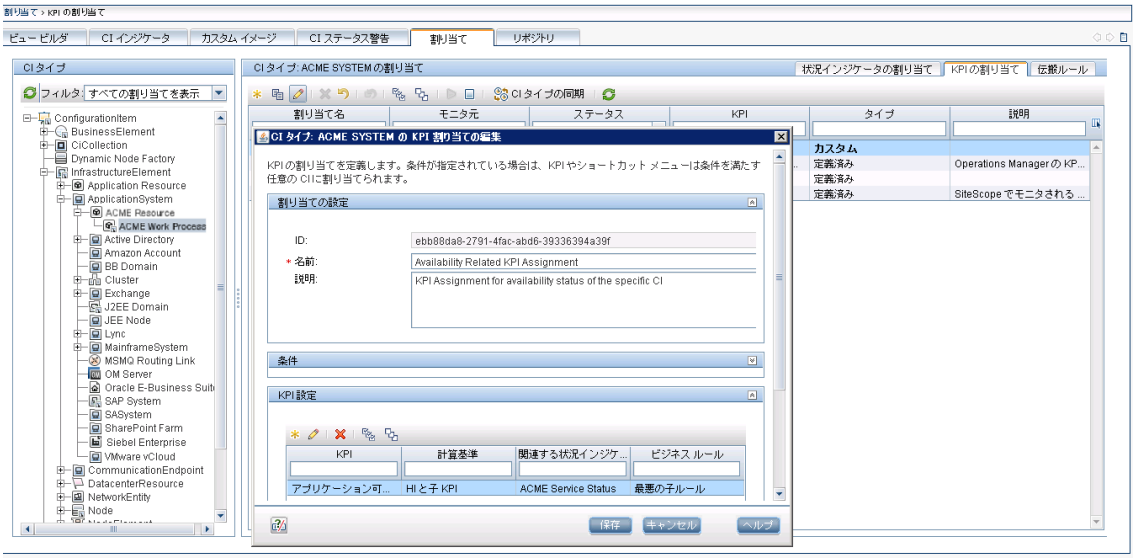
[管理] > [サービス状況] > [割り当て] > [KPI の割り当て]

KPI の割り当ての詳細については，HP Business Service Management のオンライン・ヘルプを参照してください。

151ページ「ETI および HI の概要」に示す ACME 社の HI は，ほとんどが特定の CI の可用性ステータスを表します。したがって，それらの HI は操作可用性の KPI に割り当てられます。

次の図は，HI `Job Queue Length` の操作可用性の KPI への割り当てを示しています。この設定の結果，HI `Job Queue Length` は `ACME Work Process` タイプの CI に関連するイベントの状況パースペクティブにも表示されます。

[CI タイプの KPI 割り当ての追加/編集]ダイアログの[条件]領域で，割り当てを CI のサブセット(特定のアプリケーションが監視する CI や特定の CI 属性セットを持つ CI など)に制限できます。詳細については，HP Business Service Management のオンライン・ヘルプの「インジケータの割り当ておよび伝搬」のセクションを参照してください。



HI の KPI への割り当てが完了したら，CI タイプの同期を実行して，その割り当てを既存の CI に対して有効にしてください。

HPOM のカスタム・メッセージ属性

次の表に，HPOM メッセージに指定でき，対応するオペレーション管理のイベント・プロパティの値に影響する，サポートされているカスタム・メッセージ属性を示します。

HPOM の CMA

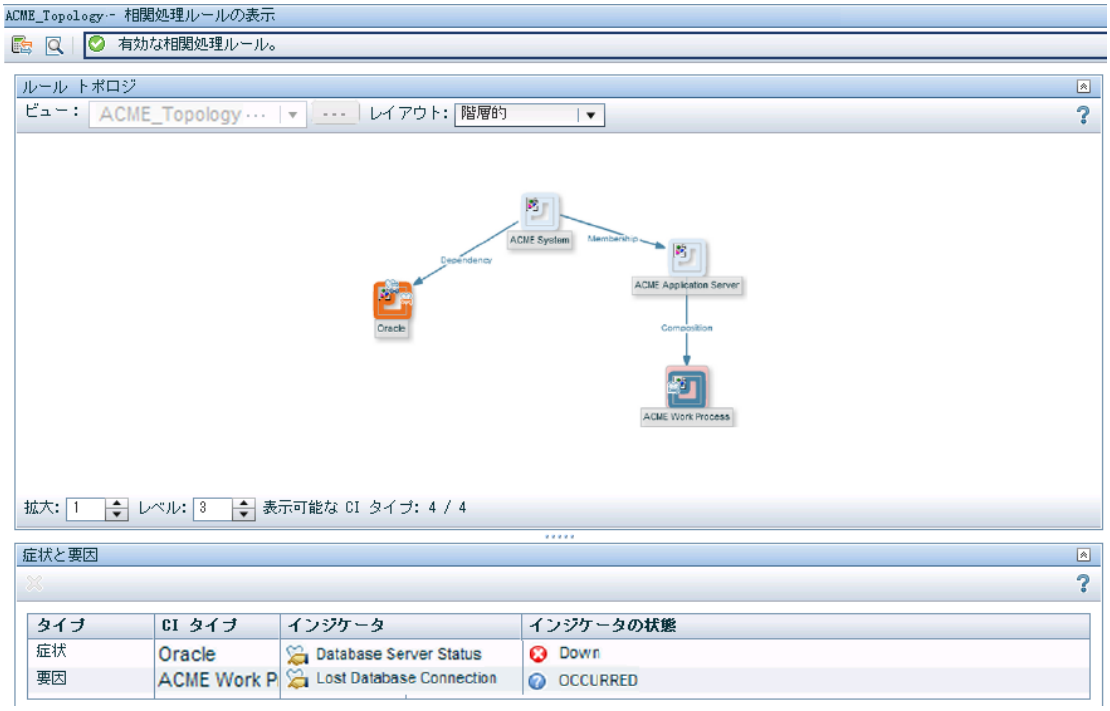
CMA	説明
Description	イベントの原因を説明するテキスト。
NoDuplicateSuppression	既存のイベントと重複するイベントが抑制されない。
Solution	イベントが示す問題の解決に役立つソリューションが記述されたテキスト・フィールド。
SubCategory	イベントが属する論理サブグループ(カテゴリ) の名前 (Oracle (データベース) , Accounts(セキュリティ) , Routers(ネットワーク) など) 。
EtiHint	各 CI の ETI を識別するためのイベント情報。
NodeHint	イベントに関連するノード CI を識別するためのイベント情報。
RelatedCiHint	イベントに関連する CI を識別するためのイベント情報。
SourceCiHint	イベントに関連するソース CI を識別するためのイベント情報。
SourcedFromExternalId	指定された ID を持つ外部システムから送信されるイベント。
SourcedFromExternalUrl	指定された URL を持つ Web アプリケーションから送信されるイベント。
SourcedFromDnsName	指定された DNS 名を持つ外部システムから送信されるイベント。

相関処理ルールとマッピング

管理対象 IT 環境の同一ドメインまたは異なるドメインで発生する関連イベントを相関させる相関処理ルールを定義できます。トポロジ・ベースのイベント相関処理 (TBEC) を使用すると，問題の原因が自動的に特定され，表示されます。Top Level Items フィルタを使用して，要因イベントの症状のみであるイベントを除外すると，解決の必要がある実際の問題の概要を明確にすることができます。イベントの相関処理によって，イベント・ブラウザに表示されるイベントの数が少なくなり，問題を原因を迅速かつ効率的に特定できるようになります。

ACME ランドスケープのどのイベントがほかのイベントの症状や原因であるかを考え，相関処理ルールを定義できるかどうかを確認する必要があります。

たとえば，次の図に示すように，ACME ランドスケープには Oracle データベースとの依存関係があります。ここでは，そのようなデータベースの監視が HP Operations Smart Plug-In for Oracle (SPI for Oracle) によって実現され，データベースが使用不能になったときにイベントが受信されることを前提とします。同時に，いずれかの ACME ポリシーが切断されたデータベース接続を検出するため，これら 2 つの関連イベントの相関処理ルールを定義することは理に適っています。そのようなルールを次の図に示します。このルールは，ETI `Lost Database Connection Occurred` を持つイベントが症状イベントであり，ETI `Database Server Status Down` を持つイベントが原因イベントであることを定義します。



相関処理ルールを定義することが適切であるもう 1 つの例は，`long job queue` イベントが多くの `Job Start Passed` メッセージの原因である場合です。

もちろん，複数の症状および原因に対する複雑な TBEC ルールも定義できます。

相関処理ルールおよびマッピングの詳細については，HP Business Service Management のオンライン・ヘルプの「イベントの相関処理」セクションを参照してください。

追加のイベント処理

追加のイベント処理を実行することにより，イベントを変更および強化できます。

イベント処理インタフェース(EPI)を使用して，イベント処理においてユーザ定義の Groovy スクリプトを実行できます。外部 SQL データベースなどからイベントを強化できると便利な場合があります。

イベント処理パイプライン，イベント処理インタフェース，および EPI スクリプトの概要については，227ページ「イベント処理インタフェース」を参照してください。

イベントに適用するカスタム・アクションを設定することもできます。イベント・ブラウザでカスタム・アクションを使用できるようにする Groovy スクリプトを設定できます。

カスタム・アクションおよびスクリプトの概要については，231ページ「カスタム・アクションのスクリプト」を参照してください。

EPI およびカスタム・アクション・スクリプトの作成方法と製品に付属のサンプル Groovy スクリプトについては，232ページ「EPI スクリプトおよびカスタム・アクション・スクリプトの作成」を参照してください。

ツール

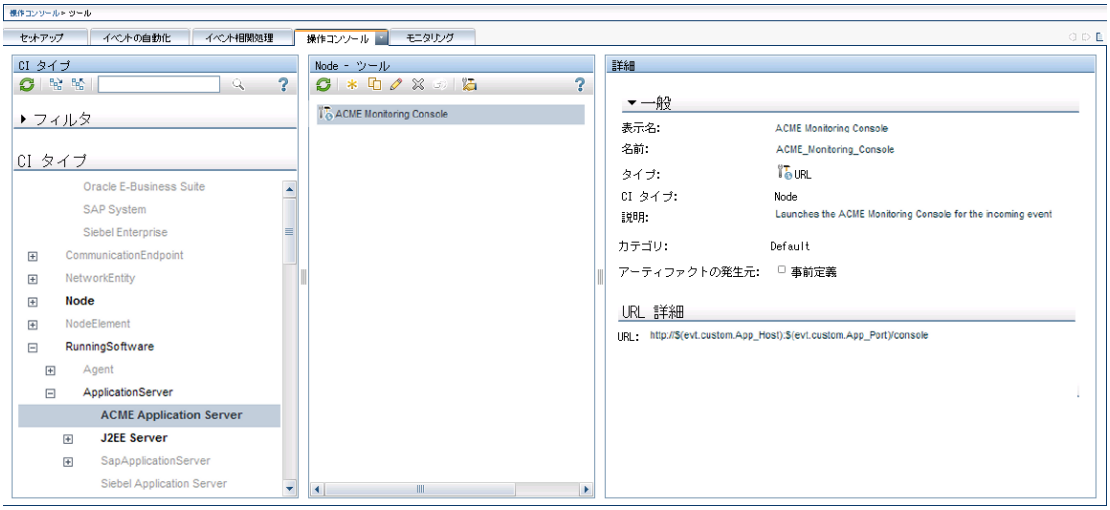
ドメイン・オペレータによる特 定 のイベントおよび CI の管 理 と監 視 や新 しいアプリケーション領 域 で発 生 した問 題 の解 決 に役 立 つツールを設 定 できます。

ツールを特 定 の CI タイプに割 り当 てることにより, その CI タイプのインスタンスに影 響 するイベントのコンテキストで割 り当 てたツールが常 に使 用 できるようになります。

ACME の例 には, オペレーション管 理 イベント・ブラウザから ACME 管 理 者 コンソールを起 動 する URL があります。

ツールの詳 細 については, オペレーション管 理 のオンライン・ヘルプを参 照 してください。

次 の図 に URL による ACME 監 視 コンソールの起 動 の例 を示 します。このツールは, カスタム・イベント属 性 `App_Host` および `App_Port` を使 用 して, 適 切 な URL を生 成 します。

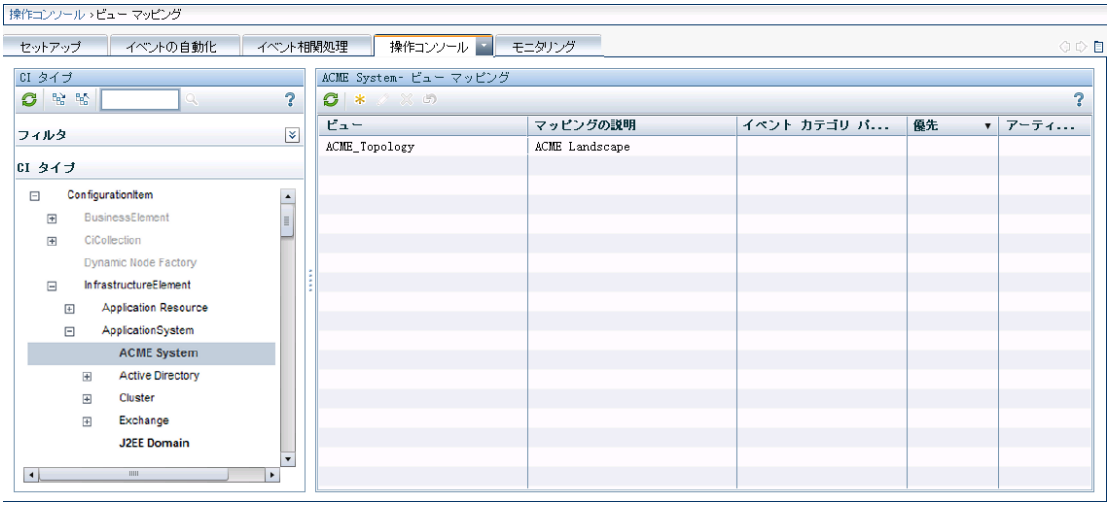


ビュー・マッピング

ビューを[ヘルストップビュー]表示枠での選択および使用の対象とするには，イベント・ブラウザで選択されたイベントの影響を受ける構成アイテムを 1 つ以上の既存のビューに明示的にマップする必要があります。RTSM モデリング・スタジオを使用して，RTSM に新しいビューを作成できます。CI タイプを複数のビューに関連付けることができます。複数のビューは優先順位に従って表示され，優先順位の最も高いビューが標準設定ビューとなります。

ビュー・マッピング・マネージャを使用して，CI タイプを RTSM ビューにマップできます。選択したイベントに関連する CI タイプにマップされたビューだけが[ヘルストップビュー]表示枠の[選択されたビュー]ドロップダウン・リストに表示されます。

次の図に[ビュー マッピング]表示枠を示します。CI タイプ `ACME System` は，ビュー `ACME_Topology` にマップ(関連付け) されています。



グラフ

パフォーマンス・グラフを特定のCIタイプに関連付けることにより，そのCIタイプに影響するイベントのコンテキストで特定のタイプのグラフおよびチャートが常に使用できるようになります。

新しいACMEアプリケーションをこのように設定するには，次の手順を実行します。

- パフォーマンス・グラフ作成ツールを使用して，新しいグラフ・テンプレートを作成するか，既存のグラフ・テンプレートを編集します。詳細については，オペレーション管理のオンライン・ヘルプを参照してください。
- 適切なグラフ・ファミリまたはカテゴリをACME CIタイプにマップします。

パフォーマンス・データの統合

パフォーマンス・データの統合プロセスには次の操作が含まれます。

- メトリクスの収集
- メトリクスの保存
- メトリクスを使用したグラフの作成

たとえば，HPOMのパフォーマンス・データは，HPOMエージェントの組み込みパフォーマンス・コンポーネント(EPC)またはHP Performance Agentによって収集できます。

これらのエージェントで収集したデータは，HPOM測定値しきい値ポリシーでアラーム・メッセージの作成に使用できます。また，HP Performance Managerまたはオペレーション管理のPerformance Grapherを使用して履歴データを表示することで，オペレータによる問題の分析および解決に役立てることができます。

パフォーマンス・データを統合する簡単な方法として，HPOM測定値しきい値ポリシー・タイプを使用することがあります。

HPOM測定値しきい値ポリシーでは，次のソースからパフォーマンス・データを収集できます。

- 外部/プログラム – 外部プログラムから送信されたデータ(`opcmon`コマンド・ライン・インタフェースまたはAPIを使用)
- MIB – SNMP Management Information Base(MIB)からメトリクスを収集
- リアルタイム・パフォーマンス – Windowsパフォーマンス・カウンタからメトリクスを収集
- WMI – Windows Management Instrumentationからメトリクスを収集

これらのソースから収集したメトリクスは，`[Store in Embedded Performance Component]`オプションを使用して簡単に組み込みパフォーマンス・コンポーネントに保存できます。

コンテンツのパッケージ化

本項では，あるBSM オペレーション管理 インスタンスで作成したコンテンツを別のインスタンスへ移動する方法について説明します。

作成したコンテンツを別のBSMオペレーション管理 インスタンスで使用する場合は，次の手順を実行する必要があります。

- RTSM パッケージを作成してエクスポートします。
- ACME コンテンツ・パックを作成してエクスポートします。
- トポロジ同期ファイルをコピーします。
- トポロジ同期ファイルをコンテンツ移行先のBSMオペレーション管理 インスタンスにアップロードします。

RTSM パッケージの作成

RTSM パッケージは，コンテンツの不可欠の要素です。RTSM パッケージを使用することにより，ビュー・モデルや構成アイテム・タイプを管理できます。たとえば，パッケージを使用して，同一サーバ上またはBSMオペレーション管理 の異なるインスタンス間で，コンテンツをエクスポート，インポート，および更新できます。

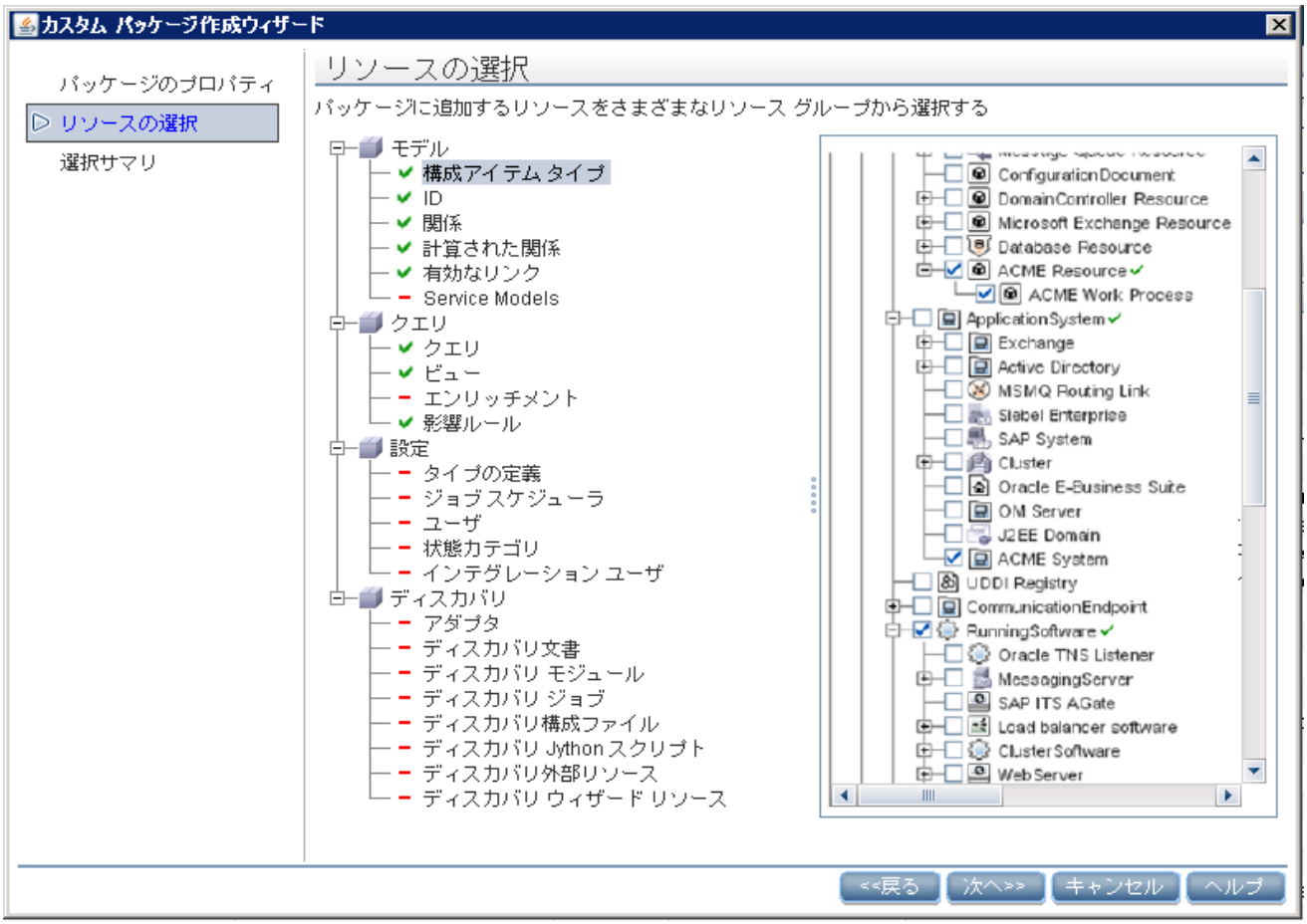
次に，ACME RTSM パッケージを作成するためのワークフローを示します。

1. ACME RTSM パッケージを作成します。

   コンテンツ・パックの作成方法の詳細については，HP Business Service Management のオンライン・ヘルプを参照してください。

2. 前の手順で作成したCI タイプ(136ページ「トポロジ」を参照) とビュー(161ページ「ビュー・マッピング」を参照) を追加します。

次の図に示す画面では，ACME RTSM パッケージに追加するアイテムを選択しています。通常，このようなパッケージには，CI タイプ，TQL，およびビューを追加します。



トポロジ同期ファイルの保存

トポロジ同期ファイルを収集して保存し，別のインスタンスにコピーすることができます。

トポロジ同期ファイルは，次の場所にあります。

`<HPBSM ルート・ディレクトリ>/conf/opr/topology-sync/sync-packages`

トポロジ同期マッピング・ルールの作成方法の詳細については，167ページ「実行時サービス・モデルへのデータの追加」を参照してください。

コンテンツ・パックの作成

コンテンツ・パックには，IT 環境の管理に役立つように定義および設定するルール，ツール，グラフ，マッピング，および割り当ての全部 (または一部) の完全なスナップショットを含めることができます。

コンテンツをコンテンツ・パック間で共有できます。コンテンツ・パック定義内で，コンテンツを直接コンテンツ・パックに含めることができます。コンテンツ・パック定義に含めたコンテンツは，参照コンテンツに依存している可能性があります。参照コンテンツとは，コンテンツ・パックに明示的に含まれておらず，ほかのコンテンツ・パック定義で参照されているコンテンツをいいます。たとえば，相関処理ルールをコンテンツとしてコンテンツ・パックに含めると，そのルールが別のコンテンツ・パック定義で参照されているインジケータを必要とする可能性があります。

参照コンテンツの処理方法は，参照コンテンツはほかのコンテンツ・パック定義に含まれていない場合にのみコンテンツ・パック定義に追加されるという原則に基づいて決定されます。したがって，ほとんどの場合，コンテンツ・パック定義はほかのコンテンツ・パック定義との間に依存関係があります。これは，ほかのコンテンツ・パックとの依存関係がどのような影響をもたらすかを考慮する必要があることを意味します。たとえば，コンテンツ・パックを別のシステムにインポートするときには，コンテンツ・パックをロードする順序が重要となる可能性があり，もちろんインポート先のシステムには参照コンテンツが存在している必要があります。

参照コンテンツの処理について次の事例で簡単に説明します。

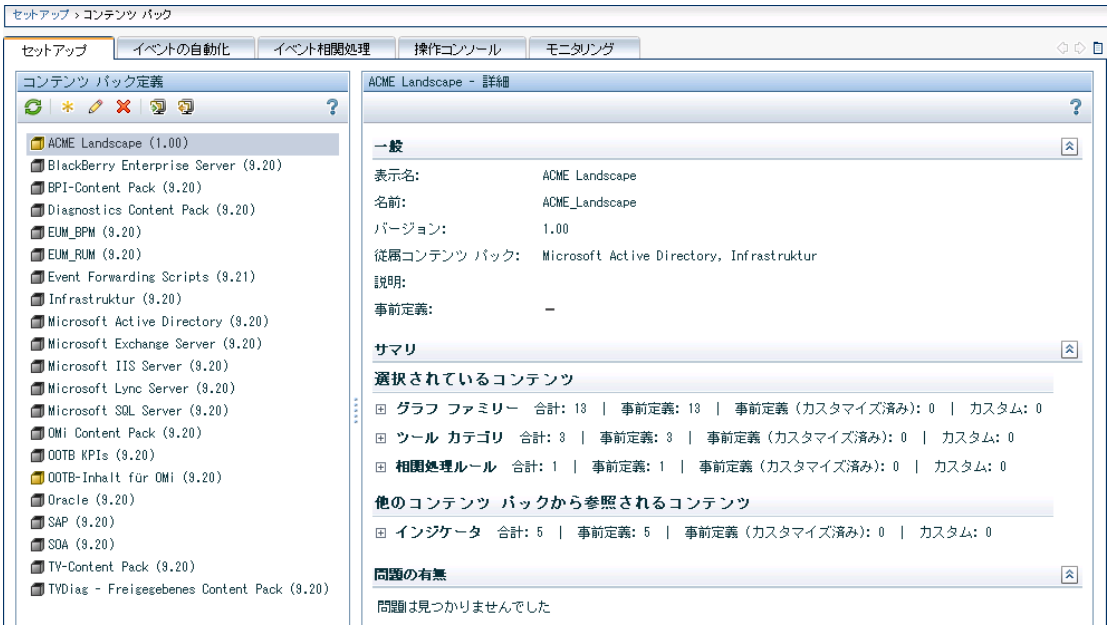
- 事例 1: 参照コンテンツがほかのコンテンツ・パック定義に含まれていない。この場合，コンテンツは作成するコンテンツ・パックに追加されます。
- 事例 2: 参照コンテンツがほかのコンテンツ・パック定義に含まれている。この場合，コンテンツは現在のコンテンツ・パックに追加されず，参照コンテンツへの依存関係がほかのコンテンツ・パック定義に対してマークされます。
- 事例 3: 参照コンテンツが複数のコンテンツ・パック定義に含まれている。この場合は，事例 2 と同じように，参照コンテンツへの依存関係がその参照コンテンツを含むいずれかのコンテンツ・パック定義に対してマークされます。

次に，ACME コンテンツ・パックを作成するためのワークフローを示します。

- ACME コンテンツ・パックを作成します。

  コンテンツ・パックの作成方法の詳細については，HP Business Service Management のオンライン・ヘルプを参照してください。
- 前の手順で作成した相関処理ルール，HI，ETI，KPI 割り当て，ツール，グラフとグラフ割り当て，およびビュー・マッピングをすべて追加します。コンテンツ・パックには EPI スクリプトとカスタム・アクションも定義できます。
- オプション。依存関係にあるコンテンツ・パック定義のコンテンツを含めます。
- コンテンツ・マネージャを使用してコンテンツ・パックを XML ファイル(ACME 環境のコンテンツ・パックの場合は ACME.xml) にエクスポートします。コンテンツ管理ツールを使用してパッケージを定義し，作成することにより，BSMオペレーション管理 のインスタンス間でコンテンツを交換できます。作成したパッケージをファイルにエクスポートし，そのファイルを使用して同じコンテンツを BSMオペレーション管理 の別のインスタンスに展開できます。

次の図に，コンテンツ・マネージャで ACME 環境のコンテンツ・パックからコンテンツを選択した後の状態を示します。



コンテンツのアップロード

エクスポートしたファイルを別 の BSM オペレーション管 理 インスタンスにコピーし，それらのファイルを次 の手 順 で移 行 先 のシステムにアップロードします。

- RTSM パッケージをアップロードします。
- トポロジ同 期 ファイルをコピーします。
- コンテンツ・パックをアップロードします。

RTSM パッケージのアップロード

RTSM パッケージをアップロード するには，次 の手 順 を実 行 します。

1. ZIP ファイルをファイル・システムの任 意 のディレクトリに置 きます。
2. RTSM パッケージ・マネージャを使 用 して RTSM パッケージをアップロードします。

トポロジ同 期 ファイルのコピー

トポロジ同 期 ファイルを移 行 先 のシステムの次 の場 所 にコピーします。

<HPBSM ルート・ディレクトリ>/conf/opr/topology-sync/sync-packages

コンテンツ・パックのアップロード

コンテンツ・パックをアップロード するには，次 の手 順 を実 行 します。

1. エクスポートした XML ファイルをファイル・システムの任 意 のディレクトリに置 きます。
2. コンテンツ・マネージャを使 用 してコンテンツ・パックをアップロードします。

第9章

実行時サービス・モデルへのデータの追加

本項では，開発者を対象として，実行時サービス・モデル(RTSM)にHPOM内のノードおよびサービスから構成アイテム(CI)およびCI関係を追加するためのトポロジ同期マッピング・ルールを作成する方法について説明します。

本項では，「セクションI: コンテンツの展開」で紹介したACME環境の例を使用して，特定のサービス・モデル用のトポロジ同期ルールを作成する方法について説明します。

本項の内容

トポロジ同期の概要

トポロジ同期をHP DFM(データフロー管理)ディスカバリの代わりに使用するか，または両方を組み合わせることによって，RTSM(実行時サービス・モデル)にCIを作成できます。トポロジベースのイベント相関処理(TBEC)，コンテキスト固有のツール，およびBSM全体のサービス状況監視(状況パースペクティブ)が動作するためには，RTSMに正確かつ最新のCIトポロジ・データが存在している必要があります。

注: また，既存のトポロジ情報をRTSMにインポートすることもできます。Excelブックなどの外部ソースからのデータ・インポートの詳細については，『HP Universal CMDB Discovery and Integration Content Guide』の「Import From Excel Workbook Discovery」および「Importing Data from External Sources」を参照してください。このガイドは，HPソフトウェア製品マニュアルのWebサイト(http://support.openview.hp.com/selfsolve/manuals)のUniversal CMDB (Application Mapping)製品のページにあります。

トポロジ同期は，HP Operations Manager(HPOM)の監視対象であるサービス，ノード，およびノード・グループを運用データベース(RTSM)のOMi構成アイテム(CI)にどのようにマップするかを決定するルールのセットとして定義できます。

トポロジ同期はOMiライセンスに標準で含まれており，HPOMに定義されたサービス・モデル(またはサービス・ツリーやサービス・グラフ)からRTSMにCIを追加するためのソリューションとなります。トポロジ同期は，特に独自のサービス・モデルを使用しているHPOMユーザに適しています。

トポロジ同期で使用されるマッピング・ルールは，トポロジ同期パッケージに含まれています。製品のインストールでは，トポロジ同期ルールはローカル・ファイル・システムにコピーされます。これらのルールは，動的同期が最初に使用されたときにデータベースにアップロードされます。既存のサービス・モデルまたは検出されたトポロジ・データに基づいてRTSMにCIを追加する独自のトポロジ同期ルールを作成することもできます。

トポロジ同期では，RTSM APIを使用し，スクリプティング・テクノロジとして，Wiseman，JAXB，JDOM，JXPath，SPRING，Hibernate，およびGroovyを使用します。

トポロジ同期には次の2つの方法があります。

- 169ページ「動的トポロジ同期」

  注: 推奨される方法は動的トポロジ同期です。この方法は動的トポロジ同期をサポートするすべてのHPOMバージョンで使用する必要があります。サポートされているHPOMのバージョンの詳細については，『HP Business Service Management リリース・ノート』を参照してください。

  動的トポロジ同期では，変更を検出すると，すぐに複数のHPOMから検出されたトポロジ・データを受け取り，RTSM内のCIおよびCI関係を更新します。

- 175ページ「基本トポロジ同期」

  注: 基本トポロジ同期は，HPOMの動的トポロジ同期をサポートしないバージョンでのみ使用します。HPOMのサポートされているバージョンの詳細については，『HP Business Service Management リリース・ノート』を参照してください。

基本トポロジ同期では，HPOM Web サービスを使用して，HPOM サービス，ノード，およびノード・グループからのトポロジ・データを 1 つの HPOM サーバからオペレーション管理に転送できます。

どちらのトポロジ同期方法でも同じ同期パッケージを使用してディスカバリ・データを RTSM 内の CI および CI 関係にマップします。

動的トポロジ同期

動的トポロジ同期によって，HPOM から取得したトポロジ・データを分散型の階層的環境において HPOM とオペレーション管理のサポートされる複数のインスタンス間で共有できます。

オペレーション管理のインスタンスは，HPOM およびほかのオペレーション管理の複数のインスタンスからトポロジ・データを受け取るように設定できます。また，トポロジ・データをオペレーション管理のほかのインスタンスに転送するようにオペレーション管理のインスタンスを設定することもできます。

トポロジ・データは，オペレーション管理のインスタンスに動的に転送されます。つまり，HP Operations Agent がトポロジの変更を検出すると，すぐにトポロジ・データが転送されます(さらに HPOM サービス・モデルに書き込まれます) 。動的トポロジ同期によって，インフラストラクチャの変更をほぼリアルタイムに検出できます。たとえば，環境にノードが追加されると，この変更がすぐに転送され，データベースが更新されます。

サポートされる HPOM のバージョンの詳細については，『HP Business Service Management リリース・ノート』を参照してください。

動的トポロジ同期を最初に設定したときには，ソース・システム(HP Operations Agent または HPOM) がそのすべてのトポロジ・データを 1 つまたは複数のターゲット・システム(HPOM または オペレーション管理) に送信します。この最初の同期の後，動的トポロジ同期では検出されたトポロジの変更だけが送信されます。オペレーション管理のインスタンスにおけるトポロジの変更は送信されず，動的トポロジ同期によって HPOM システムに同期されません。

データベース同期の詳細については，HP Business Service Management RTSM のマニュアルを参照してください。

いったん設定すると，動的なトポロジ同期は，バックグラウンドで継続的に動作します。また，バックグラウンドでの継続的動作は，直前の同期からすべての要素にタッチすることにより RTSM 内でエージングを避けるプロセスでもあります。このことは，タッチ・モードで基本トポロジ同期を動作させることに相当します。

RTSM のエージングの詳細については，『Modeling Guide』を参照してください。

マッピング・ルールに基づき，HPOM トポロジ・データをソースとして使用して RTSM に CI を作成するために，動的トポロジ同期では次の操作が実行されます。

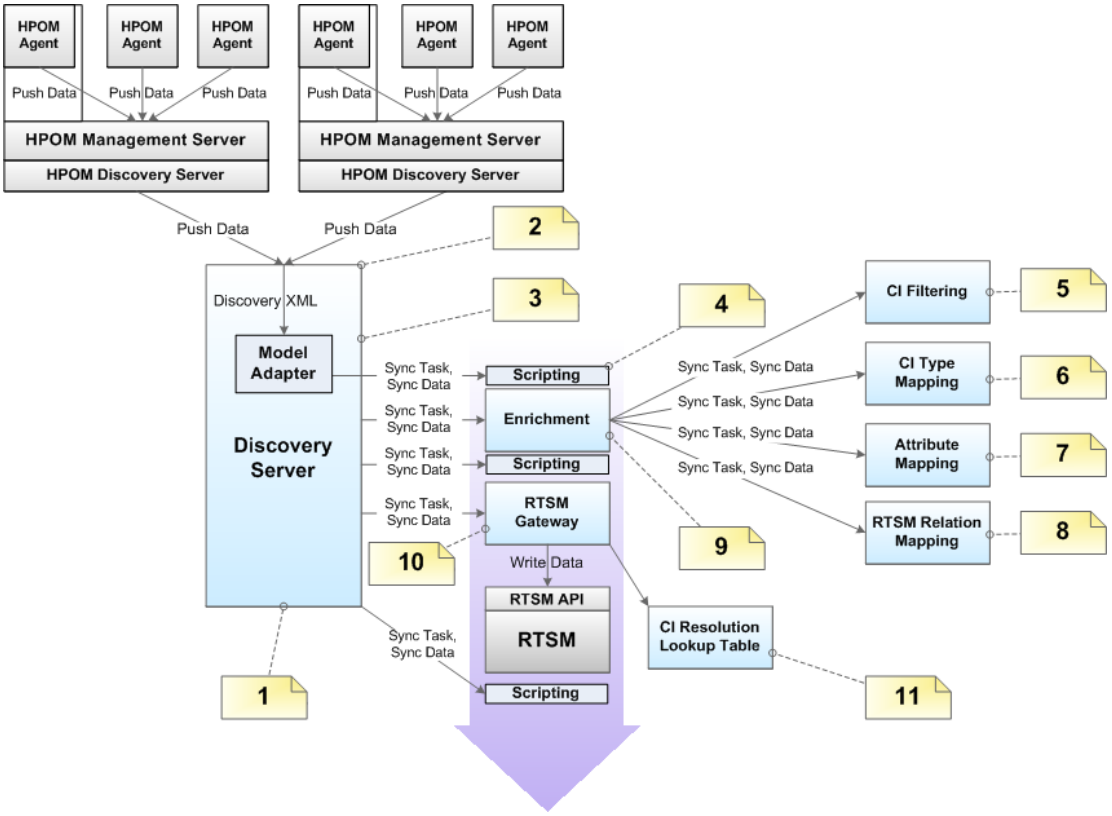
- 信頼できる証明書の交換を必要とする HTTPS ベースの通信を使用して，検出されたトポロジ・データを設定されたソース転送サーバから設定されたターゲット・サーバに転送します。
- 検出されたトポロジ・データを非同期に受信します。
- トポロジ・データをオペレーション管理と互換性のある形式に変換します。
- トポロジ・データを RTSM にアップロードします。
- データをオンデマンドで RTSM にアップロードします。

- デルタ検出を実行し，HPOM で削除された要素を RTSM から削除します。
- CI 解決のためのマッピング・テーブルを提供します。

注: トポロジ同期プロセスでは新しい CI タイプを作成できないため，オペレーション管理と HPOM のトポロジを同期させるときには，特定の CI タイプの定義が特に重要となります。トポロジ同期プロセスで RTSM に存在しない CI タイプにマップしようとすると，そのプロセスは停止します。

動的トポロジ同期アーキテクチャ

次の図に動的トポロジ同期アーキテクチャの概要を示します。



図内の番号は次の表に示す注記に対応しています。

参照情報	参照	注記
1	ディスカバリ・サーバ	• 同期プロセスを制御し，あるコンポーネントから別のコンポーネントにデータを渡す。 • ゲートウェイ・サーバ上の `hpsbm_wde` プロセスで実行される。

参照情報	参照	注記
2	ディスカバリ・サーバでトポロジ・データを受信	HPOM サーバまたは別の OMi サーバから送信されたディスカバリ XML 形式のトポロジ・データを受信します。
3	モデル・アダプタ	ディスカバリ XML データを同期データ構造に変換する。
4	スクリプティング	同期データを操作する Groovy スクリプトを実行する。
5	CI フィルタ	マッピング・エンジンを使用して，同期モデルに属している CI とそうでない CI を指定する。
6	CI タイプ・マッピング	RTSM(実行時サービス・モデル) タイプのサービス・タイプ定義 (STD) をマップする。
7	属性マッピング	タイプが指定されていない HPOM 関係を RTSM の関係にマップする。
8	RTSM 関係マッピング	• RTSM の関係を作成する。 • モデルに従って RTSM 内に CI を作成できるようにルート・コンテナを CI にマップする。
9	エンリッチメント	エンリッチメント・プロセス全体を制御し，変更された同期データを返す。
10	RTSM ゲートウェイ	同期データを RTSM に書き込む。
11	CI 解決マッピング・テーブル	同期された各 CI について，HPOM サービス ID を RTSM CI ID にマップする CI 解決コンポーネントのマッピング・テーブルを更新する。

動的トポロジ同期の実行

HPOM 管理サーバから オペレーション管理 へのトポロジ(ノードおよびサービス) データの動的転送を設定する前に，オペレーション管理 で次の設定手順を実行してください。

1. HPOM 管理サーバを オペレーション管理 内の接続サーバとして追加します。詳細については，オペレーション管理 のオンライン・ヘルプを参照してください。
2. データ処理サーバと HPOM 管理サーバとの間に信頼関係を確立します。詳細については，オペレーション管理 のオンライン・ヘルプを参照してください。
3. *任意:* `opr-sdtool.bat` コマンドライン・ツールを使用して，ファイル・システムからデータベースに新規または変更済み同期パッケージをアップロードします。詳細については，183ページ「同期パッケージのアップロードとダウンロード」を参照してください。
4. HPOM 管理サーバで，次の項の説明に従ってトポロジ・データの転送を設定します。

次の項では，トポロジ同期の設定を行う方法について説明します。

HPOM for Windows システムで動的トポロジ同期を設定する方法

本項では，HPOM for Windows 管理サーバで動的トポロジ同期を設定する方法について説明します。詳細については，HPOM for Windows のドキュメントを参照してください。

オペレーション管理 にトポロジ・データを転送するには，トポロジ情報を受信するHPOM for Windows 管理サーバで次の手順を実行します。

1. 前提条件 :HPOM for Windows 管理サーバ用の最低限のパッチ・レベルがインストールされていることを確認してください。
   - バージョン 8.16 : Patch OMW_00121 またはそれに代わるパッチ。
   - バージョン 9.00 : Patch OMW_00122 またはそれに代わるパッチ。
2. 前提条件 :信頼された証明書を複数のサーバに設定します。

   複数サーバがある環境では，各サーバにほかのサーバが発行した証明書を信頼するよう設定する必要があります。
3. コンソール・ツリーで，[**Operations Manager**]を右クリックし，[**Configure→Server...**]をクリックします。[Server Configuration]ダイアログ・ボックスが開きます。
4. [**名前空間**]をクリックし，[**検出サーバ**]をクリックします。値の一覧が表示されます。
5. サーバのホスト名を[**検出データの転送先となるターゲット サーバのリスト**]に追加します。ターゲット・サーバが複数ある場合，次のようにホスト名をセミコロンで区切ります。

   ```
   server1.example.com;server2.example.com
   ```

   ターゲット・サーバが 383 以外のポートを使用する場合，次のようにポート番号をホスト名に追加します。

   ```
   server1.example.com:65530;server2.example.com:65531
   ```
6. [**検出 WMI リスナの有効化**]の値が true であることを確認します。これはデフォルトの値です。
7. [**OK**]をクリックして設定を保存し，[Server Configuration]ダイアログ・ボックスを閉じます。
8. 変更を反映させるために `OvAutoDiscovery Server` プロセスを再起動します。
9. トポロジ・データの初期同期を開始します。
   a. コンソール・ツリーで[**ツール**]>[**HP Operations Manager ツール**]を選択します。
   b. [**Synchronize Topology**]を右クリックし，[**すべてのタスク**]>[**ツールを起動...**]を選択します。

      ツール `startInitialSync.bat` が起動し，すべてのトポロジ・データを，設定済み対象管理サーバに送信し始めます。

HPOM for Windows システムで計画的同期から移行する方法

本項では，HPOM for Windows 管理サーバで計画的同期から移行する方法について説明します。詳細については，HPOM for Windows のドキュメントを参照してください。

計画的同期から移行するには，トポロジ情報受信時の送信元とする HPOM for Windows 管理サーバで次の手順を実行します。

1. 前提条件 :HPOM for Windows 管理サーバ用の最低限のパッチ・レベルがインストールされていることを確認してください。
   - バージョン 8.16 : Patch OMW_00121 またはそれに代わるパッチ。
   - バージョン 9.00 : Patch OMW_00122 またはそれに代わるパッチ。
2. 次のコマンドを使用して HPOM 管理サーバでエージェント・リポジトリ・キャッシュをクリアします。

   ```
   %OvBinDir%\ovagtrep -clearall
   ```

3. 次のコマンドを入力して，HPOM 管理サーバ・ノードからサービス自動検出ポリシーを削除します。

   ```
   %OvBinDir%\ovpolicy -remove DiscoverOM

   %OvBinDir%\ovpolicy -remove DiscoverOMTypes
   ```

4. HPOM 管理サーバでポリシー・インベントリを同期します。
   a. コンソール・ツリーで管理サーバを右クリックします。
   b. [**すべてのタスク**]>[**Synchronize inventory**]>[**ポリシー**]を選択します。

      管理サーバは，ローカル・エージェントからインベントリを取得するためのデプロイメント・ジョブを作成します。
5. リスナ・プロセスが実行されていることを確認します。
   a. コンソール・ツリーで，[**Operations Manager**]を右クリックし，[**Configure Server**]を選択します。

      [Server Configuration]ダイアログ・ボックスが開きます。
   b. [**名前空間**]をクリックし，[**検出サーバ**]を選択します。

      値の一覧が表示されます。
   c. [**検出 WMI リスナの有効化**]の値を true に設定します。これはデフォルトの値です。
   d. [**OK**]をクリックして設定を保存し，[Server Configuration]ダイアログ・ボックスを閉じます。
   e. 変更を反映させるために，次のコマンドを使用して OvAutoDiscovery サーバ・プロセスを再起動します。

      ```
      net stop "OvAutoDiscovery Server"

      net start "OvAutoDiscovery Server"
      ```

6. トポロジ・データの初期同期を開始します。
   a. コンソール・ツリーで[**ツール**]>[**HP Operations Manager ツール**]を選択します。
   b. [**Synchronize Topology**]を右クリックし，[**すべてのタスク**]>[**ツールを起動...**]を選択します。

ツール startInitialSync.bat が起動し，すべてのトポロジ・データを，設定済み対象サーバに送信し始めます。

HPOM for UNIX または HPOM for Linux システムで動的トポロジ同期を設定する方法

本項では，HPOM for UNIX または HPOM for Linux 管理サーバで動的トポロジ同期を設定する方法について説明します。詳細については，HPOM for UNIX または Linux のドキュメントを参照してください。

オペレーション管理 にトポロジ・データを転送するには，トポロジ情報受信時の送信元とする HPOM for UNIX または Linux 管理サーバで次の手順を実行します。

1. 前提条件 : HPOM 9.10 for UNIX または Linux 管理サーバ用の最低限のパッチ・レベルがインストールされていることを確認してください。
   - HP-UX :Patch PHSS_42736 またはそれに代わるパッチ。
   - Linux: Patch OML_00050 またはそれに代わるパッチ。
   - Solaris :Patch ITOSOL_00772 またはそれに代わるパッチ。
2. 前提条件 : HPOM for UNIX または Linux 管理サーバの HP Operations Agent のバージョンが 8.60.500 以上であることを確認します(これより古いエージェントには，エージェント・ホットフィックス QCCR1A100254 が必要であり，インスタンス・データ全体が送信されるように agtrep を設定する必要があります) 。
3. 前提条件 :信頼された証明書を複数のサーバに設定します。

   複数サーバがある環境では，各サーバにほかのサーバが発行した証明書を信頼するよう設定する必要があります。
4. 次のコマンドを入力してトポロジ同期を有効にします。

   ```
   /opt/OV/contrib/OpC/enableToposync.sh -online -target <comma_
   separated_server_list>
   ```

   <サーバのコンマ区切りリスト> を対象管理サーバの完全修飾ドメイン名に置き換えます。複数の対象管理サーバがある場合，各サーバ名をコンマ(,) で区切ります。サーバ・リストにはスペースを挿入しないでください。

   このコマンドによって，サービス検出サーバが再起動されます。ソース管理サーバが，ただちにトポロジ・データ変更の送信を開始します。
5. 次のコマンドを入力して，トポロジ・データの初期同期を開始します。

   ```
   /opt/OV/bin/OpC/startInitialSync.sh
   ```

HPOM for UNIX または HPOM for Linux システムで計画的同期から移行する方法

本項では，HPOM for UNIX または Linux 管理サーバで計画的同期から移行する方法について説明します。詳細については，HPOM for UNIX または Linux のドキュメントを参照してください。

計画的同期から移行するには，トポロジ情報受信時の送信元とする HPOM for UNIX または Linux 管理サーバで次の手順を実行します。

1. 前提条件 : HPOM for Windows 管理サーバ用の最低限のパッチ・レベルがインストールされていることを確認してください。

- HP-UX :Patch PHSS_42736 またはそれに代わるパッチ。
- Linux: Patch OML_00050 またはそれに代わるパッチ。
- Solaris :Patch ITOSOL_00772 またはそれに代わるパッチ。

2. 次のコマンドを使用して，管理サーバでエージェント・リポジトリ・キャッシュをクリアします。

   ```
   /opt/OV/bin/ovagtrep -clearall
   ```

3. 次のコマンドを使用して，管理サーバ・ノードからサービス自動検出ポリシーを削除します。

   ```
   /opt/OV/bin/ovpolicy -remove DiscoverOM

   /opt/OV/bin/ovpolicy -remove DiscoverOMTypes
   ```

4. 次のコマンドを使用して，管理サーバ・ノードからサービス自動検出ポリシーの割り当てを解除します。

   ```
   /opt/OV/bin/OpC/utils/opcnode -deassign_pol node_name=<management_
   server> net_type=NETWORK_IP pol_name=DiscoverOMTypes
   pol_type=svcdisc

   /opt/OV/bin/OpC/utils/opcnode -deassign_pol node_name=<management_
   server> net_type=NETWORK_IP pol_name=DiscoverOM
   pol_type=svcdisc

   /opt/OV/bin/OpC/opcragt -dist <management_server>
   ```

   <management_server> を管理サーバの名前に置き換えます。

5. 次のコマンドを入力してトポロジ同期を有効にします。

   ```
   /opt/OV/contrib/OpC/enableToposync.sh -online
   ```

   このコマンドによって，サービス検出サーバが再起動されます。ソース管理サーバが，ただちにトポロジ・データ変更の送信を開始します。

6. 次のコマンドを入力して，トポロジ・データの初期同期を開始します。

   ```
   /opt/OV/bin/OpC/startInitialSync.sh
   ```

基本トポロジ同期

注: 基本トポロジ同期は，HPOM の動的トポロジ同期をサポートしないバージョンでのみ使用します。HPOM のサポートされているバージョンの詳細については，『HP Business Service Management リリース・ノート』を参照してください。

基本トポロジ同期は，BSM データ処理サーバ上でオンデマンドで実行されるコマンドライン・ツールです。

マッピング・ルールに基づき，HPOM ノード，ノード・グループ，および HPOM サービス・モデルをソースとして使用して RTSM(実行時サービス・モデル) に CI を作成する場合，基本トポロジ同期は次の操作を実行します。

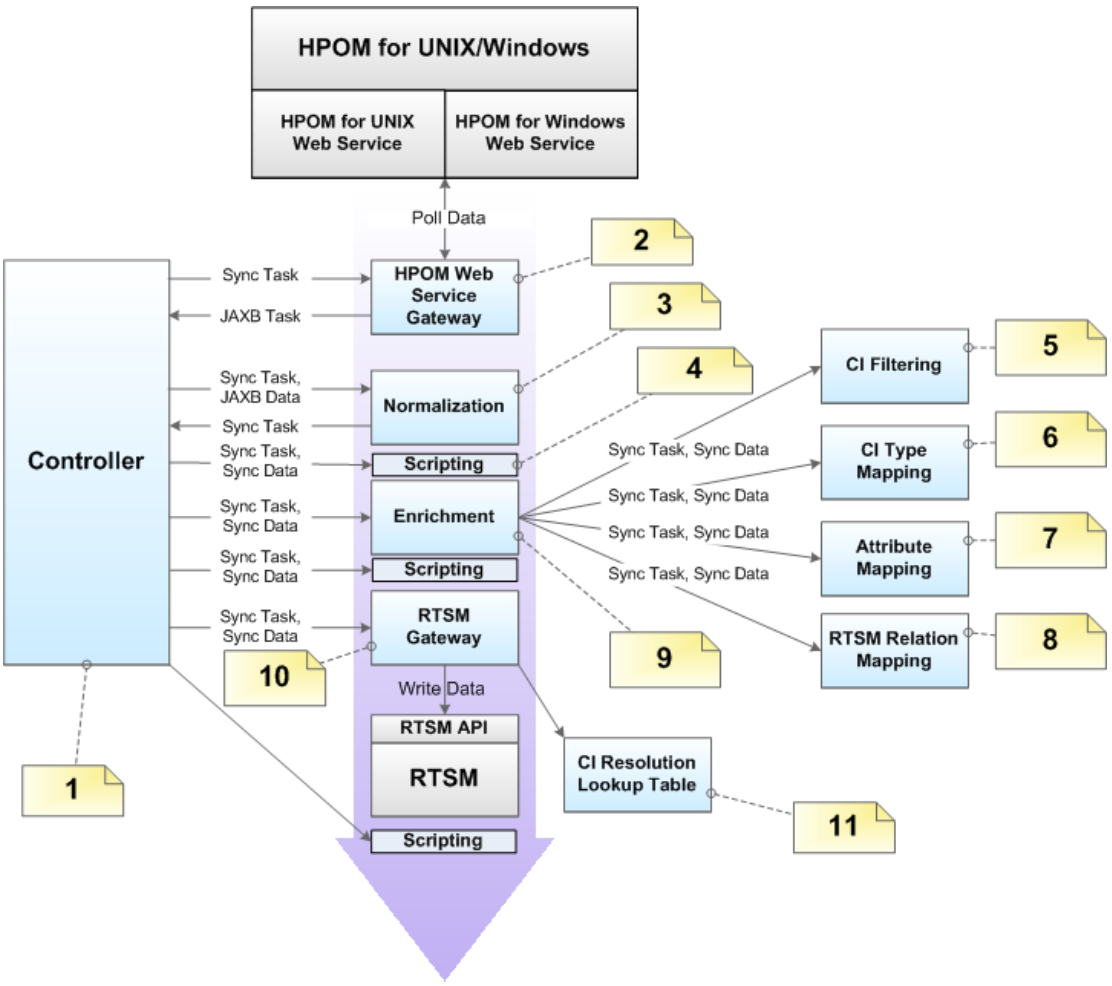
- HPOM Web サービスを使用して，サービス・モデル階層の全体 (インフラストラクチャベースのサービス管理データ) を HPOM からオペレーション管理に転送します。

- サービス管理データをオペレーション管理と互換性のある形式に変換します。
- サービス管理データを RTSM(実行時サービス・モデル)にアップロードします。
- データをオンデマンドで RTSM にアップロードします。
- デルタ検出を実行し，HPOM で削除された要素を RTSM から削除します。
- CI 解決のためのマッピング・テーブルを提供します。

注: トポロジ同期プロセスでは新しい CI タイプを作成できないため，オペレーション管理と HPOM のトポロジを同期させるときには，特定の CI タイプの定義が特に重要となります。トポロジ同期プロセスで RTSM に存在しない CI タイプにマップしようとすると，そのプロセスは停止します。

基本トポロジ同期アーキテクチャ

次の図にトポロジ同期アーキテクチャの概要を示します。



図内の番号は次の表に示す注記に対応しています。

参照情報	参照	注記
1	コントローラ	• 同期プロセスを制御し，あるコンポーネントから別のコンポーネントにデータを渡す。 • コマンド・ラインから同期を開始するための実行可能クラスを提供する。
2	HPOM Web サービス・ゲートウェイ	要求を HPOM Web サービスに送信(プル)し，JAXB オブジェクトを返す。
3	正規化	JAXB の結果を内部同期データ構造に変換する。
4	スクリプティング	同期データを操作する Groovy スクリプトを実行する。
5	CI フィルタ	マッピング・エンジンを使用して，同期モデルに属している CI とそうでない CI を指定する。
6	CI タイプ・マッピング	RTSM(実行時サービス・モデル)タイプのサービス・タイプ定義(STD)をマップする。
7	属性マッピング	タイプが指定されていない HPOM 関係をタイプが指定された RTSM 関係にマップする。
8	RTSM 関係マッピング	• RTSM の関係を作成する。 • モデルに従って RTSM 内に CI を作成できるようにルート・コンテナを CI にマップする。
9	エンリッチメント	エンリッチメント・プロセス全体を制御し，変更された同期データを返す。
10	RTSM ゲートウェイ	同期データを RTSM に書き込む。
11	CI 解決ルックアップ・テーブル	同期された各 CI について，HPOM サービス ID を RTSM CI ID にマップする CI 解決コンポーネントのマッピング・テーブルを更新する。

基本トポロジ同期の実行

基本トポロジ同期を開始するには，BSM データ処理サーバ上で `opr-startTopologySync` コマンドライン・ツールをオンデマンドで実行します。

`opr-startTopologySync` ツールのファイル拡張子は，Windows システムの場合は `.bat`，Linux システムの場合は `.sh` です。

注: Windows スケジューラでトポロジ同期タスクを設定する場合は，次のバイナリ・ファイルを使用します。

```
HPBSM ルート・ディレクトリ>/opr/bin/startTopologySync.bat
```

opr-startTopologySync.bat ツールは，次の2つのモードで実行できます。

- 通常モード
- タッチ・モード

通常モード

通常モードでは，サービス・モデル全体がロードされ，設定されたすべてのデータが1つのHPOMサーバ(インフラストラクチャ設定で設定されている)からRTSM(実行時サービス・モデル)に同期されます。通常モードでは，デルタ検出も実行され，HPOMで削除済みの要素がRTSMから削除されます。

opr-startTopologySync.bat ツールを通常モードで実行するには，次のコマンドを入力します。

Windows: <HPBSM ルート・ディレクトリ>/bin/opr-startTopologySync.bat

Linux: <HPBSM ルート・ディレクトリ>/bin/opr-startTopologySync.sh

タッチ・モード

タッチ・モードでは，前回の同期によるすべての要素をタッチすることにより，RTSM内のエージングを防止します。タッチ・モードでは，RTSMに新しいCIは作成されず，CIの削除も実行されません。

opr-startTopologySync.bat ツールをタッチ・モードで実行するには，次のコマンドを入力します。

Windows: <HPBSM ルート・ディレクトリ>/bin/opr-startTopologySync.bat –touch

Linux: <HPBSM ルート・ディレクトリ>/bin/opr-startTopologySync.sh –touch

RTSMのエージングの詳細については，HP Business Service Management『Modeling Guide』を参照してください。

スキップ・サービス・オプション

コマンドライン・オプション -skipservices を指定すると，HPOM Webサービスからのサービスのロードがスキップされます。このオプションは，HPOMサービス・モデルの規模が大きいために，ロードが失敗する可能性がある場合に役立ちます。

HPOMサービスのロードをスキップするオプションを指定して opr-startTopologySync.bat ツールを通常モードで実行する場合は，次のコマンドを入力します。

Windows: <HPBSM ルート・ディレクトリ>/bin/opr-startTopologySync.bat –skipservices

Linux: <HPBSM ルート・ディレクトリ>/bin/opr-startTopologySync.sh –skipservices

基本トポロジ同期と動的トポロジ同期の比較

次の表は，基本トポロジ同期と動的トポロジ同期の違いをまとめたものです。

基本	動的
コマンドラインから実行する。	手動での初期同期の後に自動的に実行される。結果は常に手動での操作なしに使用できる。
1つのHPOMサーバからトポロジ・データを受け取る。	複数のHPOMサーバからトポロジ・データを受け取り，接続されているほかのサーバに転送できる。
スケジュールベース: 設定された間隔でのみ実行される。	非同期: トポロジの変更が検出された場合，非同期に実行される。
次のスケジュールされた実行まで更新されない。	HPOMで変更が発生した後すぐに更新がRTSM(実行時サービス・モデル) に書き込まれる。
HPOMサービス・モデルから常に完全なトポロジ・データを取得する。そのため，リソースの消費が比較的高い。	トポロジ・データの変更(デルタ) のみを受け取る。RTSMでは変更されたデータを更新するだけでよいため，リソースの消費は基本トポロジ同期よりはるかに低い。
Webサービスにアクセスするために追加のポートが必要。	単一ポートHTTPS通信。

同期パッケージとマッピング

トポロジ同期では，同期パッケージを使用してRTSM(実行時サービス・モデル) 内にCIを作成します。トポロジ同期パッケージには，HPOM側の1つ以上のサービス，ノード，またはノード・グループとRTSM側の1つ以上のCIとのマッピングが含まれています。

トポロジ同期パッケージは，XML設定ファイルで構成されます(188ページ「マッピング・ファイル」を参照)。これらのファイルによって，HPOMサービス，ノード，およびノード・グループがRTSM内のCIに変換され，それらのCIがHPOM内の指定されたサービス，ノード，およびノード・グループのデータと同期されます。

トポロジ同期パッケージには，次の2つのタイプがあります。

- インストール・パッケージに付属の標準パッケージ。

  詳細については，186ページ「標準設定の同期パッケージ」を参照してください。

- HPOM SPIのサブセットおよび使用可能なコンテンツ・パックに対応する追加の付属パッケージ。

  詳細については，186ページ「追加の標準設定の同期パッケージ」を参照してください。

トポロジ同期パッケージを独自に作成することもできます。トポロジ同期の設定例と同期パッケージの作成例については，188ページ「トポロジ同期の設定: ACME環境の例」を参照してください。

標準設定のインストールでは，同期パッケージがファイル・システムからRTSMへ自動的にロードされます。新しい同期パッケージを作成したり既存の同期パッケージに変更を加えたりするときには，コマンドライン・ツールを使用して新規パッケージまたは変更済みのパッケージをRTSMにアップロードする必要があります。詳細については，183ページ「同期パッケージのアップロードとダウンロード」を参照してください。

同期パッケージとマッピングの詳細については，185ページ「同期パッケージ」を参照してください。

スクリプティングとトポロジ・データ

スクリプティングによって，同期プロセスにおいて，マッピングの前やHPOMからRTSM(実行時サービス・モデル)へのトポロジ・データのアップロードの前後に追加の処理やカスタマイズを実行できます。

Groovy スクリプトでは，同期プロセスの実行中に同期データを操作できます。Groovy スクリプトは，トポロジ同期パッケージに含めることができます。1つのHPOM サービスから2つのCIを作成する場合などには，XML マッピング・ファイルだけでは作成できないため，スクリプティングが必要となります。

詳細については，193ページ「スクリプティング」を参照してください。

マッピング・テーブルを使用したCI解決

トポロジ同期によって，HPOM から同期されたすべてのCIのマッピング・テーブルが作成されます。このマッピング・テーブルは，CI解決のショートカットとして利用できます。このテーブルでHPOM サービスのサービスIDが検索され，RTSM(実行時サービス・モデル)内のCIにマップされます。マッピング・テーブルの使用を有効にすると，マッピング・テーブルが分析された後でCI解決が使用されます。マッピング・テーブルで一致しない場合は，CI解決がスマート・メッセージ・マッピングなどのマッピング・プロセスを引き継ぎます。

マッピング・テーブルの使用は標準設定で有効になっています(CIリゾルバ設定の[**トポロジ同期ショートカットを使用**]設定が[**true**]に設定されています)。

マッピング・テーブルの使用を有効にするのは，通常，HPOM サービス・ツリー内のサービスとRTSM 内のそのサービスのCIとの間に直接の一対一の関係がある場合です。

サービス・ツリーの構造とRTSM の構造がまったく異なり，サービスとRTSM 内のそのサービスのCIとの間に一対一の関係がない場合など，マッピング・テーブル・ショートカットを使用しない方がよい場合もあります。RTSM 内にサービス障害の原因に関する情報を示すCIが数多くある場合は，CI解決がサービス・オブジェクトの最も適切なCIを見つける最も迅速で信頼性の高い方法です。

マッピング・テーブルを使用しない場合は，BSM インフラストラクチャ設定マネージャの[CIリゾルバ設定]で無効にできます。

[インフラストラクチャ設定] > [アプリケーション] > [オペレーション管理] > [オペレーション管理 - CIリゾルバ設定] > [トポロジ同期ショートカットを使用]

トポロジ同期ファイルの場所

初期の製品インストールでは，トポロジ同期ファイルがBSM データ処理サーバ上のローカル・ファイル・システムの本項に示す場所にコピーされます。

基本トポロジ同期

基本トポロジ同期のトポロジ同期ファイルは次の場所にあります。

バイナリ:

- <HPBSM ルート・ディレクトリ>/opr/lib/opr-ts-*.jar
- **Linux:**
  - <HPBSM ルート・ディレクトリ>/bin/opr-startTopologySync.sh
  - <HPBSM ルート・ディレクトリ>/bin/opr-sdtool.sh

    <HPBSM ルート・ディレクトリ>/bin/opr-startTopologySync.sh
- **Windows:**
  - <HPBSM ルート・ディレクトリ>\bin\opr-sdtool.bat
  - <HPBSM ルート・ディレクトリ>\bin\opr-startTopologySync.bat

ログ・ファイル:

<HPBSM ルート・ディレクトリ>/log/opr-topologysync

ログ・ファイル設定ファイル:

<HPBSM ルート・ディレクトリ>/conf/core/Tools/log4j/opr-topologysync/opr-topologysync.properties

トポロジ同期パッケージ:

<HPBSM ルート・ディレクトリ>/conf/opr/topology-sync/sync-packages

スキーマ・ファイル:

<HPBSM ルート・ディレクトリ>/conf/opr/topology-sync/schemas

動的トポロジ同期

動的トポロジ同期のトポロジ同期ファイルは次の場所にあります。

バイナリ:

Windows: <HPBSM ルート・ディレクトリ>/bin/opr-sdtool.bat

Linux: <HPBSM ルート・ディレクトリ>/bin/opr-sdtool.sh

<HPBSM ルート・ディレクトリ>/opr/lib/opr-ts-*.jar

<HPBSM ルート・ディレクトリ>/opr/lib/OvSvcDiscServer.jar

ログ・ファイル:

<HPBSM ルート・ディレクトリ>/log/wde/opr-svcdiscserver.log

<OvDataDir>/log/OvSvcDiscServer.log

ログ・ファイル設定ファイル:

<HPBSM ルート・ディレクトリ>/conf/core/Tools/log4j/wde/opr-svcdiscserver.properties

トポロジ同期パッケージ:

<HPBSM ルート・ディレクトリ>/conf/opr/topology-sync/sync-packages

スキーマ・ファイル:

<HPBSM ルート・ディレクトリ>/conf/opr/topology-sync/schemas

トポロジ同期設定

オペレーション管理 の同期が成功するためには，HPOM において次の設定が正しく構成されている必要があります。

- **HPOM トポロジ同期接続設定**:(基本トポロジ同期のみ)

  基本トポロジ同期では，同期中に HP Operations Manager Web サービス(WS) からトポロジ・データを読み取る必要があります。これを実現する設定は，［HPOM 接続設定］で行います。

  ［HPOM 接続設定］には次のようにアクセスできます。

  ［インフラストラクチャ設定］> ［アプリケーション］> ［オペレーション管理オペレーション管理 - HPOM トポロジ同期接続設定］

  詳細については，オペレーション管理 のオンライン・ヘルプを参照してください。

- **HPOM トポロジ同期設定**

  ［HPOM トポロジ同期設定］では，基本トポロジ同期と動的トポロジ同期の両方について，次の設定ができます。

  - ダンプ・データの有効化または無効化
  - Groovy スクリプトの使用の有効化または無効化
  - 使用するトポロジ同期パッケージの指定

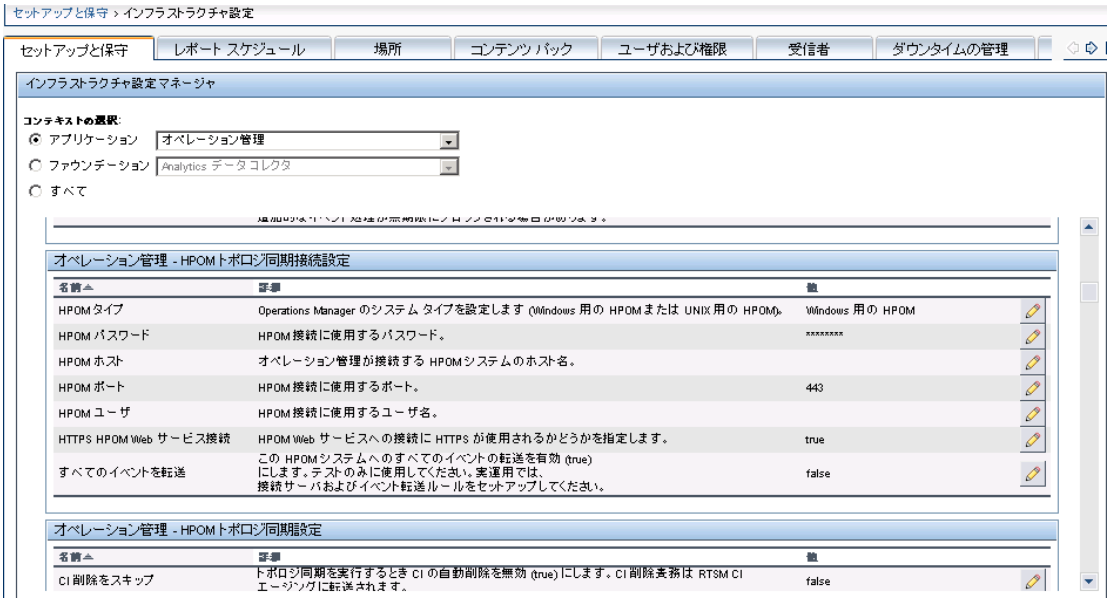
  基本トポロジ同期のみ:

  IP アドレスに関する情報がない HPOM ノードの同期中の IP アドレス解決を有効にすることもできます。

  > **注:** これらの設定は，オペレーション管理 の正しい設定とオペレーション管理 および HPOM の監視対象となる環境におけるオブジェクト・トポロジの同期の成功にとって不可欠のものです。

  HPOM トポロジ同期設定には次のようにアクセスできます。

  ［インフラストラクチャ設定］> ［アプリケーション］> ［オペレーション管理オペレーション管理 - HPOM トポロジ同期設定］



同期パッケージのアップロードとダウンロード

基本トポロジ同期と動的トポロジ同期のどちらについても，コマンドライン・ツール `opr-sdtool` を使用して，新規または変更済みの同期パッケージをファイル・システムからデータベースにアップロードしたり，データベースからファイルにダウンロードしたりできます。

同期パッケージのアップロード

トポロジ同期は，データベースにロードされた同期パッケージを常に使用します。そのため，同期パッケージ・ファイルに変更を加えた場合は，その同期パッケージを再度データベースにアップロードする必要があります。

新規または変更済みの同期パッケージをデータベースにアップロードするには，次のコマンドを実行します。

Windows: `opr-sdtool.bat -uploadpackage <path of synchronization package>`

Linux: `opr-sdtool.sh -uploadpackage <path of synchronization package>`

同期パッケージのダウンロード

ダウンロード・オプションも使用できるため，BSM を分散環境にインストールしたときには，各 BSM サーバで同期パッケージの最新バージョンを編集できます。

同期パッケージをデータベースからダウンロードするには，次のコマンドを実行します。

Windows: `opr-sdtool.bat -downloadpackage [<syncPackageName>] [-path [<downloadPath>]]`

Linux: `opr-sdtool.sh -downloadpackage [<syncPackageName>] [-path [<downloadPath>]]`

同期パッケージの名前を指定しなかった場合，現在データベースにあるすべてのパッケージがダウンロードされます。

ダウンロード・パスを指定しなかった場合，パッケージは `<HPBSM ルート・ディレクトリ>/conf/opr/topology-sync/sync-packages` にダウンロードされます。

HPOM SPI サービス・タイプ定義のデータベースへのアップロード

HPOM スマート・プラグイン(SPI) サービス・タイプ定義は，エージェントから受信したサービスの処理に使用されます。

基本トポロジ同期と動的トポロジ同期のいずれにもコマンド・ライン・ツール `opr-sdtool` が用意されており，これに `-uploadstd` コマンド・ライン・オプションを指定して，新規または変更済みのサービス・タイプ定義を HPOM スマート・プラグイン(SPI) からデータベースにアップロードできます。

新規または変更済みのサービス・タイプ定義を HPOM SPI からデータベースにアップロードするには，次のコマンドを実行します。

Windows: `opr-sdtool.bat –uploadstd <path of MofFile>`

Linux: `opr-sdtool.sh –uploadstd <path of MofFile>`

同期パッケージ

本項では，HPOMトポロジ・データをRTSM(実行時サービス・モデル)内のCIにマップするためのルールが含まれるトポロジ同期パッケージについて説明します。

本章の内容

同期パッケージの概要

トポロジ同期パッケージには，HPOM側の1つ以上のサービス・モデル，ノード，またはノード・グループとRTSM(実行時サービス・モデル)側の1つ以上のCIとのマッピングが含まれています。

トポロジ同期パッケージは，トポロジ同期の実行中に適用されるマッピング・ルール(コンテキスト，CIタイプ，属性など)が定義されたXML設定ファイルのセットで構成されます。設定ファイルは次の目的に使用されます。

- トポロジ・データ(HPOMサービス，ノード，ノード・グループなど)をRTSM内のCIに変換する。
- RTSM内のCIをHPOMのトポロジ・データと同期する。

トポロジ同期パッケージには，同期パッケージを定義するパッケージ記述子ファイル(`package.xml`)が含まれている必要があります(187ページ「パッケージ記述子ファイル: package.xml」を参照)。

同期パッケージには，次のマッピング・ファイルを含めることができます。

- `contextmapping.xml`
- `typemapping.xml`
- `attributemapping.xml`
- `relationmapping.xml`

XML設定ファイルの詳細については，188ページ「マッピング・ファイル」を参照してください。

マッピングの基本的な情報については，207ページ「マッピング・エンジンと構文」を参照してください。

トポロジ同期パッケージには，同期プロセスの実行中に同期データを操作したり，監査などの目的で同期後の操作を実行したりするために，Groovyスクリプトを含めることもできます。トポロジ同期パッケージには，次のGroovyスクリプトを含めることができます。

- `preEnrichment.groovy`
- `preUpload.groovy`
- `postUpload.groovy`

Groovy スクリプトの詳細については，193ページ「Groovy スクリプト」を参照してください。

標準設定の同期パッケージ

オペレーション管理 とHPOM とのトポロジの同期において更新するコンテンツを指定できます。

3 つの標準設定のトポロジ同期パッケージが用意されています。

- `default`

  ノードの基本的なタイプ・マッピングとノードおよびノード・グループの基本的な属性マッピングが含まれています。

  RTSM(実行時サービス・モデル) に CI を作成しません。

  有効に設定されたパッケージのリストから削除することはできません。

- `operations-agent`

  ホスト CI 自体を作成するのに加え，エージェントを持つ HPOM 管理対象ノードごとに `hp_operations_agent` タイプの CI インスタンスを作成し，それをホスト CI に関連付けます。さらに，`omserver` タイプの CI を作成し，それをホストとすべての `hp_operations_agent` CI に関連付けます。

- `nodegroups`

  ホスト CI 自体を作成するのに加え，HPOM ノード・グループを RTSM(実行時サービス・モデル) CI タイプ `ci_group` にマップし，CI タイプ `ci_group` のインスタンスを作成して，グループ内のノードの関係を作成します。さらに，`ipaddress` タイプおよび `interface` タイプの CI を作成し，それらをホストに関連付けます。

[HPOM トポロジ同期設定] の[**トポロジ同期のパッケージ**]で，トポロジ同期プロセスの実行中にコンテンツを更新するパッケージのリストを作成できます。

[インフラストラクチャ設定] > [アプリケーション] > [オペレーション管理] > [オペレーション管理 - HPOM トポロジ同期設定] > [トポロジ同期のパッケージ]

このリストのエントリは，次の例に示すようにセミコロン(;) で区切る必要があります。

```
default;nodegroups;operations-agent
```

標準設定では，パッケージは次のディレクトリにあります。

```
<HPBSM ルート・ディレクトリ>/conf/opr/topology-sync/sync-packages
```

追加のトポロジ同期パッケージがコンテンツ・パックに用意されています。

追加の標準設定の同期パッケージ

追加のトポロジ同期パッケージがコンテンツ・パックに標準で含まれています。コンテンツ・パックの内容は次のとおりです。

- ActiveDirectory
- Exchange
- MS SQL Server
- Oracle
- J2EE(WebSphere および WebLogic コンテンツを含む)
- インフラストラクチャ(UNIX および Windows オペレーティング・システム，仮想化システム，およびクラスタ・システムを含む)

これらの追加のトポロジ同期パッケージは，標準設定では有効になっていません。有効にするには，次の手順を実行します。

1. 使用するコンテンツ・パックをロードします。
2. インフラストラクチャ設定で同期パッケージを手動で有効にします。

   [インフラストラクチャ設定] > [アプリケーション] > [オペレーション管理] > [オペレーション管理 - HPOM トポロジ同期設定] > [トポロジ同期のパッケージ]

トポロジ同期パッケージは次のディレクトリに書き込まれます。

```
<HPBSM ルート・ディレクトリ>/conf/opr/topology-sync/sync-packages
```

たとえば，Oracle コンテンツ・パックでは，`HPOprOra` というパッケージ名 (およびディレクトリ名) が使用されます。トポロジ同期でこのマッピング・ルールを実行する場合は，この名前をリストに追加します。Oracle パッケージを186ページ「標準設定の同期パッケージ」の例に示す標準パッケージのリストに追加すると，リストは次のようになります。

```
default;nodegroups;operations-agent;HPOprOra
```

注: カスタム・パッケージを追加する場合は，パッケージ名がパッケージのあるディレクトリの名前と同じであることに注意してください。同期パッケージを削除すると，その同期パッケージを以前実行したときに RTSM(実行時サービス・モデル) に追加され，ほかの CI との間で調整されていない CI も削除されるため，注意が必要です。

パッケージ記述子ファイル : package.xml

トポロジ同期パッケージには，パッケージ記述子ファイル(`package.xml`) を含める必要があります。`package.xml` ファイルにはトポロジ同期パッケージを定義し，次の情報を指定します。

- `<Name>` パッケージの名前は，同期パッケージを置くサブディレクトリの名前と同じである必要があります。

  ```
  <HPBSM ルート・ディレクトリ>/con/opr/topology-sync/sync-packages
  ```

- `<Description>` パッケージの説明。
- `<Priority>` パッケージの優先度。

  最も高い優先度は `1` で表されます。標準設定の同期パッケージには，最も低い優先度の `10` が割り当てられます。優先度の高いルールの結果によって優先度の低いルールの結果が上書きされます。

注: 同じ優先度の同期パッケージが複数存在する場合があります。同じ優先度の同期パッケージ間におけるルールの実行順序は指定されません。

マッピング・ファイル

次のマッピング・ファイルをトポロジ同期パッケージに含めることができます。

コンテキスト・マッピング(フィルタ): contextmapping.xml

HPOM サービス・ツリーのどの要素をトポロジ同期パッケージに含め, RTSM(実行時サービス・モデル)でのマッピングの対象とするかを指定するには, フィルタ・ファイル `contextmapping.xml` を設定します。フィルタでは, 同期する CI にコンテキストを割り当てます。コンテキストを設定することにより, 同じコンテキストの CI にマッピング・ルールを選択的に適用できます。

たとえば, 特定の HPOM サービスにタグ付けすると, ほかの設定ファイルに含まれる後続のマッピング・ルールがタグ付けされたサービスに適用されます。コンテキストが割り当てられていないサービスは同期の対象となりません。

タイプ・マッピング: typemapping.xml

タイプ・マッピング・ファイル `typemapping.xml` には, HPOM 内のサービスから RTSM 内の CI のタイプへのサービス属性に基づくマッピングを定義します。

属性マッピング: attributemapping.xml

属性マッピング・ファイル `attributemapping.xml` には, HPOM 内のサービスの属性と RTSM 内の CI の属性とのマッピングを定義します。

属性のマッピングにより, CI 属性を変更したり新しい属性を追加したりすることで, CI の記述を改善し, より詳細な環境のビューを作成できます。

関係マッピング: relationmapping.xml

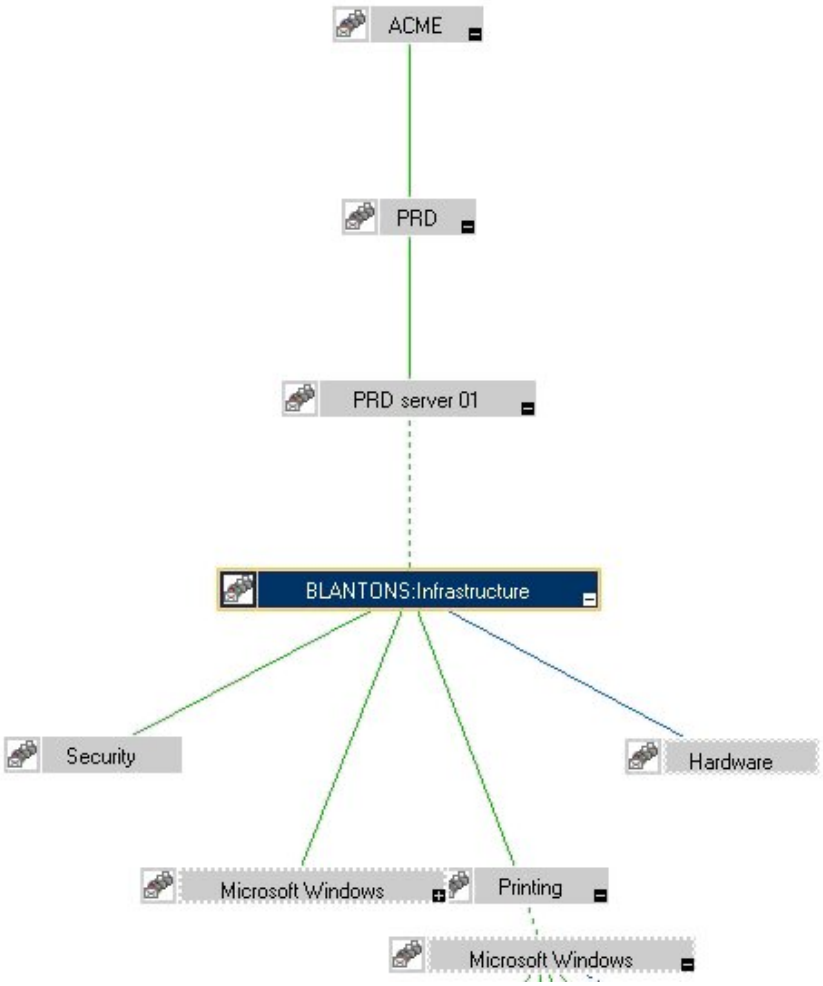
関係マッピング・ファイル `relationmapping.xml` を使用して, RTSM 内に作成済みの指定された HPOM サービス間の CI 関係を定義できます。

指定された HPOM サービスが RTSM 内に CI として作成されていることを確認してください。そうでない場合, トポロジ同期において RTSM 内に関係を作成することはできません。

トポロジ同期の設定: ACME 環境の例

本項では, 例として架空の "ACME" コンテンツ領域を使用してトポロジ同期の設定手順を説明します。

次の図は, HPOM ディスカバリからのサービス・ツリーを示しています。



パッケージ記述子ファイル package.xml の設定

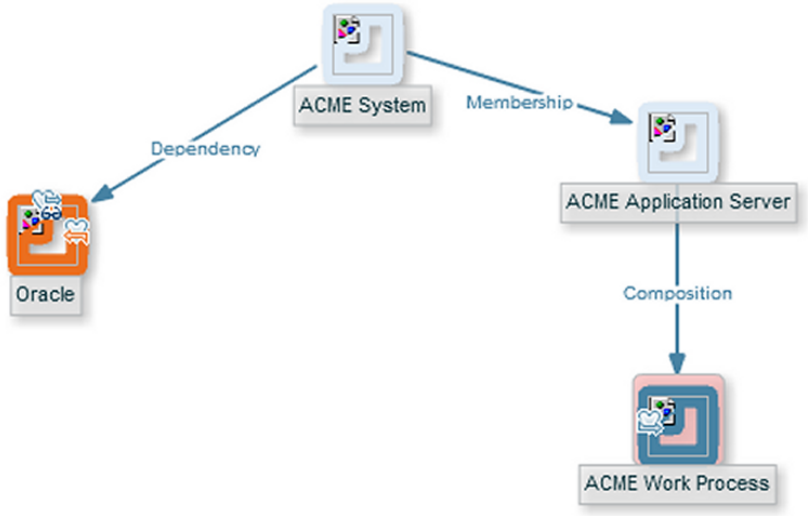
`package.xml` ファイルでは，トポロジ同期パッケージの名前を定義し，パッケージの説明と優先度を指定します。ACME トポロジ同期パッケージの `package.xml` ファイルは，次のように記述されています。

```
<Package>
   <Name>ACME</Name>
   <Description>Service to RTSM Mapping for ACME
Landscape.</Description>  <Priority>5</Priority>
</Package>
```

コンテキスト・マッピング(フィルタ)ファイル contextmapping.xml の設定

`contextmapping.xml` ファイルで，トポロジ同期の対象とするトポロジ・データの要素を RTSM(実行時サービス・モデル)でのマッピングのためにタグ付けします。タグ付けした要素には，ほかの設定ファイルに定義されたマッピング・ルールが適用されます。

次の図は，RSTM 内の ACME ビューを示しています。



次に，`contextmapping.xml` の設定例を示します。この例では，RTSM 内に CI を作成する，`ACME_System` タイプおよび `ACME_Application_Server` タイプの HPOM サービス要素に `ACME_Landscape` と呼ばれるコンテキストが割り当てられています。

```
<?xml version="1.0" encoding="UTF-8"?> <Mapping
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="../../../schemas/mapping.xsd"> <!--
CONFIGURE THE CIs THAT DEFINE THE CONTEXT FOR THE MAPPING --> <Rules>
<Rule name="Filter ACME Items"> <Condition> <!-- Select all Service
Elements of interest further refinements will be made later --> <Or>
<Equals> <OMType /> <Value>ACME_System</Value> </Equals> <Equals>
<OMType /> <Value>ACME_Application_Server</Value> </Equals> </Or>
</Condition> <MapTo> <Context>ACME_Landscape</Context> </MapTo>
</Rule> </Rules> </Mapping>
```

タイプ・マッピング・ファイル typemapping.xml の設定

タイプ・マッピング・ファイル `typemapping.xml` では，HPOM 内のサービスのサービス・タイプ定義と RTSM(実行時サービス・モデル) 内の CI のタイプとのマッピングを定義します。

ACME の例のタイプ・マッピングを次の表に定義します。

HPOM サービス・タイプ	CI タイプ(CI 名)
`ACME_System`	`acme_system`
`ACME_Application_Server`	`acme_appserver`

次に，コンテキスト `ACME_Landscape` を使用した，ACME 同期パッケージのタイプ・マッピング・ファイル `typemapping.xml` の設定例を示します。`ACME_System` タイプの HPOM サービス要素が RTSM 内の `acme_system` タイプの CI にマップされ，`ACME_Application_Server` タイプの HPOM サービス要素が RTSM 内の `acme_appserver` タイプの CI にマップされています。

```
<?xml version="1.0" encoding="UTF-8"?> <Mapping
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="../../../schemas/mapping.xsd"> <Rules
Context="ACME_Landscape"> <Rule name="Map ACME System"> <Condition>
<Equals> <OMType /> <Value>ACME_System</Value>       </Equals>
</Condition> <MapTo> <CMDBType> <Value>acme_system</Value> </CMDBType>
</MapTo> </Rule> <Rule name="Map ACME Application Server"> <Condition>
<Equals> <OMType /> <Value>ACME_Application_Server</Value> </Equals>
</Condition> <MapTo> <CMDBType> <Value>acme_appserver</Value>
</CMDBType> </MapTo> </Rule> </Rules> </Mapping>
```

属性マッピング・ファイル attributemapping.xml の設定

属性マッピング・ファイル `attributemapping.xml` を設定して，HPOM 内のサービスの属性と RTSM(実行時サービス・モデル) 内の CI の属性とのマッピングを定義します。

ACME 環境の例において，どの HPOM サービス属性が RTSM 内のどの CI 属性にマップされているかを次の表に示します。

サービス・タイプ	サービス属性名	CI タイプ	CI 属性名
ACME_System	Caption	ALL	display_label
ACME_Application_Server	OMId	ALL	data_name

次に，ACME 同期パッケージの対応する属性マッピング・ファイル `attributemapping.xml` の一部を示します。ここには 2 つの属性マッピングが示されています。

- `ACME_system` タイプの HPOM サービス要素では，HPOM サービス属性 `Caption` が RTSM 内の CI 属性 `display_label` にマップされています。
- `ACME_Application_Server` タイプの HPOM サービス要素では，HPOM サービス属性 `OMId` が RTSM 内の CI 属性 `data_name` にマップされています。

```
<?xml version="1.0" encoding="UTF-8"?> <Mapping
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="../../../schemas/mapping.xsd"> <Rules
Context="ACME_Landscape"> <Rule name="Map ACME System attributes">
<Condition> <Equals> <OMType /> <Value>ACME_System</Value> </Equals>
</Condition> <MapTo> <CMDBAttribute> <Name>display_label</Name>
<SetValue> <Caption /> </SetValue> </CMDBAttribute> </MapTo> </Rule>
<Rule name="Map ACME Application Server attributes"> <Condition>
<Equals> <OMType /> <Value>ACME_Application_Server</Value> </Equals>
</Condition> <MapTo> <CMDBAttribute> <Name>data_name</Name> <SetValue>
<OMId /> </SetValue> </CMDBAttribute> </MapTo> </Rule> </Rules>
</Mapping>
```

関係マッピング・ファイル relationmapping.xml の設定

関係マッピング・ファイル `relationmapping.xml` では，RTSM(実行時サービス・モデル) に作成済みの指定された HPOM サービス間の CI 関係を定義します。

次に，ACME 同期パッケージの関係マッピング・ファイル `relationmapping.xml` の設定例を示します。この例では，次の関係を作成しています。

- HPOM サービス・タイプが `ACME_Application_Server` である CI の CI 属性 `root_container` をホストに設定します。さらに，ホストと CI の間に `container_f` 関係を暗黙的に作成します。
- `ACME_System` タイプおよび `ACME_Application_Server` タイプの HPOM サービス要素間の構成関係 `container_f` 。

```
<?xml version="1.0" encoding="UTF-8"?> <Mapping
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="../../../schemas/mapping.xsd"> <Rules
Context="ACME_Landscape"> <Rule name="Create relation ACME Application
Server to node"> <Condition> <StartsWith> <OMType /> <Value>ACME_
Application_Server</Value> </StartsWith> </Condition> <MapTo>
<RootContainer> <DependencyCI relationType="hosted_on"> <True />
</DependencyCI> </RootContainer> </MapTo> </Rule> <Rule name="Create
relation between ACME Application Server and ACME System"> <Condition>
<And> <Equals> <OMType /> <Value>ACME_Application_Server</Value>
</Equals> <Equals> <AncestorCI relationType="container_f"> <Equals>
<OMType/> <Value>ACME_System</Value> </Equals> </AncestorCI>
<ParentCI/> </Equals> </And> </Condition> <MapTo> <RelationFrom>
<From> <AncestorCI relationType="container_f"> <Equals> <OMType/>
<Value>ACME_System</Value> </Equals> </AncestorCI> </From>
<Type>member</Type> </RelationFrom> </MapTo> </Rule> </Rules>
</Mapping>
```

同期のカスタマイズとスクリプティング

スクリプティングによって，同期プロセスにおいて，マッピングの前や RTSM(実行時サービス・モデル) へのトポロジ・データのアップロードの前後に追加の処理やカスタマイズを実行できます。各同期パッケージに，プレマッピング・スクリプト，プレアップロード・スクリプト，およびポストアップロード・スクリプトをそれぞれ 1 つ関連付けることができます。

詳細については，193ページ「スクリプティング」を参照してください。

同期パッケージの場所

`sync-packages` ディレクトリには，各同期パッケージに専用のサブディレクトリがあります。同期パッケージ名と一致するディレクトリ名を使用することを推奨しますが，必須ではありません。

同期パッケージを展開するには，パッケージを次のディレクトリに配置します。

`<HPBSM ルート・ディレクトリ>/conf/opr/topology-sync/sync-packages/<SyncPackageName>`

スクリプティング

スクリプティングによって，同期プロセス中に追加の処理やカスタマイズを実行できます。

- プレマッピング・スクリプトは，マッピング・ルールが適用される前に実行されます。
- プレアップロード・スクリプトは，マッピングの完了後，データを RTSM(実行時サービス・モデル) にアップロードする前に実行されます。
- ポストアップロード・スクリプトは，データを RTSM にアップロードにした後に実行されます。

個々の同期パッケージに各タイプのスクリプトを 1 つずつ関連付けることができます。これらのオプションのスクリプト・ファイルは，関連する同期パッケージ・ディレクトリにあります。同期パッケージの場所の詳細については，192ページ「同期パッケージの場所」を参照してください。

同期パッケージにスクリプト・ファイルを関連付けることにより，スクリプトの配布が簡略化され，スクリプトの展開を作業環境に依存することなく処理できます。同期スクリプトの実行は，次の同期パッケージ設定に従います。

- アクティブな同期パッケージ内のスクリプトだけが実行されます。
- スクリプトは，同期パッケージの優先順位に従って実行されます。

注意: スクリプトの実行は安全ではない可能性があります。特に，`scriptInterface.exec(...)` コマンドを使用すると，インストールが破損するおそれがあります。セキュリティを高めるために，編集を目的とするスクリプトへのアクセスはファイル・システム・レベルでのみ許可されています。そのため，OMi ホストへのログオン資格情報を持つユーザだけがスクリプトを編集できます。これにより，スクリプトが OMi ホストのログオン・セキュリティによって保護されます。

Groovy スクリプト

Groovy スクリプティングがサポートされています。Groovy は，Java プラットフォーム用の高水準のオブジェクト指向スクリプト言語であり，Java バイトコードにコンパイルされます。

Groovy は Java 開発者向けの言語であり，Java プラットフォームと緊密に統合されています。Groovy では，Java 仮想マシン(JVM) 上で Python や Ruby などの言語と同等の強力で簡潔なコーディング構文を使用できます。Java ビーンがサポートされており，JVM 内で Java クラスと Groovy クラスの置き換えが可能です。Groovy は Java コードおよびライブラリと緊密に統合されており，Java セマンティクスとすべての J2SE および J2EE API を再利用できるため，新たなセマンティクスや API の習得，実装，管理は必要ありません。

Groovy および Groovy 言語を説明するドキュメントに関する詳細については，次の URL にアクセスしてください。

http://groovy.codehaus.org(英語サイト)

トポロジ同期パッケージには 3 つの Groovy スクリプトを含めることができ，それらは同期パッケージ・ディレクトリ内で固定された名前で識別されます。各スクリプトは，同期プロセス内の指定されたポイントで実行されます。

- `preEnrichment.groovy` — マッピングの前に実行されるスクリプト

  `preEnrichment.groovy` スクリプトは，トポロジ同期のマッピング・プロセスが開始される前に実行されます。

- `preUpload.groovy` — アップロードの前に実行されるスクリプト

  `preUpload.groovy` スクリプトは，マッピング・プロセスの完了後，データがRTSM(実行時サービス・モデル)に書き込まれる前に実行されます(追加のCIの作成や既存のCIインスタンスへの詳細の追加などを目的として)。

- `postUpload.groovy` — アップロードの後に実行されるスクリプト

  `postUpload.groovy` スクリプトは，アップロードしたデータをRTSMに保存した後で，アップロード・プロセス中に保存されたデータを変更するために実行されます(ログ記録や監査などを目的として)。

アップロードは，`preUpload.groovy` スクリプトと`postUpload.groovy` スクリプトの間に実行されます。

スクリプトの有効化と無効化

Groovy スクリプトの使用は，標準設定で有効になっています([HPOMトポロジ同期設定]で`[Enable usage of Groovy scripts]`が`[true]`に設定されています)。

同期が失敗した原因を特定する場合は，スクリプティングを無効にしてください。スクリプトにエラーがある場合，スクリプティングを無効にすることで，同期が成功する可能性が高くなります。

トポロジ同期パッケージ・スクリプトの実行を無効にするには，インフラストラクチャ設定マネージャの[HPOMトポロジ同期設定]で`[Groovy スクリプト使用の有効化]`の設定を`[true]`から`[false]`に変更します。

[インフラストラクチャ設定] > [アプリケーション] > [オペレーション管理] > [オペレーション管理 - HPOMトポロジ同期] > [Enable usage of Groovy scripts]

Groovy スクリプトの場所

Groovy スクリプトは，トポロジ同期マッピング・ルールと同じディレクトリに配置する必要があります。

`<HPBSM ルート・ディレクトリ>/conf/opr/topology-sync/sync-packages/<SyncPackageName>`

スクリプト変数

各スクリプトに次の2つの定義済み変数があります。

- `ScriptInterface`

  オブジェクト・タイプ: `com.hp.opr.ts.scripting.ScriptInterface`

  説明: 同期データを操作し，同期を制御するCI情報関数呼び出しへのアクセスを有効にします。

  このオブジェクト・タイプは次のインタフェースを実装します。

  `com.hp.opr.ts.interfaces.scripting.IScriptingInterface`

- SyncData

  オブジェクト・タイプ: com.hp.opr.ts.common.data.sync.SyncData

  説明: 同期されたデータへのアクセスを提供します。

  このオブジェクト・タイプは次のインタフェースを実装します。

  com.hp.opr.ts.interfaces.data.sync.ISyncData

独自のトポロジ同期スクリプトを作成するために必要なインタフェースおよびオブジェクト・タイプの詳細については，次のディレクトリにある Java API のマニュアルを参照してください。

<HPBSM ルート・ディレクトリ>/opr/api/doc/opr-ts-interfaces-javadoc.zip

1 つの同期パッケージに含まれるスクリプトは，同じ変数スコープを共有します。したがって，preEnrichment.groovy で割り当てた変数は，対応する preUpload.groovy および postUpload.groovy でも使用できます。異なる同期パッケージのスクリプトは同じ名前の変数を共有しません。これは，名前の競合や好ましくない副次的影響を避けるためです。

エラーの処理

スクリプト内にエラーがあると例外が生成されます。エラー処理はスクリプトの呼び出しごとに行われます。標準設定では，スクリプト内で例外が発生すると同期が失われます。この動作は次のコマンドを呼び出すことによって変更できます。

```
scriptInterface.setAbortSyncOnError(boolean)
```

false に設定した場合は，abortSync("...") メソッドを使用してスクリプト・エラーを強制的に発生させることができます。たとえば，スクリプトで条件をチェックすると，この強制エラーによって，同期化を完了できません。

次の表は，同期ステータス(同期成功または同期失敗)とスクリプト動作の関係を示しています。

ステータス	スクリプト動作
同期成功	スクリプト内部のエラーも強制同期中断もなしでスクリプト作成が完了した。 例外がスローされ，AbortSyncOnError が false に設定された場合でもエラーなしでスクリプト作成が完了した。
同期失敗	スクリプト実行で例外が発生したか，abortSync(String) コマンドを使用するというスクリプト作成条件のために，スクリプトで強制的にエラーが発生した。

サンプル・スクリプト: preUpload.groovy

次のスクリプトは，サンプル・スクリプト preUpload.groovy の一部です。

```
import com.hp.opr.ts.interfaces.data.ci.*  ; import
com.hp.opr.ts.common.data.ci.*  ; import java.util.*; import
java.lang.String;
```

```
List resourceGroups = new LinkedList (); List haMembers      = new
LinkedList ();

// Get all HPOM services, hosts and node groups for (ICi ci
:syncData.getConfigurationItems()) {


if (ci.getOmTypeId() == "Class_RG") { // タイプが "Class_RG" である場
合，HPOM 属性 IP アドレスのすべてのエントリ     // について，タイプが IP の CI を作成す
る

scriptInterface.logInfo ("add resource group"); resourceGroups.add
(ci); }


}


// ip-ci および関係をクラスタ・パッケージに作成する for (ICi ipCi
:resourceGroups) { HashMap hm = new HashMap(); // HPOM サービス固有の属性
を取得する hm = ipCi.getOmAttributes();



// IP アドレス属性の CI を作成する ICi newCi = scriptInterface.createCi();
newCi.setContext ("cluster"); newCi.setCmdbAttribute ("ip_address",
hm.get("ipaddr")); newCi.setCmdbAttribute ("ip_domain",
"\${DefaultDomain}"); newCi.setCmdbTypeId ("ip");
scriptInterface.logInfo ("create relationship between two ip-ci:"
        + hm.get("ipaddr") + " and cluster package " ); // クラスタ・パッケー
ジと IP との "contained"(包含)関係を作成する
scriptInterface.createCmdbRelation(ipCi, newCi, "contained");

    } }
```

テストとトラブルシューティング

本章の内容

XML 設定ファイルの検証

提供されている XML スキーマ定義を使用することにより，XML 設定ファイルの正確性を検証できます。提供されている XML スキーマ定義を使用すると，適切な XML エディタを使用した場合，新しい設定ファイルの作成も容易になります。XML ファイルをスキーマとの照合によって検証する機能を備えた Eclipse などのエディタを使用できます。

XSDXML スキーマ定義は，XML ファイルのコンテンツの記述および検証について定めた World Wide Web Consortium(W3C) の標準です。すべての XML 設定ファイルに XSD ファイルが用意されています。

詳細については，W3C が作成した XML スキーマに関するドキュメントを参照してください。このドキュメントは http://www.w3.org/XML/Schema(英語サイト) で入手できます。

XSD ファイル

スキーマ・ファイルは次のディレクトリに保存されています。

```
<HPBSM ルート・ディレクトリ>/conf/opr/topology-sync/schemas
```

このディレクトリには以下のファイルがあります。

`package.xsd`
各同期パッケージ内の `package.xml` ファイルを検証します。

`containmentrelations.xsd`
`containmentrelations.xml` ファイルを検証します。

`datadump.xsd`
データ・ダンプを有効にすることによって作成される同期データ・ファイルやエンリッチメント・シミュレータへの入力データとして使用される同期データ・ファイルを検証します。

`mapping.xsd`
同期パッケージに含まれる次のマッピング・ファイルを検証します。

- コンテキスト・マッピング - `contextmapping.xml`
- タイプ・マッピング - `typemapping.xml`

- 属性マッピング - `attributemapping.xml`
- 関係マッピング - `relationmapping.xml`

`nodetypes.xsd`
各同期パッケージ内のノード・タイプ・マッピング・ファイル `nodetypes.xml` を検証します。

ファイルの自動検証

設定ファイルは，読み込まれるたびに，関連するXSDファイルとの照合によって自動的に検証されます。ファイルを検証できない場合は，そのファイル内のエラーの位置を示すエラー・メッセージがエラー・ログに書き込まれます。

ファイルの手動検証

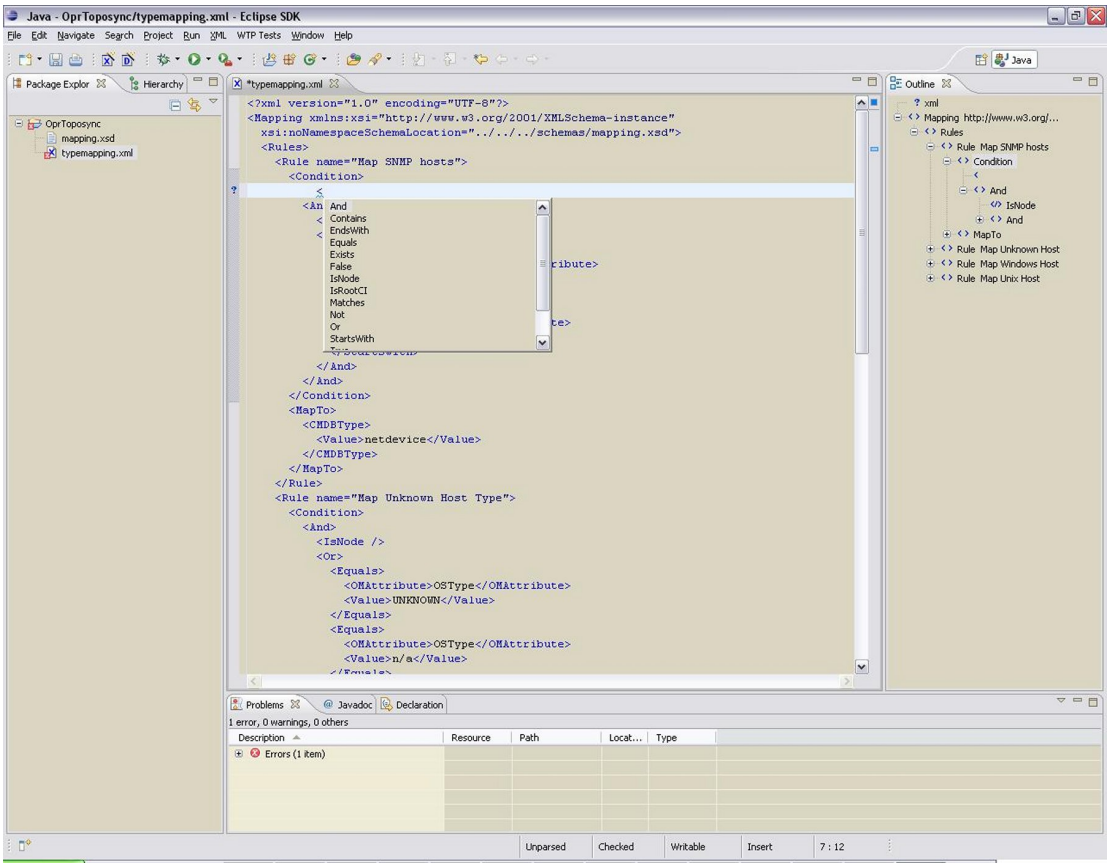
最新のXMLエディタでは，ファイルをスキーマに基づいて検証できます。たとえば，Eclipseでは，XMLドキュメントの最上位の要素がXSDファイルの参照である場合，スキーマに基づいてXMLファイルを検証できます。検証を有効にするには，XMLファイルの最上位の要素に次の属性を追加します。

```
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="<path or URL to schema file>"
```

<path or URL to schema file> には，検証に使用するスキーマ・ファイルの個別のパスまたはURLを指定します。たとえば，`contextmapping.xml` ファイルの場合は，次の参照を追加します。

```
<?xml version="1.0" encoding="UTF-8"?> <Mapping
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="<HPBSM root
directory>/conf/opr/topology-sync/schemas/mapping.xsd"> ...</Mapping>
```

この参照を追加すると，編集中に **CTRL+SPACE** キーを押すことにより，ファイルが検証され，有効な要素が提示されます。例については，次の図を参照してください。



注: XML ファイルに XSD の参照を追加した後で XML ファイルをいったん閉じて再度開かなければ，検証と有効な要素の提示が開始されないことがあります。

同期データのダンプ

同期データのダンプを次のような目的に使用できます。

- マッピング・ルールのトラブルシューティングによって誤ったマッピングを検出する。
- RTSM(実行時サービス・モデル)に送信されたデータとマッピング中に変更および追加されたデータを比較する。
- ダンプ・ファイルを作成してルールの XPath 式をチェックする。

同期データ・ダンプの作成

同期データ・ダンプは，同期されたトポロジ・データをマッピング・ルールでの XPath 式の一致に基づくデータ形式で記述した XML ファイルです。

ダンプには 2 つの種類があります。

- 1 つ目は，CI データの正規化が完了した後に記録されるダンプです。
- もう 1 つは，マッピング・ルールの処理が終了した後に記録されるダンプです。

同期データ・ダンプの作成をアクティブにするには，次の手順を実行します。

1. インフラストラクチャ設定マネージャの[HPOMトポロジ同期設定]に移動します。

   [インフラストラクチャ設定] > [アプリケーション] > [オペレーション管理] > [オペレーション管理 - HPOMトポロジ同期設定] > [ダンプ データ]

2. [**ダンプ データ**]の値を[**true**]に変更します。
3. 次のコマンドを入力して，トポロジ同期ツールを実行します。

   Windows: `<HPBSM ルート・ディレクトリ>/bin/opr-startTopologySync.bat`

   Linux: `<HPBSM ルート・ディレクトリ>/bin/opr-startTopologySync.sh`

データ・ダンプの例

次に，マッピングの実行後に記録されたデータ・ダンプの一部を示します。

```
<CI> <OMId>Root</OMId> <OMType /> <Caption>Root</Caption>
<Node>false</Node> <Service>false</Service> <OMAttributes /> <CMDBId
/> <CMDBAttributes /> <CMDBType /> <RootContainerId /> <Children>
<RelationType>container_f</RelationType> <CI> <Context>operations-
agent</Context> <OMId>03a2f7b2-ec88-7539-0532-c5b07da188dd</OMId>
<OMType>agent</OMType> <Caption>Operations-agent on met</Caption>
<Node>false</Node> <Service>false</Service> <OMAttributes>
<AgentId>03a2f7b2-ec88-7539-0532-c5b07da188dd</AgentId>
<Name>met.deu.hp.com</Name> </OMAttributes>       <CMDBId />
<CMDBAttributes> <data_name>03a2f7b2-ec88-7539-0532-
c5b07da188dd</data_name> </CMDBAttributes>      <CMDBType>hp_
operations_agent</CMDBType> <RootContainerId>{8BB8864B-CEC9-4B26-BD4C-
41F2C97C108E}</RootContainerId> <Dependencies> <RelationType>hosted_
on</RelationType> <CI> <Context>VISPI</Context>
<Context>nodegroups</Context> <OMId>{8BB8864B-CEC9-4B26-BD4C-
41F2C97C108E}</OMId> <OMType>node</OMType> <Caption>met</Caption>
<Node>true</Node> <Service>false</Service> <NodeGroupList>
<NodeGroupID>OpenView_Windows2000</NodeGroupID> <NodeGroupID>Root_
Nodes</NodeGroupID> </NodeGroupList> <MACAddressList /> <OMAttributes>
<AgentId>03a2f7b2-ec88-7539-0532-c5b07da188dd</AgentId>
<CommType>HTTPS</CommType> <DiscoveryDomain>${DefaultDomain}
</DiscoveryDomain> <Domain>deu.hp.com</Domain>
<Name>met.deu.hp.com</Name> <OSType>Windows_32</OSType>
<OSVersion>2000 (5.0)</OSVersion> <SystemType>x86/x64
Compatible</SystemType> <VirtualNodeType>0</VirtualNodeType>
</OMAttributes> <CMDBId />           <CMDBAttributes> <host_
dnsname>met.deu.hp.com</host_dnsname> <host_hostname>met</host_
hostname> <host_key>met.deu.hp.com</host_key> <host_os>2000 (5.0)
</host_os> </CMDBAttributes> <CMDBType>nt</CMDBType> <RootContainerId
/> </CI> </Dependencies> </CI> </Children> </CI>
```

同期データ・ダンプの表示

同期データ・ダンプを表示するには，次のディレクトリを開きます。

`<HPBSM ルート・ディレクトリ>/opr/tmp/datadump`

このディレクトリには，次のサブディレクトリがあります。

- `pre-enrichment`

  CI データ構造が正規化された後の同期データが含まれます。このデータには，HPOM から RTSM (実行時サービス・モデル) にロードされた内容が反映されています。

- `post-enrichment`

  正規化されたデータに対してマッピング・ルールが実行された後の同期データが含まれます。

- `ws-data`

  HPOM Web サービスから読み込まれた未処理データが含まれます。HPOM ノード，ノード・グループ，およびサービスごとに，`Caption_OMId.xml` と呼ばれる XML ファイルがあります。

  RTSM への書き込みが失敗した場合にのみ，XML ファイルが `post-ucmbd` ディレクトリに作成されます。

マッピング・ルールの検証

マッピング・ルールを検証するには，次の手順を実行します。

1. ファイルの差分を比較します。

   任意のファイル比較ツールを使用することにより，エンリッチメントにおいて何が変更されたかを簡単に確認できます。

2. XPath 式を検証します。

   正規化された同期データ・ダンプをXML エディタ XPath クエリをサポートしている XML エディタにロードすることによって，マッピング・ルールに使用されるXPath 式を検証します。

   注: XML ドキュメントでは，データ・ダンプに 1 つのルート要素 (`<ci>`) が含まれている必要があります。マッピング・ルール内の XPath クエリを実行するときには，このルート要素は存在していません。ダンプ・ファイルで検証する場合，絶対表記の式を作成するときには，式の先頭に `/ci` を付けてください。

ルールの作成

本項では，ルールを作成する際のガイドラインについて説明します。

ルール作成の簡略化

XML スキーマに従って要素を検証および提示する機能を備えた XML エディタを選択することにより，ルールの作成が容易になります。詳細については，197ページ「XML 設定ファイルの検証」を参照してください。

複雑な XPath クエリの回避

複雑な XPath クエリの使用は避けてください。特に，一般的な条件の複雑な XPath クエリは，すべての CI に適用することが必要となるため，使用すべきではありません。複雑な XPath クエリの使用が

避けられない場合は，OMType などの演算子とAnd 演算子を組み合わせることによって条件を絞り込んでください。XPath 以外の単純な条件から先にチェックされるようにします(ヒント: And は排他的演算子です)。

既存の属性に対してのみ一致させる

すべてのCI に存在するわけではない属性にアクセスする場合，相対表記の式を組み合わせると，CI 階層の複雑度によっては，パフォーマンスが大きな影響を受けます。

範囲の広い XPath 式の回避

複雑な XPath 式には，過度の処理負荷をもたらすものがあります。例として，次のような特性を持つXPath 式が挙げられます。

- // やdescendants:*/ を含む式など，複数のノードに適用される
- 現在のノードからきわめて離れた位置にあるノードと一致しないか，またはそのようなノードにのみ一致する

この点は，一致したすべての値を返すXPathResultList 演算子にも当てはまります。このような演算子に要する時間は，階層のサイズに対してほぼ倍の割合で増加します。このような式はできるだけ使用しないでください。

子孫演算子を使用するときには，ノード・テストとしてアスタリスク(*)を使用せず，ノード名を指定してください。たとえば，descendants:*/caption ではなくdescendants:ci/caption を使用します。

このような XPath 式を条件内で使用することが避けられない場合は，排他的演算子のAnd を使用して実行を制限し，XPathResult オペランドを使用する前に簡単なテストを行ってください。たとえば，CI タイプのチェックを先に実行します。

ログ・レベル設定

トポロジ同期では，同期プロセスの詳細がログ・ファイルに記録されます。デバッグのためにログの詳細レベルを変更できます。

サービス・ディスカバリ・サーバ・ログ・レベル設定

動的トポロジ同期のみ。サービス・ディスカバリ・サーバは，次のログ・レベルをサポートしています。

- ログ・レベル 1 では，エラーだけが記録されます。
- ログ・レベル 3 では，エラーと情報(エージェントから受け取った未処理データを含む) が記録されます。
- ログ・レベル 10 では，デバッグ用のトレース情報(メソッドのパラメータなど) が記録されます。

サービス・ディスカバリ・サーバのログ・レベルを変更するには，次の手順を実行します。

1. コマンド・プロンプトで次のコマンドを実行します。

   ```
   ovconfchg -edit
   ```

ログはメモ帳のウィンドウで開くことができます。

2. このファイルで，[om.svcdiscsserver] 名前空間に LOG_LEVEL=10 を追加します。

サービス・ディスカバリ・サーバは次のログ・ファイルを生成します。

Windows: %OvShareDir%\server\log\OvSvcDiscServer.log

Linux: /var/opt/OV/shared/server/log/OvSvcDiscServer.log

マッピング・ログ・レベル設定

マッピング・エンジンのログ・レベルを変更するには，次の手順を実行します。

1. 次のファイルをテキスト・エディタで開きます。

   <HPBSM ルート・ディレクトリ>/conf/core/Tools/log4j/wde/opr-svcdiscserver.properties

2. loglevel= で始まる行を探します。
3. ログ・レベルを次のいずれかの値に設定します(例: loglevel=INFO)。

   DEBUG は，アプリケーションのデバッグに最も役立つ詳細な情報イベントを示します。

   INFO は，アプリケーションの進行状況に関する詳細度の低い情報メッセージを示します。

   WARN は，害を及ぼす可能性のある状況を示します。

   ERROR は，アプリケーションが動作を継続できる可能性のあるエラー・イベントを示します。

   FATAL は，アプリケーションが停止する可能性が高い，きわめて重大なエラーを示します。

マッピング・エンジンは次のログ・ファイルを生成します。

<HPBSM ルート・ディレクトリ>/log/wde/opr-svcdiscserver.log

トラブルシューティング，一般的な問題，ヒント

トラブルシューティングを行うときには，まず180ページ「トポロジ同期ファイルの場所」に示すログ・ファイルを調べます。

次の表に，一般的な問題を示します。この問題は，ほかに指定がないかぎりトポロジ同期全般に適用されます。

症状	要因	ソリューション
トポロジ同期が失敗する。	HPOM for Windows に必要なパッチがインストールされていません。 必要なエージェント Hotfix やパッチに関する情報を含め，詳細については BSM Readme を参照してください。	Operations Manager for Windows : HPOM 8.1x for Windows では，Patch OMW_00138 またはそれに代わるパッチと OMW_00123 をインストールします。 HPOM 9.00 for Windows では，Patch OMW_00139 またはそれに代わるパッチと OMW_00124 をインストールします。 詳細については，『BSM 9.10 リリース・ノート』を参照してください。
		Operations Manager for UNIX or Linux : HPOM 9.10 for HP-UX では，Patch PHSS_42736 またはそれに代わるパッチをインストールします。 HPOM 9.10 for Linux では，Patch OML_00050 またはそれに代わるパッチをインストールします。 HPOM 9.10 for Solaris では，Patch ITOSOL_00772 またはそれに代わるパッチをインストールします。
基本トポロジ同期が失敗する。	Web サービスのポートが正しく設定されていません。	Web サービスのポートが正しく設定されていることを確認します。
	ユーザ名またはパスワードが間違っています。	HPOM の場合の形式: DOMAIN\Username.ユーザは少なくとも PowerUser 権限を持ち，HP-OVE-Admins のメンバである必要があります。
動的トポロジ同期が失敗する。	同期パッケージがディスク上で変更されたにもかかわらず，データベースにアップロードされていません。	`opr-sdtool` コマンドライン・ツールを実行して，同期パッケージへの変更をデータベースにアップロードします(183ページ「同期パッケージのアップロードとダウンロード」を参照)。

症状	要因	ソリューション
動的トポロジ同期の結果が不完全または空である。	ディスカバリ・ポリシー (`DiscoverOMTypes` および `DiscoverOM`) がオペレーション管理インスタンスを設定する前にターゲット・サーバとして HPOM に展開されています。	HPOM 管理サーバで `ovagtrep -publish` コマンドを実行します。このコマンドは，すべてのトポロジ・データを HPOM またはオペレーション管理インスタンスに再送信します。
急にノード CI が作成できなくなり，同期が失敗するようになった。	標準設定の同期パッケージが[トポロジ同期設定]から削除されています。	標準設定の同期パッケージが[トポロジ同期設定]から削除されているかどうかを確認します。標準設定 パッケージは，セミコロンで区切られたリストに常に存在している必要があります。
ログ・ファイルに警告がある。	モデル関連の問題。	直ちに対処する必要はありませんが，トポロジ同期のパフォーマンスが低下するおそれがあります。
独自の同期パッケージを作成したにもかかわらず，ログ・ファイルで不可解な RTSM(実行時サービス・モデル) 例外ばかりが起こる。	マッピング関連の問題。	データ・ダンプ・オプションを有効にして，*<HPBSM ルート・ディレクトリ>*`/opr/tmp/datadump/post-enrichment` ディレクトリにあるファイルに同期パッケージの CI に必要な属性がすべて含まれているかどうか確認します。

制限

本項では，トポロジ同期に関連する既知の制限について説明します。

デルタ検出の制限

HPOM 内のサービスまたはノードの属性が変更され，その属性またはノードが RTSM(実行時サービス・モデル) 内のキー属性にマップされている場合，デルタ検出では元の RTSM(実行時サービス・モデル) CI インスタンスが削除および置換されず，新たに追加のインスタンスが生成されます。

その例を次に示します。HPOM for Windows 内のエージェント ID が `aaaaa-bbbb-cccc-dddd` である管理対象ノードについて考えます。このノードは，RTSM(実行時サービス・モデル) 内のホスト CI とキー `aaaaa-bbbb-cccc-dddd` を持つエージェント CI にマップされます。

HPOM では，エージェント ID が変更されたため，現在のエージェント ID は `aaaaa-bbbb-cccc-eeee` です。キー属性が `aaaaa-bbbb-cccc-eeee` であるエージェント CI が新たに作成されますが，元のエージェント CI は削除されません。そのため，現在は，同じ(変更された) 管理対象ノードに関連する RTSM(実行時サービス・モデル) インスタンスが 2 つ存在します。

回避策: この問題を解決するためには，元の RTSM(実行時サービス・モデル) インスタンスを手動で削除する必要があります。

トポロジ同期の制限

HPOM ディスカバリ・サーバからのイベント変更は WMI リスナに登録され，WMI を介して RTSM(実行時サービス・モデル) に転送されます。WMI への負荷が高い場合など，特定の状況において，一部のイベントがオペレーション管理に到着せず，RTSM(実行時サービス・モデル) に反映されないことがあります。その結果，HPOM 内のステータスと RTSM(実行時サービス・モデル) の間に一時的な不一致が発生する可能性があります。この不一致は，次にツール `startInitialSync.bat` が実行されたときに解決されます。

マッピング・エンジンと構文

マッピングは，HPOM 内のサービス，属性，ノードを RTSM(実行時サービス・モデル) の CI にマップするために使用されるメカニズムです。ファイル形式，マッピング構文，および CI データ構造内のナビゲートに使用される XPath クエリ言語について，次の項で説明します。

- 207ページ「共通マッピング・ファイル形式」
- 207ページ「マッピング・ファイルの構文」
- 221ページ「XPath ナビゲーション」

共通マッピング・ファイル形式

注: ルール名は，現在のファイルのすべてのルールに対して一意である必要があります。

この例では，マッピング・ファイルの共通部分を示します。

```
<?xml version="1.0" encoding="utf-8"?> <Mapping> <Rules Context="web
server SPI"> <Rule name="Apache Server"> <Condition> <!-- ...Boolean
operators ...--> </Condition> <MapTo> <!-- ...Target Mappings ...-->
</MapTo> </Rule> <!-- ...More Rules ...--> </Rules> <!-- ...More Rule
sets with different contexts ...--> </Mapping>
```

マッピング・ファイルのコンポーネントについては，207ページ「マッピング・ファイルの構文」を参照してください。

マッピング・ファイルの構文

次の項では，トポロジ同期マッピング・ファイルで使用される有効な構文について説明します。

- 208ページ「ルール」
- 208ページ「ルール条件」
- 209ページ「演算子要素」
- 211ページ「オペランド要素」
- 216ページ「マッピング要素」
- 217ページ「フィルタリング」
- 217ページ「タイプ・マッピング」
- 218ページ「属性マッピング」
- 220ページ「関係マッピング」

ルール

<Rules> タグには一連のルールが含まれています。オプションの Context 属性を使用して，これらのルールを特定のコンテキストに制限できます。詳細については，217ページ「フィルタリング」を参照してください。

ルール条件

ルールの <Condition> 要素には，個別の条件が相互に関連する方法を指定するブール演算子が含まれており，たとえば Ant の <condition> タスクに似ています。

各演算子はオペランドに対する操作を実装できます。たとえば，属性 hosted_on には，.europe.example.com(属性 hosted_on と .europe.example.com はオペランドです) で終わる値，または <And>，<Or>，<Not> などのその他のネストされた演算子の 1 つまたはセットに対する操作があります。

条件の例

現在の HPOM サービスのタイプが testtype であるかどうかを確認します。

```
<Condition> <Equals> <OMType/> <Value>testtype</Value> </Equals>
</Condition>
```

CI が europe.example.com ドメインにあるノードに関連するかどうかを確認します。

```
<Condition> <EndsWith> <XPathResult>//.[node='true']/attributes/
            host_dnsName<XPathResult>
<Value>.europe.example.com</Value> </EndsWith> </Condition>
```

演算子要素

True

```
<True/>
```

この演算子は，ネストされたすべての演算子がtrueを返すときは常にtrueを返します。標準設定(フォールバック)ルールの宣言に有用です。アーリー・アウト・モードを使用しているマッピング・エンジンでは，この演算子が優先度が一番低い同期パッケージの最後でのみ使用されていることを確認します。

False

```
<False/>
```

常にfalseを返します。`False`要素を使用して，ルールを一時的に無効にすることができます。

And

```
<And> <!-- Operator --> <!-- Operator --> [... more operators ...]
</And>
```

ネストされているすべての演算子がtrueを返すときにtrueを返します。

`<And>`演算子は排他的です。つまり，最初の演算子の結果がfalseの場合，次の演算子は評価されません。この演算子は，最も単純な条件を最初に，最も複雑な条件を最後に置くことによってより高いパフォーマンスのルールを実装するために使用されます。

Or

```
<Or> <!-- Operator --> <!-- Operator --> [... more operators ...]</Or>
```

演算子の少なくとも1つがtrueを返す場合にtrueを返します。

Not

```
<Not> <!-- Operator --> </Not>
```

演算子がtrueを返さない場合にtrueを返します。

`<Not>`演算子は排他的です。つまり，子の演算子がtrueを返すとすぐに評価は停止されます。

Exists

```
<Exists> <!-- Operand --> <Exists>
```

オペランドの値はnullにしないでください。

Is Node

```
<IsNode/>
```

CIがノードとしてインポートされる場合，つまりCIタイプが`nodetypes.xml`ファイルにリストされる場合はtrue。

要素がHPOMで管理されるノードの場合はtrue。

Is Root CI

```
<IsRootCI/>
```

CIがルートCIである(ルートCIに親がない)場合はtrue。

Equals

```
<Equals> <!-- Operand --> <!-- Operand --> <!-- ...--> </Equals>

<Equals ignoreCase="[true|false]"> <!-- Operand --> <!-- Operand -->
<!-- ...--> </Equals>
```

オペランドの値が等しくなければなりません。2つ以上のオペランドがある場合，すべてのオペランドはそれぞれ等しくなる必要があります。オプションの属性 `ignoreCase` を使用して，大小文字の区別なしでオペランドの文字列値を比較することもできます。標準設定では，equals 演算子は大文字と小文字の区別を無視しません。

Starts With

```
<StartsWith> <!-- Operand --> <!-- Operand --> </StartsWith>
```

最初のオペランドの文字列値は，2番目のオペランドの値で始まる必要があります。

Ends With

```
<EndsWith> <!-- Operand --> <!-- Operand --> </EndsWith>
```

最初のオペランドの文字列値は，2番目のオペランドの値で終わる必要があります。

Matches

```
<Matches> <!-- Operand --> <!-- Operand --> </Matches>
```

最初のオペランドの文字列値が2番目のオペランドの正規表現に一致する必要があります。

例：

```
<Matches> <Attribute>host_dnsname</Attribute>
<Value>.*\.example\.com</Value> </Matches>
```

適用可能な正規表現の詳細については，次を参照してください。

http://java.sun.com/j2se/1.5.0/docs/api/java/util/regex/Pattern.html(英語サイト)

Contains

```
<Contains> <!-- Operand --> <!-- Operand --> <Contains>
```

最初のオペランドによって返される値には，2番目のオペランドの値が含まれている必要があります。オペランドの戻り値の型がリストの場合，リストには2番目のオペランドと等しい要素が少なくとも1つ含まれている必要があります。オペランドの戻り値の型が文字列の場合，2番目のオペランドの値は最初のオペランドのサブ文字列である必要があります。

Is Deletion CI

```
<IsDeletionCI/>
```

CI が CI の削除に使用される場合は true。この演算子は，動的トポロジ同期にのみ使用できます。基本トポロジ同期では CI の削除に異なるメカニズムが使用されるためです。基本トポロジ同期では，`IsDeletionCI` 演算子が無視されます。

オペランド要素

Operations Manager サービス ID

```
<OMId/>
```

戻り値の型 :String

オペレーション管理に保存されているCIのOM ID文字列を返します。OM IDは次のように異なる値を返します。

サービス:OM IDはサービスID

ノード :OM IDは一意のID

ノード・グループ:OM IDはノード・グループID

Operations Manager タイプ

```
<OMType/>
```

戻り値の型 :String

オペレーション管理に保存されているOMタイプを返します。HPOMサービスの場合，OMタイプはサービス・タイプ定義です。ノードの場合，OMタイプは定数値 "node" に設定されます。

CMDB タイプ

```
<CMDBType/>
```

戻り値の型 :String

RTSM(実行時サービス・モデル) に保存されているCIのCMDB CIタイプID文字列を返します。最初にこれはnullとして返されます。CMDBタイプが最初にタイプ・マッピングで設定される必要があるためです。これが設定されると，CMDB CIタイプID文字列は利用可能になります。

キャプション

```
<Caption/>
```

戻り値の型 :String

RTSMまたはBSMのCIのキャプション文字列を返します。

OM属性

```
<OMAttribute>[Name]</OMAttribute>
```

戻り値の型 :String

指定名のOM属性の値を返します。

CMDB 属性

```
<CMDBAttribute>[Name]</CMDBAttribute>
```

戻り値の型 :String

RTSMに書き込まれる指定名のCMDB属性の値を返します。属性マッピングが実行されるまで利用可能な属性はありません。

置換

```
<Replace [regExp="true|false"]>
    <In>
        <!-- 1st.Operand -->
    </In>     <For>
        <!-- 2nd.Operand -->
    </For>     <By>
        <!-- 3rd.Operand -->
    </By> </Replace>
```

戻り値の型 :String

2番目の演算子の戻り値のすべての発生に対する最初のオペランドの戻り値の文字列を3番目のオペランドの戻り値に置き換えます。たとえば，CI キャプションのバックスラッシュのすべての発生をアンダースコアに置き換えるには，次のように宣言する必要があります。

```
<Replace>     <In>
        <CiCaption/>
    </In>     <For>
        <Value>\</Value>
    </For>     <By>
        <Value>_</Value>
    </By> </Replace>
```

オプションで，2番目のオペランドに正規表現を使用できます。3番目のオペランドで前方参照を使用することもできます。

適用可能な正規表現の詳細については，次を参照してください。

http://java.sun.com/j2se/1.5.0/docs/api/java/util/regex/Pattern.html(英語サイト)

次の例では，正規表現を使用してドメイン名の一部を抽出します。

```
<Replace regExp="true">
    <In>
        <Attribute>host_dnsname</Attribute>     </In>
    <For>
        <Value>^[^.]*\.([^.]*).*</Value>     </For>
    <By>
        <Value>$1</Value>
    </By> </Replace>
```

属性 `host_dnsname` に値 `server.rio.example.com` が含まれている場合，`Replace` オペランドの結果は `rio` となります。

XPath 結果

```
<XPathResult>[XPath]</XPathResult>
```

戻り値の型 :String

XPath 式の値を返します。この値により，同じ階層に含まれる CI のデータにアクセスできるようになります。XPath 式では文字列値を選択する必要があり，複数の一致がある場合は任意の要素が返されます。

XPath の詳細については，221ページ「XPath ナビゲーション」を参照してください。

XPath 結果リスト

```
<XPathResultList>[XPath]</XPathResultList>
```

戻り値の型 :リスト

一致したすべての値のリストを返します。

XPath の詳細については，221ページ「XPath ナビゲーション」を参照してください。

値

```
<Value>[String]</Value>
```

戻り値の型 :String

定数値を返します。

リスト

```
<List> <!--Operand--> <!--Operand--> <!--...--> </List>
```

戻り値の型 :リスト

リスト・オペランドは，`contains` 演算子など，リストを入力パラメータとして受け入れる演算子と使用するためのものです。リスト・オペランドには，ほかのオペランドのリスト，返されたリストに追加される値が含まれます。

親 CI

```
<ParentCI/>
```

戻り値の型 :CI

現在の CI の親 CI を返します。現在の CI がルート CI の場合，null が返されます。

> **ヒント:** ルート CI を確認するには，`IsRoot` 演算子を使用します。

子 CI

```
<ChildCI> [Operator] </ChildCI>
```

```
<ChildCI relationType="[relationType]"> [Operator] </ChildCI>
```

戻り値の型 :CI

説明 :括弧で囲まれた演算子に一致する現在の CI の最初の子 CI を返します。

オプションの要素 :

`relationType` : 指定された関連タイプの関係にのみ従います。

子 CI リスト

```
<ChildCIList> [Operator (Optional)] </ChildCIList>
```

```
<ChildCIList relationType="[relationType]"> [Operator (Optional)]
</ChildCIList>
```

戻り値の型 :CI のリスト

現在の CI のすべての子 CI を返します。

オプションの要素:

`Operator`: 演算子に一致するCIのみ返されます。

`relationType`: 指定された関連タイプの関係にのみ従います。

祖先 CI

```
<AncestorCI> [Operator] </AncestorCI>
```

```
<AncestorCI relationType="[relationType]"> [Operator] </AncestorCI>
```

戻り値の型 :CI

括弧で囲まれた演算子に一致する現在のCIの最初の祖先 CIを返します。祖先 CIは，現在のCIの親，または親の親(など)です。

オプションの要素:

`relationType`: 依存関係に指定の関係タイプが含まれている必要があります。

子孫 CI

```
<DescendantCI> [Operator] </DescendantCI>
```

```
<DescendantCI relationType="[relationType]"> [Operator]
</DescendantCI>
```

戻り値の型 :CI

括弧で囲まれた演算子に一致する現在のCIの最初の子孫 CIを返します。子孫 CIは，現在のCIの子，または子の子(など)です。

オプションの要素:

`relationType`: 指定された関連タイプの関係にのみ従います。

子孫 CIリスト

```
<DescendantCIList> [Operator (Optional)] </DescendantCIList>
```

```
<DescendantCIList relationType="[relationType]"> [Operator (Optional)]
</DescendantCIList>
```

戻り値の型 :CIのリスト

現在のCIのすべての子孫 CIを返します。子孫 CIは，現在のCIの子，または子の子(など)です。

オプションの要素:

`Operator`: 演算子に一致するCIのみ返されます。

`relationType`: 指定された関連タイプの関係にのみ従います。

依存関係 CI

```
<DependencyCI> [Operator] </DependencyCI>
```

```
<DependencyCI relationType="[relationType]"> [Operator]
</DependencyCI>
```

戻り値の型 :CI

含まれる演算子に一致する最初の依存関係 CI を返します。

オプションの要素：

`relationType`：指定された関連タイプの関係にのみ従います。

依存関係 CI リスト

```
<DependencyCIList> [Operator (Optional)] </DependencyCIList>
```

```
<DependencyCIList relationType="[relationType]"> [Operator (Optional)]
</DependencyCIList>
```

戻り値の型 :CI

依存関係のリストを返します。

オプションの要素：

`Operator`：演算子に一致する CI のみ返されます。

`relationType`：依存関係に指定の関係タイプが含まれている必要があります。

依存 CI

```
<DependentCI> [Operator] </DependentCI>
```

```
<DependentCI relationType="[relationType]"> [Operator] </DependentCI>
```

戻り値の型 :CI

含まれる演算子に一致する最初の依存 CI を返します。

依存 CI の例：

ServiceA > hosted_on > HostB

この場合，ServiceA は HostB の依存 CI です。つまり，HostB を持っていて，このホストに依存するすべてのサービスを持とうとする場合，`<DependentCI>` オペランドを使用する必要があります。ServiceA を持っていて，HostB を持とうとする場合は，代わりに `<DependencyCI>` オペランドを使用する必要があります。

オプションの要素：

`relationType`：指定された関連タイプの関係にのみ従います。

依存 CI リスト

```
<DependentCIList> [Operator (Optional)] </DependentCIList>
```

```
<DependentCIList relationType="[relationType]"> [Operator (Optional)]
</DependentCIList>
```

戻り値の型 :CI

依存 CI のリストを返します。

依存 CI の例：

ServiceA > hosted_on > HostB

この場合，ServiceA は HostB の依存 CI です。つまり，HostB を持っていて，このホストに依存するすべてのサービスを持とうとする場合，<DependentCI> オペランドを使用する必要があります。ServiceA を持っていて，HostB を持とうとする場合は，代わりに <DependencyCI> オペランドを使用する必要があります。

オプションの要素：

Operator : 演算子に一致する CI のみ返されます。

relationType : 依存関係に指定の関係タイプが含まれている必要があります。

From CI Get

```
<From> <CI> [CI Operand] </CI> <Get> [Operand] </Get> </From>
```

戻り値の型 :2 番目のオペランドの戻り値の型

このオペランドを使用して，別の CI から値を取得できます。最初のオペランド [CI Operand] は，CI インスタンスを返す必要があります。2 番目のオペランドはその CI インスタンスで動作し，この 2 番目のオペランドの値はこの From オペランドによって返されます。

例：

```
<From> <CI> <ParentCI> </CI> <Get> <Caption/> </Get> </From>
```

現在の CI の親 CI からキャプションを返します。

接続元サーバ

```
<OriginServer/>
```

戻り値の型 :String

このオペランドは，最初に検出データを受信してからほかのサーバに転送するサーバのホスト名を返します。

マッピング要素

<MapTo> によってマッピングが定義されます。ここでは，エンジンを明確に実装するたびにその個別のマッピング用の独自の XML 要素が追加されます。

条件の例

現在の HPOM サービスのタイプが testtype であるかどうかを確認します。

```
<Condition> <Equals> <OMType/> <Value>testtype</Value> </Equals>
</Condition>
```

CI が europe.example.com ドメインにあるノードに関連するかどうかを確認します。

```
<Condition> <EndsWith> <XPathResult>//.[node='true']/attributes/
            host_dnsName<XPathResult>
<Value>.europe.example.com</Value> </EndsWith> </Condition>
```

フィルタリング

フィルタリングを行うと，コンテキストをこれらのCIに割り当てることによりトポロジ・データの興味のある部分を選択できます。このコンテキストにより，マッピング・ルールを同じコンテキストのCIに選択的に適用できます。コンテキストがアタッチされていないCIはすべて同期されません。

注: フィルタ・マッピング・ルール用の `<Rules>` タグには，コンテキスト属性を含めないでください。

コンテキスト・マッピング

```
<Context>[Context Name]</Context>
```

例：

```
<Mapping xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="../mapping.xsd"> <Rules> <Rule
name="Exchange Server Role Filter"> <Condition> <And> <Exists>
<XPathResult>ancestor::ci[omType='exch_spi_std_server']</XPathResult>
</Exists> <Equals> <OMType/> <Value>exch_spi_std_server_role</Value>
</Equals> </And> </Condition> <MapTo> <Context>exchange</Context>
</MapTo> </Rule> </Rules> </Mapping>
```

サービス・タイプ定義 `exch_spi_std_server_role` に割り当てられ，サービス・タイプ定義が `exch_spi_std_server` のサービスの下にあるすべてのCIは，`exchange` コンテキストに割り当てられています。

タイプ・マッピング

サービス・マッピングでは，HPOMサービス・タイプ定義がそのCMDBタイプにマップされます。

マッピング

```
<CMDBType> [Operand] ...</CMDBType>
```

演算子の結果の連結された文字列である指定のCMDBタイプにCIをマップします。2つ以上の `<CMDBType>` 要素が `<MapTo>` セクションにあってはなりません。

例：

```
<?xml version="1.0" encoding="utf-8"?> <Mapping>

  <Rules context="exchange"> </Rule> <Rule name="Map Exchange Server">
<Condition> <Equals> <OMType/> <Value>Exch2k7_ByServer</Value>
</Equals> </Condition> <MapTo> <CMDBType>
<Value>exchangeserver</Value> </CMDBType> </MapTo> </Rule> </Rules>
</Mapping>
```

OMタイプ `Exch2k7_ByServer` があり，コンテキスト `exchange` が割り当てられているすべてのCIがCMDBタイプ `exchangeserver` にマップされます。

```
<Mapping> <Rules> <Rule name="Map Windows Host Type"> <Condition>
<And> <IsNode/> <StartsWith> <OMAttribute>OSType</OMAttribute>
<Value>Windows</Value> </StartsWith> </And> </Condition> <MapTo>
<CMDBType> <Value>nt</Value> </CMDBType> </MapTo> </Rule> </Rules>
</Mapping>
```

文字列 `Windows` で始まる属性 `OSType` を持つすべてのノードが CMDB タイプ `nt` にマップされます。

属性マッピング

属性マッピング・ファイル `attributemapping.xml` により，HPOM のサービスの属性と RTSM(実行時サービス・モデル) の CI の属性の間のマッピングが定義されます。

指定の名前の属性の値を指定のオペランドの戻り値に設定します。2 つ以上のオペランドが指定されると，それらの値は連結されます。

RTSM の文字列値へのマッピング

```
<CMDBAttribute> <Name>[Attribute Name]</Name> <SetValue> [Operands]
</SetValue> </CMDBAttribute>
```

RTSM の最大長の文字列値へのマッピング

```
<CMDBAttribute> <Name>[Attribute Name]</Name> <SetValue Length="
[IntegerValue]"> [Operands] </SetValue> </CMDBAttribute>
```

RTSM の整数値へのマッピング

```
<CMDBAttribute> <Name>[Attribute Name]</Name> <SetIntValue> [Operands]
</SetIntValue> </CMDBAttribute>
```

RTSM のブール値へのマッピング

```
<CMDBAttribute> <Name>[Attribute Name]</Name> <SetBoolValue>
[Operands] </SetBoolValue> </CMDBAttribute>
```

RTSM のロング値へのマッピング

```
<CMDBAttribute> <Name>[Attribute Name]</Name> <SetLongValue>
[Operands] </SetLongValue> </CMDBAttribute>
```

RTSM の日付値へのマッピング

```
<CMDBAttribute> <Name>[Attribute Name]</Name> <SetdateValue>
[Operands] </SetdateValue> </CMDBAttribute>
```

RTSM の浮動小数値へのマッピング

```
<CMDBAttribute> <Name>[Attribute Name]</Name> <SetFloatValue>
[Operands] </SetFloatValue> </CMDBAttribute>
```

RTSM のバイト値へのマッピング

```
<CMDBAttribute> <Name>[Attribute Name]</Name> <SetByteValue>
[Operands] </SetByteValue> </CMDBAttribute>
```

RTSM の倍精度値へのマッピング

```
<CMDBAttribute> <Name>[Attribute Name]</Name> <SetDoubleValue>
[Operands] </SetDoubleValue> </CMDBAttribute>
```

RTSM の StringList 値へのマッピング

```
<CMDBAttribute> <Name>[Attribute Name]</Name> <SetStringListValue>
[Operands] (comma-separated) </SetStringListValue> </CMDBAttribute>
```

RTSM の IntList 値へのマッピング

```
<CMDBAttribute> <Name>[Attribute Name]</Name> <SetIntListValue>
[Operands] (comma-separated) </SetIntListValue> </CMDBAttribute>
```

属性マッピングの例

```
<Mapping xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="../../../schemas/mapping.xsd"> <Rules>
<Rule name="Map Display Label"> <Condition> <True/> </Condition>
<MapTo> <CMDBAttribute> <Name>display_label</Name> <SetValue>
<Caption/> </SetValue> </CMDBAttribute> </MapTo> </Rule> </Rules>
<Rules Context="exchange"> <Rule name="Set data_name key attribute">
<Condition> <True/> </Condition> <MapTo> <CMDBAttribute> <Name>data_
name</Name> <SetValue> <OMId/> </SetValue> </CMDBAttribute> </MapTo>
</Rule> <Rule name="Set host_key key attribute for nodes"> <Condition>
<IsNode/> </Condition> <MapTo> <CMDBAttribute> <Name>host_key</Name>
<SetValue> <OMId/> </SetValue> </CMDBAttribute> </MapTo> </Rule>
</Rules> </Mapping>
```

すべての CI(どのコンテキストが割り当てられていても) に対し，CMDB 属性 `display_label` は OM CI のキャプションに設定されます。コンテキスト `exchange` に割り当てられている CI には `data_name` が含まれ，ノードに対しては `host_key` 属性が OM ID に設定されます。

関係マッピング

関係マッピングを使用して，CI 間の関係を作成できます。トポロジ同期の場合，標準設定で OM の関連付けは関係として同期されません。これらの関係を明示的に定義する必要があります。

```
<RelationTo> <To> [Operand] </To> <Type>[RelationType]</Type>
</RelationTo>
```

現在の CI からオペランドによって返される CI までの関係を定義します。オペランドは，文字列，CI のインスタンス，CI のリスト，文字列のリストのいずれかを返す場合があります。文字列値は，関係が作成される CI の OM ID と一致する必要があります。リストの場合，関係はリストに含まれる各アイテム(文字列または CI) に対して作成されます。

関係には [RelationType] によって指定されるタイプがあります。このタイプは，ラベルではなく関係の名前です。

```
<RelationFrom> <From> [Operand] </From> <Type>[RelationType]</Type>
</RelationFrom>
```

前のマッピングと同様に動作しますが，方向は逆です。

ルート・コンテナ・マッピング

CMDB モデルにより，実際の CI を作成する前に作成が必要な特定のルート・コンテナ CI が定義されます。トポロジ同期では，CI を正しい順序で作成できるようにするためにそのような関係が認識される必要があります。

```
<RootContainer> [Operand] </RootContainer>
```

現在の CI のルート・コンテナは，オペランドの戻り値によって指定される CI に設定されます。戻り値は，文字列または CI のいずれかです。

CI 解決用メッセージ・エイリアス・マッピング

```
<RedirectMessagesOf> [Operand] </RedirectMessagesOf>
```

現在の CI のエイリアスは，オペランドの戻り値によって指定される OMId に設定されます。戻り値は，文字列，CI，CI または文字列のリストです。

例：

```
<Mapping> <Rules Context="exchange"> <Rule name="Create relation
server to node"> <Condition> <Equals> <OMType/> <Value>Exch2k7_
ByServer</Value> </Equals> </Condition> <MapTo> <RelationTo> <To>
<DependencyCI relationType="hosted_on"> <True/> </DependencyCI> </To>
<Type>is_impacted_from</Type> </RelationTo> <RelationTo> <To>
<DescendantCIList> <StartsWith> <OMType> <Value>Exch2k7_Role_</Value>
</StartsWith> </DependencyCI> </To> <Type>deployed</Type>
</RelationTo> <RootContainer> <DependencyCI relationType="hosted_on">
<True/> </DependencyCI> </RootContainer> <RedirectMessagesOf>
<ChildCIList/> </RedirectMessagesOf> </MapTo> </Rule> </Rules>
</Mapping>
```

STD `Exch2k7_ByServer` の CI により，(ホスト先のノードとの) タイプ `is_impacted_from` の関係，(`Exch2k7_Role_` で始まる OM タイプとの) すべての子孫 CI にデプロイされているタイプの関係が取得されます。

同じノードもルート・コンテナ CI です。

XPath ナビゲーション

XPath は，XML ドキュメントの一部を定義するための構文です。XPath では，XML ドキュメントのナビゲーションにパス式が使用されます。

XPath は CI データ構造内をナビゲートするためのマッピング・エンジンで使用されます。

XPath クエリ言語を熟知していない場合は，次の Web サイトの XPath チュートリアルを参照してください。

http://www.w3schools.com/xpath/(英語サイト)

データ構造

マッピング・ルールで使用される XPath 式の一致に表示されるデータ構造については，222ページ「次の図に，ナビゲーションに表示されるデータ構造を示します。」に記載されています。

CI データ構造

OMAttributes

元の RTSM(実行時サービス・モデル) CI 属性すべてのマップが含まれています。このマップのキーは，RTSM CI 属性の RTSM 値を参照する RTSM CI 属性の名前です。

Caption

Service Navigator に表示される CI の名前を表します。Caption には，RTSM CI 属性 `display_label` と同じ値が含まれています。

Children

現在の CI からほかの CI への包含関係を持つ CI との関係のリストを参照します。このフィールドを使用して，複合 XPath クエリを作成し，". ." XPath セレクタを使用して子および親の値を取得できます。

Dependencies

依存 CI との関係のリストを参照します。`Children` に似ています。ただし，参照されるオブジェクトは異なる階層に含まれています。

OMid

CI の一意の ID。

Node

これが HPOM のノードかどうかを示すブール値です。

Type

HPOM サービスのサービス・タイプが含まれています。

これは CI タイプの表示ラベルではありません。

Service

この要素が HPOM のサービスであるかどうかを示すブール値です。

ノード・グループでは Node と Service が FALSE に設定されています。

関係データ構造

CI

現在の CI が関連する CI への参照が含まれています。

RelationType

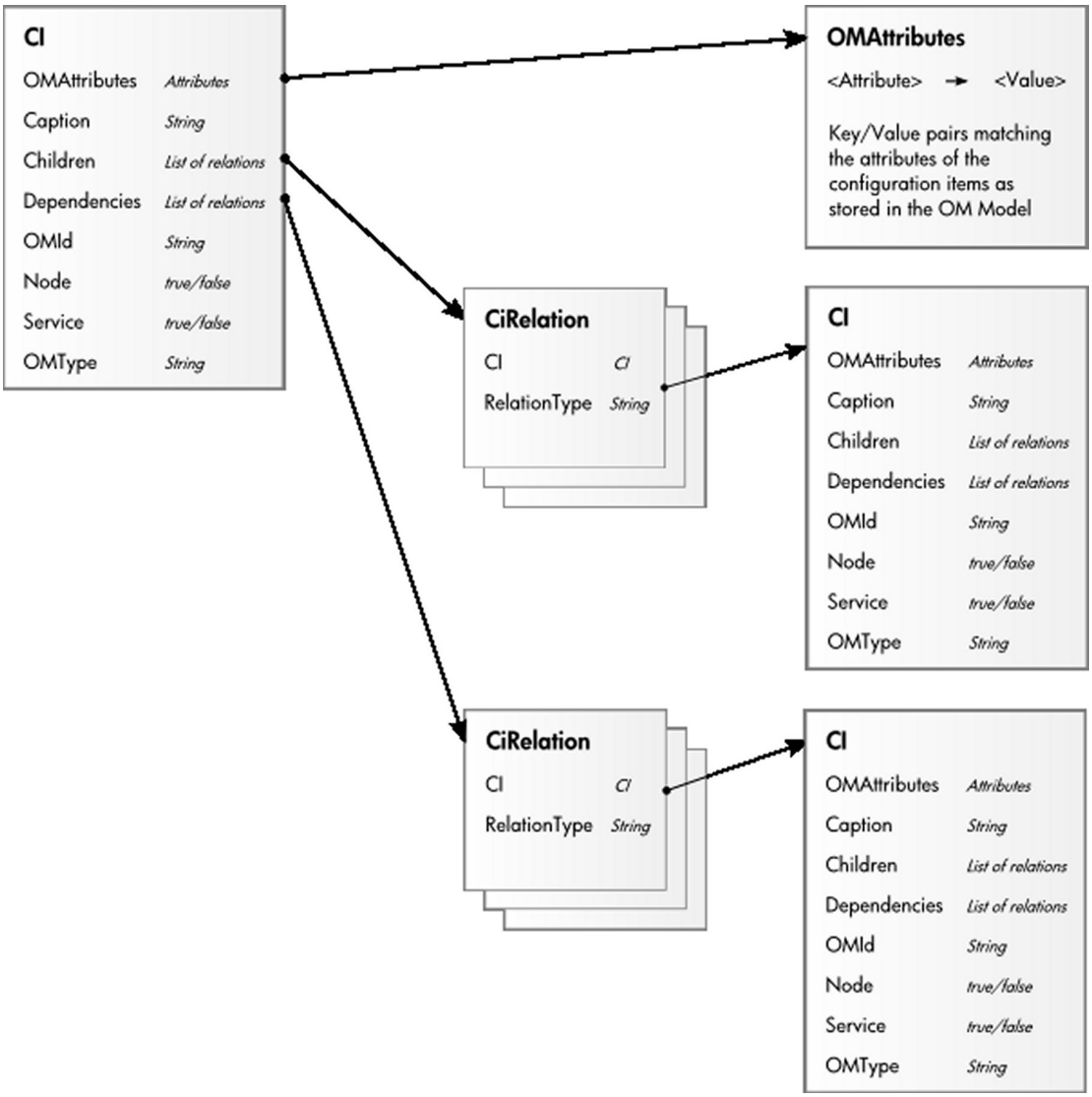
RTSM に保存されている関係タイプ。

これは CI タイプの表示ラベルではありません。

サービスの場合：
HPOM の包含関係である場合は `container_f` が含まれます。
HPOM の依存関係である場合は `dependency` が含まれます。
HPOM のホスト・オン関係である場合は `hosted_on` が含まれます。.

ノードの場合：
`container_node` または `dependency_node` が含まれています。

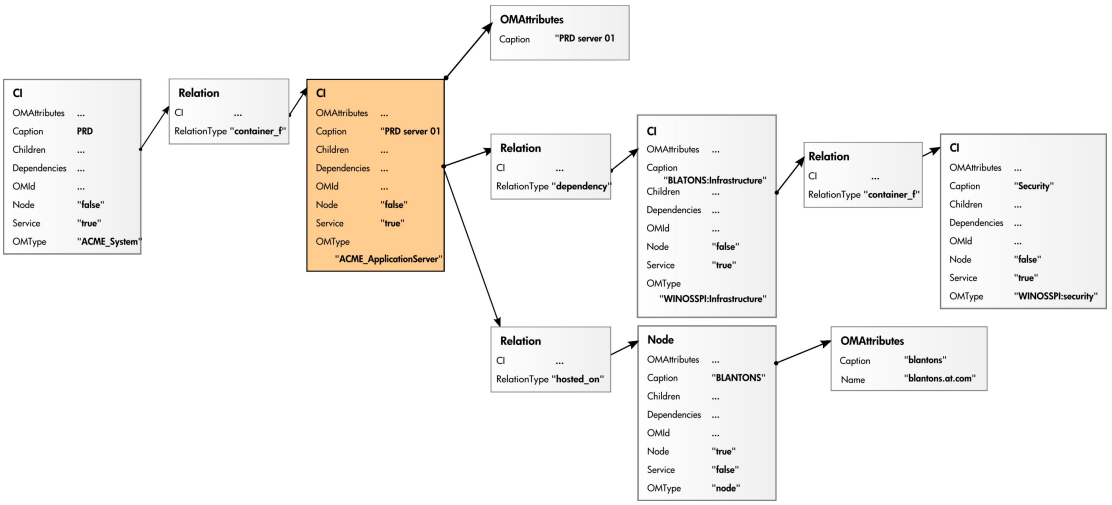
次の図に，ナビゲーションに表示されるデータ構造を示します。



XPath ナビゲート・データ構造の例

XPath ナビゲード・データ構造の例については，223ページ「次の図に，一部の XPath の例を示します。」を参照してください。ホストは，Oracle アプリケーションを HP-UX オペレーティング・システムで実行している UNIX システムです。ナビゲーションの開始位置またはコンテキストは，Oracle アプリケーションを表す CI です(オレンジ色の背景)。

次の図に，一部の XPath の例を示します。



XPath 式と例の値

次の表に，典型的な XPath 式をリストし，各式ごとに例を示します。

XPath 式，意味，例

XPath 式	意味	例
`Caption`	CI からのキャプション	`PRD server 01`
`./Caption`	CI からのキャプション	`PRD server 01`
`/Caption`	ルート(データベース) CI のキャプション	`PRD`
`../../Caption`	親 CI の親からのキャプションを選択	`PRD`
`../RelationType`	親の関係タイプを選択	`container_f`
`../../OMType`	親タイプの親を選択	`ACME_System`
`/OMType`	ルート CI のタイプを選択	`ACME_System`
`//.[type='WINOSSPI:Infrastructure']/Caption`	次のタイプの全 CI のキャプションを選択。`WINOSSPI:Infrastructure`	`BLANTONS:Infrastructure`
`//Dependencies[type='hosted_on']/CI/Caption`	`hosted_on` 依存関係を持つすべての CI のキャプションを選択	`BLANTONS`
`//Dependencies/CI/Caption`	依存関係を持つすべての CI のキャプションを選択	`BLANTONS`

注: XPath 式で開始データベース・ノード下のノードが選択されると，".." によって 1 つのステップがリード・バックされます。次の式ではノード db に読み込まれ，開始データベース・ノードにリンクされます。

```
//dependencies[type='hosted_on']/CI/../..
```

ただし，ノード db が開始ノードである場合，式 ../.. はノード db の包含リンクに続きます。これはこの例で示される依存関係ではありません。結果は，異なる階層であるノードの親コンテナに依存します。

第10章

イベント処理インタフェース

本項では，イベント処理においてイベントを変更および強化するイベント処理スクリプトとカスタム・アクションの役割について説明します。

本項の内容

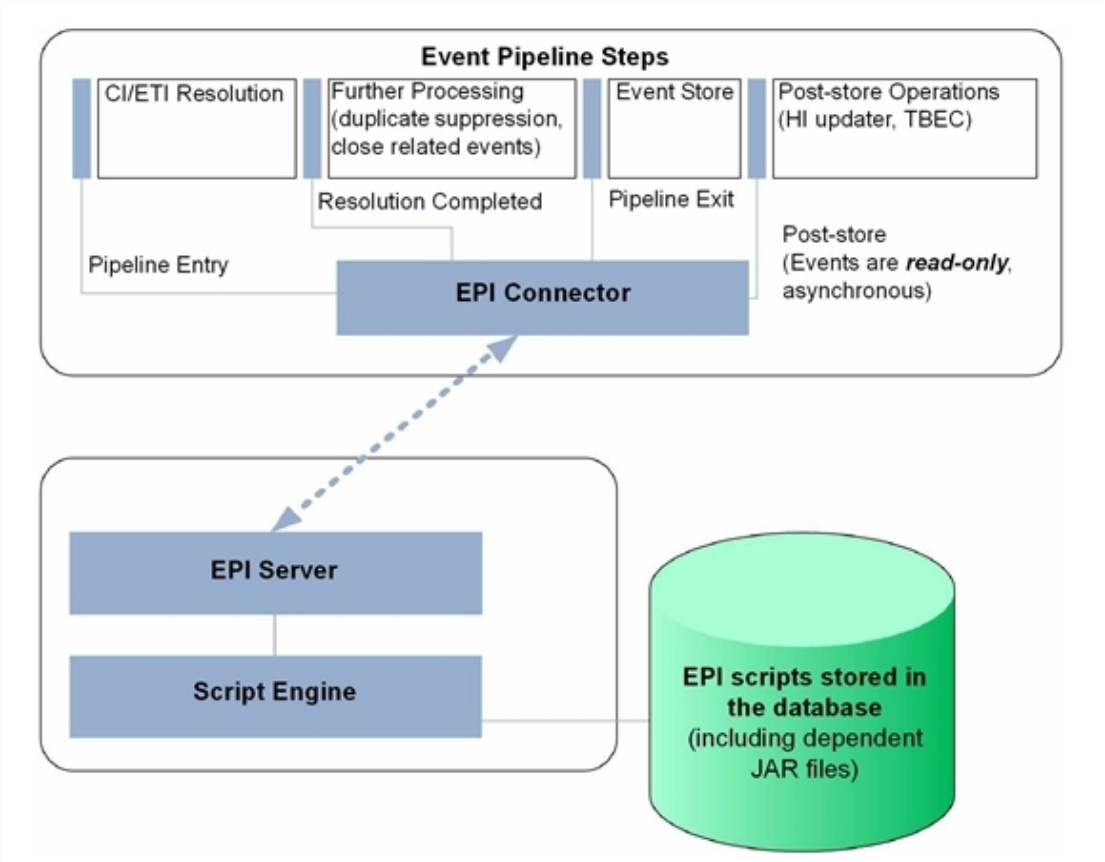
イベント処理インタフェース

本項では，イベント処理インタフェース(EPI) の概要を示し，イベント・パイプラインのさまざまな段階と，イベントを変更および強化するスクリプトを実行するための適切なエントリ・ポイントについて説明します。

イベント処理インタフェースとスクリプト

EPI によって，イベント処理においてユーザ定義の Groovy スクリプトを実行できます。Groovy スクリプトでは，イベントを外部データで変更および強化できます。たとえば，外部 SQL データベースや Excel リストから取得した追加のデータでイベントを強化することもできます。

227ページ「スクリプトのパイプライン・エントリ・ポイントを含むイベント処理」では，イベント処理の概要を示し，イベント・パイプラインとEPI スクリプト実行をこのプロセスに統合する方法を説明します。



スクリプトのパイプライン・エントリ・ポイントを含むイベント処理

スクリプトはデータベースに保存されます。スクリプトによって参照または使用される JAR ファイルやライブラリもデータベースにロードされます。

各パイプライン手順には，スクリプトを 1 つ以上設定できます。実行するスクリプト数には制限がありません。ただし，実行されるスクリプトの数が多いほど，パイプラインでのイベントの処理に要する時間が長くなることに注意してください。

注 : スクリプトの設計および実行は，イベント処理全体のコンテキストで行う必要があります。つまり，設定マネージャで行った，重複イベント抑制や関連イベントの終了などのほかのイベント処理の設定とスクリプトの相互作用を考慮する必要があります。

EPI スクリプト実行のエントリ・ポイント

イベント・パイプラインとは，イベント処理のさまざまな段階です。EPI スクリプトをイベント・パイプラインで実行するポイントは次の 4 つです。

- CI/ETI 解決の前
- CI/ETI 解決の後
- イベントがデータベースに保存される前
- イベントがデータベースに保存された後

CI/ETI 解決の前

CI および ETI が解決される前，つまりイベントのイベント・パイプラインが開始される前にスクリプトを実行します。

このポイントでスクリプトを実行すると，たとえば CI および ETI の解決に影響するヒントをさらに設定できます。これ以降のイベント・パイプラインのエントリ・ポイントでは CI および ETI の解決には影響を与えることができません。

CI/ETI 解決の後

CI および ETI の解決の直後の，重複イベントの抑制および関連イベントの自動終了などの処理が実行される前にスクリプトを実行します。

重複イベントの処理に影響を与える場合，イベント・パイプラインのこのエントリ・ポイントでスクリプトを実行します。通常は重複イベントの抑制を有効にするが，特定のイベント・タイプに対しては重複イベント抑制の設定を変更する場合があります。その場合，特定のイベント・タイプで重複イベント抑制を無効にするスクリプトをこのエントリ・ポイントで実行します。これ以降のイベント・パイプラインのエントリ・ポイントでは，重複イベント抑制の処理には影響を与えることはできません。

イベントを保存する前

すべてのイベント処理の実行が完了し，イベントがデータベースに保存される前にスクリプトを実行します。

イベント・パイプラインのこのエントリ・ポイントでは，イベントがデータベースに保存される前に，一部のテキストを変更したり，ナレッジ・ベースにリンクを挿入するスクリプトを実行できます。

イベントを保存した後

イベントをデータベースに保存後，スクリプトを実行します。この場合，スクリプトはすべて読み取り専用モードで実行されます。イベントは，データベースに保存されると変更できなくなるからです。

データベースにすでに保存されている特定のタイプのイベントを，イベント・パイプラインのこのエントリ・ポイントでスクリプトを実行して，別のアプリケーションに送信するとます。または，指定したデータベースに保存されているイベントを監査ログに記述するスクリプトを実行できます。

スクリプトの指定

EPI スクリプトは，設定ユーザ・インタフェースの次の領域で設定します。

[管理] > [オペレーション管理] > [イベントの自動化] > [イベント処理のカスタマイズ]

EPI スクリプトの作成，コピー，編集，および削除を行うことができます。スクリプトの実行順序を指定することもできます。

スクリプトの作成の詳細については，232ページ「EPI スクリプトおよびカスタム・アクション・スクリプトの作成」を参照してください。

スクリプトの管理およびイベント処理への適用の詳細については，オペレーション管理のオンライン・ヘルプを参照してください。

EPI スクリプトはコンテンツ・パックに定義し，コンテンツ・マネージャを使用してインポートまたはエクスポートできます。

EPI スクリプトの実行

スクリプトは次の手順で実行されます。

1. EPI サーバがスクリプトを読み込み，`init()` 関数を呼び出します。
2. EPI サーバが `process()` 関数を呼び出し，パラメータとして指定されたイベントのリストを渡します。スクリプトはメモリに残り，一致するイベントが到着すると，`process()` 関数が呼び出されます。
3. ユーザがスクリプトを無効したり，システムがシャットダウンされたりすることにより，スクリプトがアンロードされた場合，EPI サーバは `destroy()` 関数を呼び出します。

   `destroy()` 関数は，スクリプトが Event Processing Customizations マネージャで変更された場合にも呼び出されます。この場合，`destroy()` 関数は元の変更されていないスクリプトに対して呼び出され，その後で変更後の新しいスクリプトに対して `init()` 関数が呼び出されます。

スクリプトの `process()` 関数の呼び出しは，データベース・トランザクションの処理の信頼性を確保する ACID(原子性，一貫性，独立性，永続性) 原則に従います。

- **原子性**

  スクリプトを正常に実行できない(エラーなしで終了せず，例外がスローされない) 場合，その時点までに実行された変更がロールバックされます。それ以降にイベントが発生すると，イベント・リストはエラーとなったスクリプトが実行される前の状態に戻ります。

- **一貫性**

  イベントの一貫性に違反する値は，システムによってオーバーライドされます。たとえば，ユーザ ID とグループ ID を設定し，そのユーザ ID がそのグループ ID のメンバでない場合，一貫性への違反となり，この設定はオーバーライドされます。

- **独立性**

読み取り/書き込みアクセスを持つスクリプトは，並行して実行されるのではなく，順次実行されます。

- **永続性**

  スクリプトの実行によってイベントに加えられた変更は，データベースに保存されます。

スクリプトの作成

EPI スクリプトとカスタム・アクション・スクリプトは，同じスクリプト定義形式を共有します。

スクリプトの作成の詳細については，232ページ「EPI スクリプトおよびカスタム・アクション・スクリプトの作成」を参照してください。

カスタム・アクションのスクリプト

本項では，カスタム・アクションのスクリプトを設定する方法について説明します。カスタム・アクションによって，イベントに適用する独自のアクションを定義できます。イベント・ブラウザでカスタム・アクションを使用できるようにする Groovy スクリプトを設定できます。

- 231ページ「カスタム・アクション・スクリプトの指定」
- 231ページ「スクリプトの作成」

カスタム・アクション・スクリプトの指定

注: オペレーション管理カスタム・アクションを指定または実行するためには，[BSM User Management]設定でユーザに適切な権限が付与されている必要があります。これを行う方法の詳細については，BSM のオンライン・ヘルプを参照してください。

カスタム・アクション・マネージャによって，イベントに対してカスタム・アクションを実行するスクリプトを設定できます。簡単な例を挙げると，特定のイベントにテキスト文字列を追加し，そのイベントをイベント・ブラウザで見つけやすくすることができます。

カスタム・アクションは，Groovy スクリプトとして指定します。カスタム・アクションのスクリプトは，設定ユーザ・インタフェースの次の領域で設定します。

[管理] > [オペレーション管理] > [操作コンソール] > [カスタム アクション]

オペレーション管理でカスタム・アクションを設定すると，そのスクリプトが[スクリプト]ペインのスクリプト・リストに表示されます。このスクリプトは，ショートカット・メニューの[**custom action list**]でイベントからトリガできます。選択したカスタム・アクションは，選択したイベントに関連付けられている CI のコンテキストで起動されます。未割り当てのイベントからカスタム・アクションが実行された場合，そのイベントはそのカスタム・アクションを実行したユーザに自動的に割り当てられ，対応するエントリがイベント履歴に作成されます。

カスタム・アクションのスクリプトを作成，コピー，編集，および削除できます。スクリプトの実行を停止することもできます。

スクリプトの作成の詳細については，232ページ「EPI スクリプトおよびカスタム・アクション・スクリプトの作成」を参照してください。

カスタム・アクションの設定方法の詳細については，オペレーション管理 のオンライン・ヘルプを参照してください。

カスタム・アクション・スクリプトをコンテンツ・パックに定義し，コンテンツ・マネージャを使用してインポート/エクスポートできます。

スクリプトの作成

EPI スクリプトとカスタム・アクション・スクリプトは，同じスクリプト定義形式を共有します。

スクリプトの作成の詳細については，232ページ「EPI スクリプトおよびカスタム・アクション・スクリプトの作成」を参照してください。

EPI スクリプトおよびカスタム・アクション・スクリプトの作成

本項では，EPI スクリプトとカスタム・アクション・スクリプトの作成方法について説明します。

本項の内容

スクリプト定義属性

スクリプトにはスクリプト定義が必要であるため，スクリプト定義属性を指定する必要があります。

スクリプト定義は，次の属性で構成されます。

- **Name**: 内部スクリプト名 (スクリプトのファイル名 **ではない**) 。
- **Classpath / JAR files**: 1 つまたは複数の JAR ファイルをスクリプトとともにアップロードできます。JAR ファイルの内容は，スクリプトの実行中にクラスパスで使用できます。クラスパスでの JAR ファイルの順序は，UI で JAR ファイルを上下に移動することによって変更できます。
- **Filter**: 任意，EPI スクリプトのみ。フィルタは名前で指定できます。フィルタに一致するイベントだけがスクリプトに渡されます。
- **Read-only**: オプション。Read-only 属性が指定されたスクリプトは，イベントを変更しません。これらのスクリプトは読み取り/書き込みスクリプトと非同期に実行されます。この非同期の実行によってイベント処理全体が高速化されるため，イベントの変更を目的としないスクリプトに Read-only 属性を設定することをベスト・プラクティスとしてお勧めします。
- **Active**: スクリプトの実行を有効または無効にします。スクリプトを有効にすると，`init()` 関数が呼び出されます。同じように，スクリプトを無効にすると，`destroy()` 関数が呼び出されます。
- **Timeout**: オプション。各スクリプト呼び出しの最大時間。この属性はイベント数に依存しません。タイムアウトの標準設定値は 0 です。これは，スクリプトの実行がタイムアウトを起こさないことを意味します。

  同期スクリプトの場合は，タイムアウトになると，スクリプトの実行が中止され，イベントに対するすべての変更がロールバックされます。

  非同期スクリプトの場合は，タイムアウトになると，スクリプトの実行が中止されます。

Groovy スクリプト API

独自の EPI スクリプトやカスタム・アクション・スクリプトを作成する場合は，最低でも `init()` 関数，`process()` 関数，および `destroy()` 関数の呼び出しを含む Groovy スクリプトを実装する必要があります。Groovy スクリプト API とすべての引数および型の詳細は，製品に含まれる Java API のマニュアルに記載されています。

Java API のマニュアルには，スクリプトの作成に必要な次の情報が含まれています。

- イベント・クラス: メイン・インタフェース
- 変更に使用できるイベント属性の完全なリスト

Java API のマニュアルは次の場所にあります。

<HPBSM のルート・ディレクトリ>/opr/api/doc/opr-external-api-javadoc.zip

スクリプト定義形式

EPI スクリプトおよびカスタム・アクション・スクリプトのスクリプト定義の基本的な形式は，次のとおりです。

```
import com.hp.opr.api.scripting.Event;

def init() {    // このメソッドは，スクリプトがロードされたとき (設定ユーザ・インタフェースで    //
スクリプトが有効にされた場合など) に呼び出される。}

def destroy() {    // このメソッドは，スクリプトがアンロードされたとき (設定ユーザ・インタフェー
ス // でスクリプトが無効にされた場合など) に呼び出される。 }

def process(List<EpiEvent> events) {    // このメソッドは，イベントが処理されるときに
呼び出される。このリストの型
  // は java.util.List。


  // このメソッドでは，イベントのプロパティを変更できる。スクリプト  // が読み取り専用モードである
場合は，UnsupportedOperationException が   // スローされる。

// 変更に使用できるイベント属性のリストは，Java API のマニュアルにある // opr-external-
api.jar の説明を参照。 }
```

サンプル EPI Groovy スクリプト

本項には，製品に付属のサンプル EPI Groovy スクリプトが含まれています。

サンプル EPI Groovy スクリプトは次のディレクトリにあります。

<HPBSM ルート・ディレクトリ>/opr/examples/epi_scripts

SimpleExampleEPI.groovy

ここでは，すべてのイベント属性をサンプル値に設定する簡単なスクリプトの例を示します。

```
import java.util.Date; import java.util.List;

import com.hp.opr.api.scripting.Action; import
com.hp.opr.api.scripting.Event; import
com.hp.opr.api.scripting.EventActionFlag; import
com.hp.opr.api.scripting.LifecycleState; import
com.hp.opr.api.scripting.MatchInfo; import
com.hp.opr.api.scripting.NodeInfo; import
com.hp.opr.api.scripting.PolicyType; import
```

```
com.hp.opr.api.scripting.Priority; import
com.hp.opr.api.scripting.ResolutionHints; import
com.hp.opr.api.scripting.Severity;

/* * このサンプル・スクリプトは, すべてのイベント属性を特定のサンプル値に設定する。 */
class SimpleExample { def init() { }
def destroy() { }
def process(List<Event> events) { events.each { event -> modifyEvent
(event); } }
def modifyEvent(Event event) { String application =
event.getApplication();
event.setApplication("Modified by EPI:" + application);
long groupId = event.getAssignedGroupId(); event.setAssignedGroupId
(groupId);
int assignedUserId = event.getAssignedUserId();
event.setAssignedUserId(assignedUserId);
Action autoAction = createSampleAction(); event.setAutoAction
(autoAction);
String category = event.getCategory(); event.setCategory("Modified by
EPI:" + category);
String correlationKeyPattern = event.getCloseKeyPattern();
event.setCloseKeyPattern("Modified by EPI:" + correlationKeyPattern);
String description = event.getDescription(); event.setDescription
("Modified by EPI:" + description);
String etiInfo = event.getEtiHint(); event.setEtiHint(etiInfo);
String correlationKey = event.getKey(); event.setKey("Modified by
EPI:" + correlationKey);
MatchInfo matchInfo = createSampleMatchInfo(); event.setMatchInfo
(matchInfo);
event.setNoDedup(true);
ResolutionHints hints = createSampleResolutionHints();
event.setNodeHints(hints);
String object = event.getObject(); event.setObject("Modified by EPI:"
+ object);
String omServiceId = event.getOmServiceId(); event.setOmServiceId
(omServiceId);
String omUser = event.getOmUser(); event.setOmUser(omUser);
String originalText = event.getOriginalData(); event.setOriginalData
("Modified by EPI:" + originalText);
String originalId = event.getOriginalId(); event.setOriginalId
(originalId);
event.setPriority(Priority.HIGHEST);
String ciInfo = event.getRelatedCiHint(); event.setRelatedCiHint
("Modified by EPI:" + ciInfo);
event.setSeverity(Severity.CRITICAL);
String solution = event.getSolution(); event.setSolution("Modified by
EPI:" + solution);
ResolutionHints sourceCiHints = createSampleResolutionHints();
event.setSourceCiHints(sourceCiHints);
event.setState(LifecycleState.IN_PROGRESS);
```

```
String subCategory = event.getSubCategory(); event.setSubCategory
("Modified by EPI:" + subCategory);
event.setTimeReceived(new Date());
String title = event.getTitle(); event.setTitle("Modified by EPI:" +
title);
String type = event.getType(); event.setType("Modified by EPI:" +
type);
Action userAction = createSampleAction(); event.setUserAction
(userAction); }
def ResolutionHints createSampleResolutionHints() { ResolutionHints
hints = new ResolutionHints(false);
hints.setCoreId("CoreId"); hints.setDnsName("mydqdn.com");
hints.setHint("My Hint"); hints.setIpAddress("0.0.0.0"); return hints;
}
def MatchInfo createSampleMatchInfo() { MatchInfo matchInfo = new
MatchInfo(false);
matchInfo.setConditionId("conditionId"); matchInfo.setPolicyName
("policyName"); matchInfo.setPolicyType(PolicyType.CONSOLE); return
matchInfo; }
def Action createSampleAction() { NodeInfo actionNodeInfo = new
NodeInfo(false);
Action action = new Action(false); actionNodeInfo.setCoreId("CoreId");
actionNodeInfo.setDnsName("myfqdn.com"); actionNodeInfo.setIpAddress
("0.0.0.0");
action.setCall("Call"); action.setNode(actionNodeInfo);
action.setStatus(EventActionFlag.AVAILABLE); return action; } }
```

RegExample.groovy

ここでは，単語を 1 つおきに前の単語と入れ替えるスクリプトの例を示します。

```
import java.util.Date; import java.util.List;

import com.hp.opr.api.scripting.Action; import
com.hp.opr.api.scripting.Event; import
com.hp.opr.api.scripting.EventActionFlag; import
com.hp.opr.api.scripting.LifecycleState; import
com.hp.opr.api.scripting.MatchInfo; import
com.hp.opr.api.scripting.NodeInfo; import
com.hp.opr.api.scripting.PolicyType; import
com.hp.opr.api.scripting.Priority; import
com.hp.opr.api.scripting.ResolutionHints; import
com.hp.opr.api.scripting.Severity;

/* * このスクリプトは単語を 1 つおきに前の単語と入れ替える。*/ class RegExpExample {
def init() { }

        def destroy() { }

        def process(List<Event> events) { events.each { event ->
```

```
event.setTitle(event.getTitle().replaceAll(/(\w+)\s+(\w+)/, '$2 $1'));
} } }
```

ResolveLocationFromDB.groovy

次のスクリプトは，IP アドレスをノード名と一致させ，データベースから場所を解決します。

このサンプル・スクリプトを MS SQL Server で使用するには，次の手順を実行します。

1. 新しい DB `asset_db` を作成します(または，次の名前を調整します)。
2. 新しいテーブル `LocationMapping` を次の属性で作成します(または，次の名前を調整します)。
   - `ip (varchar(50), primary key)`
   - `location (varchar(50))`
   - `phone (varchar(50))`
   - `contact (varchar(50))`
3. このテーブルに，イベントのノード・ヒントと一致する各 IP アドレスの行を追加します。
4. `Sql.newInstance` のデータベース・パラメータを次のように調整します。
5. このスクリプトを EPI スクリプトとして追加します。JTDS ドライバをアップロードします。JTDS ドライバは次の場所にあります。

   `<HPBSM .ルート・ディレクトリ>/lib/jtds-1.0.jar`

```
import groovy.sql.Sql;

class ResolveLocationFromDB
 {
 def connection; void init()
 {
 connection = Sql.newInstance("jdbc:jtds:sqlserver://localhost/asset_
db", 'sa', 'installed', "net.sourceforge.jtds.jdbc.Driver");
  }
void process(eventList)
{
try
{
for (event in eventList)
{
def nodeHints = event.getNodeHints(); def nodeName =
nodeHints.getDnsName(); if(nodeName != null)
{
def ipaddress = InetAddress.getByName(nodeName).hostAddress
connection.eachRow('select * from LocationMapping',
{
if (it.ip == ipaddress)
{
if (event.getDescription() == null) event.setDescription("CI located
```

```
in building:" + it.location) else event.setDescription
(event.getDescription() + "\nCI located in building:" + it.location)
event.addCustomAttribute('phone', it.phone) event.addCustomAttribute
('contact', it.contact) event.addCustomAttribute('location',
it.location)
}
});
}
}
}

finally { }
 }

void destroy() { connection.close(); }
 }
```

サンプル・カスタム・アクション Groovy スクリプト

本項では，カスタム・アクション Groovy スクリプトの例をいくつか示します。

サンプル・カスタム・アクション Groovy スクリプトは次のディレクトリにあります。

<HPBSM ルート・ディレクトリ>/opr/examples/ca_scripts

SimpleExample.groovy

次に，イベントを変更する簡単なカスタム・アクション・スクリプトの例を示します。

```
import com.hp.opr.api.scripting.Event; import
com.hp.opr.api.scripting.Priority; import
com.hp.opr.api.scripting.Severity;

class SimpleExample { def init() { // 何も初期化しない } def destroy() {
// 何も破棄しない } def process(List<Event> events) { events.each { event
-> modifyEvent(event); } } def modifyEvent(Event event) {
event.addCustomAttribute("CA_SCRIPT", "MODIFIED"); event.setSeverity
(Severity.CRITICAL); event.setPriority Priority.HIGHEST; } }
```

TranslateEventTitle.groovy

このスクリプトは，タイトルを英語からドイツ語に翻訳します。

注: このカスタム・アクション・スクリプトを実行するためには，Google Translate API をダウンロードする必要があります。Google Translate API は，次のサイトからダウンロードできます。

```
http://google-api-translate-java.googlecode.com/files/google-api-
translate-java-0.92.jar
```

この jar ファイルをスクリプトと合わせてカスタム・アクションとしてアップロードします。

```
import java.util.List; import com.hp.opr.api.scripting.Event; import
com.google.api.translate.Language; import
com.google.api.translate.Translate; class TranslateTitleToGerman { def
init() { // HTTP 参照元をユーザの Web サイト・アドレスに設定す
る。Translate.setHttpReferrer("***MY_REFERER***");
// 送信接続のプロキシ・アドレスを設定する。プロキシが不要な場合は, // 次の 2 行を削除しても
よい。System.setProperty("http.proxyHost", "***PROXY_HOST***");
System.setProperty("http.proxyPort", "***PROXY_PORT***"); }

       def destroy() { // 何も破棄またはシャットダウンしない }

       def process(List events) { for (Event e :events) { translateTitle(e);
} }

       def translateTitle(Event event) { String translatedText =
com.google.api.translate.Translate.execute( event.getTitle(),
Language.ENGLISH, Language.GERMAN); event.setTitle(translatedText); }
}
```

EPI トラブルシューティング

本項には，EPI スクリプトおよびカスタム・アクション・スクリプトの実行のトラブルシューティングに役立つ情報が含まれています。

ログ・ファイル

EPI スクリプトおよびカスタム・アクション・スクリプトの実行のトラブルシューティングを行う場合は，最初に次のログ・ファイルを調べることをお勧めします。

`<HPBSM ルート・ディレクトリ>\log\opr-scripting-host\opr-scripting-host.log`

デバッグ

デバッグを行うには，次の手順を実行します。

1. 次の場所へ移動します。

   `<HPBSM ルート・ディレクトリ>/conf/core/Tools/log4j/opr-scripting-host/opr-scripting-host.properties`

2. `opr-scripting-host.properties` ファイルで，`loglevel` を望ましい値に設定します。

   デバッグ・ログ・レベル(`loglevel=DEBUG`) は，問題を検出するのに役立ちます。

ログ・ファイル・エントリ

ログ・ファイル・エントリから，スクリプトとその実行に関する有益な情報が得られます。たとえば，次のような情報を得ることができます。

- スクリプトがいつロードまたはシャットダウンされたか。
- スクリプトの実行がいつタイムアウトになったか。
- スクリプトの実行に関する統計情報(スクリプトの実行に要した時間など) 。

第11章

オペレーション管理 UI とほかのアプリケーションとの統合

本項では，ドリルダウン URL 起動を使用して，オペレーション管理ユーザ・インタフェースの一部と外部アプリケーションを統合する方法について説明します。

本項の内容

イベント・ブラウザの URL 起動

URL リンクを使用して，イベント・ブラウザを起動できます。これは特に，オペレーション管理 ユーザ・インタフェース(UI) の一部と外部アプリケーションを統合しようとしている統合担当者に関連しています。統合によって，グラフィカルなユーザ・インタフェースを持つ外部アプリケーションのオペレータは，オペレーション管理 UI にドリルダウンできます。たとえば，オペレータがイベント・ブラウザとイベント詳細をアプリケーションのブラウザで起動できる，ポータル・アプリケーションの例を挙げられます。

本項では，そのような URL 起動の指定方法について説明します。次の項が含まれます。

- 241ページ「URL 起動の指定」
- 242ページ「パラメータとパラメータ値」
- 244ページ「カラムの定義」
- 246ページ「フィルタの設定」
- 250ページ「イベント詳細の URL 起動」

URL 起動の指定

イベント・ブラウザの URL 起動を実行する場合，既定 URL を使用するか，追加パラメータを指定できます。URL で指定するオプション・パラメータにより，イベント・ブラウザでの表示と動作を定義できます。

既定 URL 起動

既定 URL 起動を使用すると，イベント・ブラウザは標準設定のオペレーション管理 UI 設定で開きます。

注: 既定 URL で起動されたイベント・ブラウザでは，行う変更(表示カラム，カラム幅など) は自動的に保存されません。

標準設定の UI 設定でイベント・ブラウザを起動するには，次のように URL を入力します。

`http://<ホスト名 :ポート>/opr-console/opr-evt-browser`

注: 使用しているポートが標準設定の HTTP ポート(80) でない場合，ポート番号を指定する必要があります。標準設定のポートを使用している場合，ポート番号を指定する必要はありません。

オプション・パラメータの指定

追加パラメータを指定して，イベント・ブラウザに何を表示するかを定義できます。

利用可能なパラメータがその可能な値とともに242ページ「イベント・ブラウザの URL 起動用パラメータ」に記載されています。

URL 起動用の追加パラメータを次のように指定します。

```
http://<ホスト名:ポート>/opr-console/opr-evt-browser?<ここにパラメータを設定>
```

注: 使用しているポートが標準設定の HTTP ポート(80) でない場合，ポート番号を指定する必要があります。標準設定のポートを使用している場合，ポート番号を指定する必要はありません。

注: 文字 "?" を URL の最初の引数の前に置き，その後，各引数を "&" 文字で区切ります。

オプション・パラメータが指定された URL 起動の例を次に示します。

```
http://my.example.com:8080/opr-console/opr-evt-
browser?sortField=severity&sortOrder=desc&filter_severities=critical,
major,minor
```

パラメータとパラメータ値

242ページ「イベント・ブラウザの URL 起動用パラメータ」には，オペレーション管理 イベント・ブラウザの URL 起動に利用可能なパラメータが含まれています。

イベント・ブラウザの URL 起動用パラメータ

パラメータ	説明	可能な値	標準設定値
activeUpdates	イベント・ブラウザで更新がサーバから定期的に取得されるかどうかを指定します。定期的な更新がない場合，すべてのデータをブラウザに取得できない可能性があることに注意してください。	次のうちいずれか: • true • false	true
checkParams	不明な URL 起動パラメータが検証されるかどうかを決定します。 checkParams パラメータを指定しない場合，不明なパラメータ(スペルが間違っているパラメータなど) は検証されません。イベント・ブラウザでは不明なパラメータは無視され，エラー通知なしで開かれます。既知のパラメータは常に検証されます。 checkParams パラメータを true に設定すると，イベント・ブラウザではすべてのパラメータが検証され，不明なパラメータがエラーとしてレポートされます。	次のうちいずれか: • true • false	false

パラメータ	説明	可能な値	標準設定値
columns	イベント・ブラウザで表示されるカラムを定義します。	次のうち 1 つ以上 :<*Column key*>(244 ページ「表示するカラムを定義する固定カラム・キー」を参照)	標準設定のカラム・セット
context	ブラウザ設定のコンテキストを定義します。コンテキストの選択時にブラウザ設定(表示カラム, カラム幅など)を編集すると, その設定は指定のコンテキストに保存されます。次回同じコンテキストでブラウザを開くと, そのブラウザ設定が復元されます。 context パラメータを指定しない場合, ブラウザ設定は保存されません。 使用例 :context=myOwnContext	任意のユーザ定義文字列	標準設定値セットなし
headerText	イベント・ブラウザのヘッダ・テキストを設定します。	任意のユーザ定義文字列	Event Browser
showClosed	システム起動時に閉じられたイベント・ブラウザ設定を起動します。	次のうちいずれか: • true • false	false
showDetails	埋め込みのイベント詳細がすでに開いているイベント・ブラウザを起動します。	次のうちいずれか: • true • false	false

パラメータ	説明	可能な値	標準設定値
sortField	イベント・ブラウザが並べ替えられるカラムを定義します。	次のうちいずれか *:<Column key>* (244ページ「表示するカラムを定義する固定カラム・キー」を参照)	timeReceived
sortOrder	sortField パラメータによって指定されたカラムの並べ替え順序を昇順または降順で定義します。	次のうちいずれか: • asc • desc	desc

カラムの定義

利用可能なカラムの固定カラム・キーは, 244ページ「表示するカラムを定義する固定カラム・キー」で定義されています。245ページ「カスタム属性カラム・キー」には, ほかのカラム・キーとは異なり, カスタマ設定可能な特殊なカラム・キーが含まれています。すべてのカラム・キーでは大文字と小文字が区別されません。たとえば "ID", "Id", "id" はすべて ID 列にマップします。

カラム・キーの詳細については, オペレーション管理のオンライン・ヘルプを参照してください。

表示するカラムを定義する固定カラム・キー

カラム・キー	説明
annotations	注釈
application	アプリケーション
automaticAction	自動アクション
category	カテゴリ
ciType	CI タイプ
controlTransferred	イベントのコントロールが転送されているかどうかを表示します
correlation	相関処理(症状または要因)
description	説明
duplicateCount	重複数
eti	イベント・タイプ・インジケータ値
externalId	外部イベント ID

カラム・キー	説明
group	割り当てられたグループ
id	ID
node	ノード
nodeHint	ノードのヒント
object	オブジェクト
operatorAction	オペレータのアクション
originatingServer	起点サーバ
ownedInOM	HP Operations Manager(HPOM) で所有
priority	優先度
receivedOnCiDowntime	CI ダウンタイム中に受信
relatedCi	関連 CI
relatedCiHint	関連 CI のヒント
sendingServer	送信サーバ
severity	重大度
solution	ソリューション
sourceCi	ソース CI
sourceCiHint	ソース CI のヒント
state	ライフサイクル状態
subCategory	サブカテゴリ
timeCreated	イベントの作成時間
timeReceived	イベントの受信時間
timeStateChanged	イベントのライフサイクル状態の最終変更時間
title	タイトル
type	タイプ
user	割り当てられたユーザ

カスタム属性カラム・キー

カラム・キー	説明
ca_<*columnname*>	カスタム属性カラム。<*columnname*> はカスタム属性の名前です。 これは，[オペレーション管理 - カスタム属性設定]の[利用可能なカスタム属性]設定で設定する設定可能なカラム・キーです。詳細については，オペレーション管理のオンライン・ヘルプを参照してください。

フィルタの設定

イベント・ブラウザのイベント数をフィルタを使用して絞り込むことができます。フィルタは次の項目に指定できます。

- 文字列属性
- 時間プロパティ
- フラグ属性
- イベントの優先順位

246ページ「イベント・ブラウザの URL 起動用フィルタ・パラメータ」には，イベント・ブラウザの URL 起動に利用可能なフィルタ・パラメータが含まれています。

イベント・ブラウザの URL 起動用フィルタ・パラメータ

フィルタ・パラメータ	説明	値
`filter_assignment`	割り当てフィルタをイベント・ブラウザに適用します。	次のうち 1 つ以上： • `me` • `my_workgroups` • `others` • `nobody`
`filter_severities`	重大度フィルタをイベント・ブラウザに適用します。	次のうち 1 つ以上： • `unknown` • `normal` • `warning` • `minor` • `major` • `critical`
`filter_states`	ライフサイクル状態フィルタをイベント・ブラウザに適用します。	次のうち 1 つ以上： • `open` • `in_progress`, • `resolved`

文字列属性によるフィルタリング

イベントを文字列属性によってフィルタリングできます。

次の形式を使用して，文字列タイプのプロパティによってフィルタリングするためのフィルタを指定します。

`filter_<stringAttributeName>_<filterType>`

文字列属性名とフィルタ・タイプの可能な値を247ページ「文字列属性に可能なフィルタ・タイプと値」にリストします。

文字列属性に可能なフィルタ・タイプと値

可能な文字列属性名	application
	ca_<*customAttribute*>
	category
	correlationKey
	description
	object
	originalText
	relatedCiHint
	subCategory
	title
	type
可能なフィルタ・タイプ	contains
	equals
	isEmpty
	isNotEmpty
	notContains
	notEquals

文字列属性によるフィルタリングの例を次に示します。ここでは，`Network` というフィルタ・カテゴリを持つリターン・イベントにのみ関連しています。

```
filter_category_equals=Network
```

時間プロパティによるフィルタリング

イベントを時間プロパティによってフィルタリングすることもできます。

イベント・ブラウザで時間属性 `timeAttributeName` が `fromTime` と `toTime` の間にあるイベントのみ表示されるようにフィルタを設定できます。

次の形式を使用して，時間によってフィルタリングするためのフィルタを指定します。

`filter_<timeAttributeName>=fromTime-toTime`

説明：

`<timeAttributeName>` には，次のうちいずれかが可能です。`timeReceived`，`timeCreated`，`timeStateChanged`

時間を次の形式で指定する必要があります。

`<yyyyMMddHHmmss>`

説明：

- `yyyy` は年
- `MM` は月の値 (01～12)
- `dd` は日の値 (01～31)
- `HH` は時間の値 (00～23)
- `mm` は分の値 (00～59)
- `ss` は秒の値 (00～59)

優先順位によるフィルタリング

イベントを優先順位によってフィルタリングすることもできます。

イベント・ブラウザでイベントがその優先順位に従って表示されるようにフィルタを設定できます。

次の形式を使用して，優先順位によってフィルタリングするためのフィルタを指定します。

`filter_priorities=<priority_level>`

説明：

`<priority_level>` には，次のうち 1 つ以上 (カンマ区切りリスト) が可能です。`none`，`lowest`，`low`，`medium`，`high`，`highest` のいずれかの値に設定できます。

たとえば，イベント・ブラウザで優先順位が `highest` と `high` のイベントのみ表示する場合，フィルタを次のように指定します。

`filter_priorities=highest,high`

CI と CI タイプによるフィルタリング

イベント・ブラウザに指定の CI に関連するイベントのみ表示されるようにフィルタを設定できます。そのようなフィルタを指定するには，次の形式を使用します。

`filter_relatedCi_equals=<CI Id>`

説明：

`<CI Id>` は CI の ID です。

可能なフィルタ操作は次のとおりです。equals, isempty, notisempty

イベント・ブラウザで関連する CI の CI タイプが指定されている CI タイプに一致するイベントのみ表示されるようにフィルタを設定することもできます。そのようなフィルタを指定するには, 次の形式を使用します。

filter_ciType_<*operator*>=<*type*>

この場合, <*type*> は指定されている CI タイプで, <*operator*> には次の値のうちいずれかが可能です。equals, is_derived

グローバル CI ID によるフィルタリング

イベントをグローバル CI ID によってフィルタリングできます。グローバル CI ID は, コンテンツ管理システム(CMS) データベースなどの外部 (非 BSM) データベースの CI のグローバル ID です。

globalCiId パラメータの指定時に, ローカル(ODB) CI ID が検索され, その CI ID のフィルタが設定されます。

そのようなフィルタを指定するには, 次の形式を使用します。

filter_globalCiId_equals=<*global id*>

説明 :

<*global id*> は, CI のグローバル ID です。

可能なフィルタ操作は次のとおりです。equals, isempty, notisempty

フィルタ指定 filter_globalCiId_equals=<*global id*> により, フィルタ指定 filter_relatedCi_equals=<*CI Id*> と同じ結果が返されます。

ETI と ETI 値によるフィルタリング

イベント・ブラウザで指定の ETI に一致するイベントのみ表示されるようにフィルタを設定できます。

ETI によるフィルタを指定するには, 次の形式を使用します。

filter_eti_equals=<*ETI Id*>

説明 :

<*ETI Id*> は, ETI の UUID です。

可能なフィルタ操作は次のとおりです。equals, isempty, notisempty, isoneof

isoneof フィルタ操作の使用時は, 次のように ETI UUID のカンマ区切りリストが期待されます。

filter_eti_isoneof=<*ETI Id1,ETI Id2,ETI Id3,...*>

フィルタ操作 isempty と notisempty に対し, 引数は必要ありません。

イベント・ブラウザで特定の ETI 値を設定するイベントのみ表示されるようにフィルタを設定することもできます。そのようなフィルタを指定するには, 次の形式を使用します。

filter_etiValue_<*operator*>=<*ETI value Id*>

説明 :

<ETI value Id> は，ETI 値の UUID です。

可能なフィルタ操作は次のとおりです。equals, isempty, notisempty, isoneof

isoneof フィルタ操作の使用時は，次のように ETI 値 UUID のカンマ区切りリストが期待されます。

filter_etiValue_isoneof=<*ETI value Id1,ETI value Id2,ETI value Id3,...*>

フィルタ操作 isempty と notisempty に対し，引数は必要ありません。

ほかのイベント特性によるフィルタリング

同じ特性を共有するイベントをグループ化するためのフィルタを設定することもできます。たとえば，重複があるイベントのみを表示できます。

イベント特性によってグループ化するためにイベントをフィルタリングする場合，指定可能な多くのブール・フラグ属性があります。これらのブール・フラグ属性によりイベントで特定の特性があるかどうか，たとえばイベントに症状または注釈があるかどうかが指定されます。イベント・ブラウザで指定された特性のブール・フラグ属性に一致するイベントのみ表示されるようにフィルタを設定できます。

特定の特性を共有するイベントを表示するためのフィルタを設定するには，次の形式を使用してその特性に対するブール・フラグ属性を指定します。

filter_<*flagAttributeName*>

説明：

<*flagAttributeName*> には，次の値のうちいずれかが可能です。hasSymptoms, hasCause, hasDuplicates, hasAnnotations

特性の組み合わせを指定することもできます。次の例では，重複と症状があるイベントのみ表示するためにフィルタが設定されています。

http://<my.example.com:8080>/opr-console/opr-evt-browser?filter_hasDuplicates&filter_hasSymptoms

イベント詳細の URL 起動

URL を使用したイベント・ブラウザの起動と似た方法で，次の URL を使用してイベント詳細を起動することもできます。

http://<ホスト名：ポート>/opr-console/opr-evt-details?eventId=<イベント ID>

説明：

<イベント ID> は，イベント詳細で表示するイベントの ID です。

イベント詳細の直接起動用の URL の例を次に示します。パラメータ eventId が表示するイベントの ID に設定されます。

http://my.example.com:8080/opr-console/opr-evt-details?eventId=e004e66b-cada-407f-84ac-32f2d613eec4

第12章

オペレータ機能およびイベント変更検出の自動化

本項では，オペレータ機能の自動化およびイベント変更の検出をプログラミングする際にインテグレータにとって役に立つ情報を示します。イベントの作業時にオペレータがコンソールで行うほとんどの操作をプログラミングでき，効率性の向上を実現できます。

イベント同期に向けたアプリケーションの統合に関する詳細については，304ページ「イベントの転送およびイベント変更の同期」を参照してください。

本項の内容

イベント Web サービス・インタフェースを使用したオペレータ機能の自動化

インテグレータには，イベントをほかのアプリケーションに統合するためのインタフェースが提供されています。このインタフェースを使用すると，オペレータ機能の自動化およびイベント変更の検出をプログラミングできます。イベントの作業時にコンソールで行うほとんどの操作をプログラミングでき，効率性の向上および外部アプリケーションとの統合を実現できます。

イベントをほかのアプリケーションに統合し，オペレータ機能を自動化するインタフェースはイベント Web サービスです。これは REST ベースの Web サービスで，イベント・モデル・サポートおよび Atom フィード機能を介したサブスクリプション・サポートを提供します。イベントのリストが表示される Atom フィードをブラウザで読んだり，Atom サービスを使用してイベントを作成および更新することもできます。

イベント Web サービス・データには，イベント・ブラウザを起動するための URL へのリンクが含まれています(外部アプリケーションからオペレーション管理ユーザ・インタフェースにドリルダウンするため) 。ドリルダウン URL の指定および起動方法に関する詳細については，241ページ「イベント・ブラウザの URL 起動」を参照してください。

通常，インテグレータは次の項目に関心があります。

- 252ページ「イベント Web サービスにアクセスする方法」
- 255ページ「新規イベントを検出する方法」
- 256ページ「イベント変更の検出方法」
- 257ページ「イベントの変更方法」
- 262ページ「新規イベントの作成方法」
- 266ページ「イベント Web サービスのセキュリティ」

イベント Web サービスにアクセスする方法

イベント Web サービス・インターフェイスへのエントリ・ポイントは，次のベース URL を使用してアクセスできるサービス・ドキュメントです。

- 標準環境:

  `http://<bsmserver.example.com>/opr-console/rest/`

- セキュア保護された環境:

  `https://<bsmserver.example.com>/opr-console/rest/`

説明:

<bsmserver.example.com> は，ゲートウェイ・サーバの名前です。

イベント Web サービスにアクセスするには，有効なユーザが必要です。さらに，ユーザ認証の資格情報としてユーザ名およびパスワードを提供する必要があります。許可されたユーザのみがイベントの表示，イベントの変更，アクションの実行が可能です。

サービス・ドキュメントには，異なる OPR イベント・サービスの URL が一覧表示されます。これらのサービスの一覧については，253ページ「サービス・ドキュメントの OPR イベント・サービスのリスト」を参照してください。

サービス・ドキュメントの URL のリストでは，2 つの URL のみが実際のリンクです。

- イベント(複数)サービス:

  `http://<bsmserver.example.com>/opr-console/rest/<version>/event_list`

  `<version>` は，BSM リリースのバージョンです(たとえば `9.10`)。バージョン番号が省略されている場合，9.10 未満のバージョンがアドレスされます。

  これにより，すべてのイベントの一覧が表示されます。

- イベント変更(複数)サービス:

  `http://<bsmserver.example.com>/opr-console/rest/event_change_list`

  これにより，イベントへの変更の一覧が表示されます。

その他すべての URL では，パラメータを指定する必要があります(たとえば，イベント ID)。注釈の場合，注釈が必要なイベントに対して，注釈 ID を指定する必要があります。253ページ「サービス・ドキュメントの OPR イベント・サービスのリスト」では，{event}，{annotation} または {custom_attribute} など，各 URL に対して指定する必要のある変数が波括弧で閉じられた状態で示されています。`532d3674-684f-419f-a752-b8681ee01a72` という ID のイベントを指定する例の URL を次に示します。

```
http://<bsmserver.example.com>/opr-console/rest/9.1/event_
list/532d3674-684f-419f-a752-b8681ee01a72
```

サービス・ドキュメントの OPR イベント・サービスのリスト

OPR イベント・サービス	URL
注釈サービス:	`http://<bsmserver.example.com>/opr-console/rest/9.10/event_list/{event}/annotation_list/{annotation}`
注釈サービス:	`http://<bsmserver.example.com>/opr-console/rest/9.10/event_list/{event}/annotation_list`
自動アクション・サービス	`http://<bsmserver.example.com>/opr-console/rest/9.10/event_list/{event}/auto_action`
カスタム属性サービス:	`http://<bsmserver.example.com>/opr-console/rest/9.10/event_list/{event}/custom_attribute_list/{custom_attribute}`
カスタム属性(複数)サービス	`http://<bsmserver.example.com>/opr-console/rest/9.10/event_list/{event}/custom_attribute_list`
イベント変更サービス	`http://<bsmserver.example.com>/opr-console/rest/event_change_list/{event_change}`

OPR イベント・サービス	URL
イベント変更(複数)サービス	http://<*bsmserver.example.com*>/opr-console/rest/event_change_list
イベント・サービス	http://<*bsmserver.example.com*>/opr-console/rest/9.10/event_list/{event}
イベント(複数)サービス	http://<bsmserver.example.com>/opr-console/rest/9.10/event_list
履歴行サービス	http://<*bsmserver.example.com*>/opr-console/rest/9.10/event_list/{event}/history_line_list/{history_line}
履歴行(複数)サービス	http://<*bsmserver.example.com*>/opr-console/rest/9.10/event_list/{event}/history_line_list
オペレータ・アクション・サービス	http://<*bsmserver.example.com*>/opr-console/rest/9.10/event_list/{event}/user_action
症状サービス	http://<*bsmserver.example.com*>/opr-console/rest/9.10/event_list/{event}/symptom_list/{symptom}
症状サービス	http://<*bsmserver.example.com*>/opr-console/rest/9.10/event_list/{event}/symptom_list

一般的に，ほとんどのOPR イベント・サービスでは，サポートされる4つの操作(読み取り，作成，更新，削除)を実行できます。次の例外があります。

- 自動アクション・サービスおよびオペレータ・アクション・サービス。これらのサービスでは，次の操作を実行できます。
  - 読み取り操作を実行してアクションの状態を取得する。
  - 作成操作を実行してアクションを開始する(まだ実行されていない場合)。
  - 削除操作を実行してアクションを停止する(既に実行中の場合)。
- 読み取り操作のみ実行可能なサービス:
  - イベント変更サービス
  - イベント変更(複数)サービス
  - 履歴行サービス
  - 履歴行(複数)サービス

イベント変更とイベント変更(複数)という2つのサービスは，履歴行および履歴行(複数)サービスに非常に相似する結果を返すように見えます。しかし，イベント変更サービスは，すべてのイベント変更をリストします。特定のイベントに固有のサービスではありません。イベント変更(複数)サービスは，イベント固有であり，特定のイベントに関連するイベントの変更をリストします。履歴行(複数)に関する詳細については，300ページ「履歴行」を参照してください。

新規イベントを検出する方法

すべてのイベントのリストを返すには，サービス・ドキュメントでイベント・リストの URL をクリックします。

```
http://<bsmserver.example.com>/opr-console/rest/9.10/event_list
```

デフォルトでは，この URL をクリックすると，XML 形式でイベント・リストが開かれます。HTTP クエリ・パラメータをこのベース URL に追加して，リスト内のデータの表示方法を変更することができます。

- `alt=atom`:このパラメータは，イベントのリストを Atom フィード形式で表示します。イベント・データの表示方法は，メタデータ(たとえば，カテゴリ，作成者など)によって決定します。`alt` メディア タイプの詳細については，288ページ「メディア・タイプ」を参照してください。
- `alt=json`:このパラメータは，イベントのリストを JSON 形式で表示します。
- `alt=xml`:このパラメータは，イベントのリストを XML 形式で表示します。これは Web ブラウザの標準設定です。

イベントを Atom フィードとして受信する

ほとんどのブラウザは，Atom および RSS フィードの非常に基本的な内部フィード・リーダーを備えています。フィード・リーダーの機能およびデータの表現方法はそれぞれ異なるため，同じデータがリーダーによって異なる方法で表示される場合があります(通常，データのアクセス方法も異なります)。

フィード フィーダーを備えた最も一般的に使用されるブラウザは Mozilla Firefox および Microsoft Internet Explorer です。以下の例では，Atom Web サービスがこれらのブラウザでどのように機能するかを示します。

Atom フィードでのイベント・リストは，「前回の変更」に基づいて順序設定されます。したがって，新規イベントまたは前回変更されたイベントがリストの一番上に表示されます。

イベントのリストを Atom フィード形式で返すには，次の手順を実行します。

1. イベント・リストの URL をご使用の Web ブラウザに入力します。その際，`alt=atom` パラメータを次のように指定します。

   ```
   http://<bsmserver.example.com>/opr-console/rest/9.10/event_
   list/?alt=atom
   ```

2. イベントのリストが表示されます。Firefox と Internet Explorer では，多少異なる方法でデータが表示されます。

   - **Firefox の場合**:

     イベントのリストが各イベントのタイトルと説明とともに表示されます。

     Atom フィード内でイベントのタイトルをクリックすると，そのイベントのイベント詳細の URL 起動が開始されます。イベント・ブラウザおよびイベント詳細の URL 起動に関する詳細については，241ページ「イベント・ブラウザの URL 起動」を参照してください。

     > **注:** より詳細なイベント・リストを表示するには，ページを右クリックし，[**View Page Source**]を選択します。この操作により Atom フィード全体が XML 形式で表示されます。これで，リストに含まれるイベントに関するその他すべてのプロパティおよび情報を表示することができます。

- **Internet Explorer の場合**:

  イベントのリストが各イベントのタイトルと説明とともに表示されます。リストに加え，リストを日付およびタイトルでソートできるフィルタ ボックスが右側に表示されます。特定のカテゴリでイベントをフィルタリングすることもできます。

  Atom フィード内でイベントのタイトルをクリックすると，オペレーション管理 ユーザ・インターフェイスにドリルダウンするイベント詳細の URL 起動が開始されます。イベント・ブラウザおよびイベント詳細の URL 起動に関する詳細については，241ページ「イベント・ブラウザの URL 起動」を参照してください。

  イベントの基本的な詳細しか表示されません。詳細すべてを表示するには，ページを右クリックし，[**ソースの表示**]を選択します。この操作によって，ページが XML エディタで開かれます。

パラメータを指定してイベント・リストをフィルタリングする

イベント Web サービスでは，特定の条件でイベント・リストをフィルタリングするための数多くのパラメータが提供されています。

たとえば，特定のイベント・パラメータを URL クエリ・パラメータの条件と一致させて結果をフィルタリングすることができます。表示するイベントの数を制限することでリストのサイズを削減し，リストを複数のページに分割するように選択することもできます。標準設定値のページ・サイズは 20 です。

もう 1 つの例として，特定の時間後に作成または更新されたイベントのみを一覧表示する時間パラメータを指定する方法が挙げられます。Web サービスを使用して新規イベントおよび変更されたイベントを監視するアプリケーションのシャットダウン時に最後のイベントの変更時間が記憶されます。この方法では，このタイムスタンプ後に作成された新規イベントや変更されたイベントのみに対するクエリを実行することができます。

特定の日付後のイベントのリストを返す場合は，`watermark` パラメータを指定できます。

また，`sequence_number` パラメータを指定すると，特定のシーケンス番号以降のイベントのリストを取得することもできます。

次の例の URL では，2010 年 3 月 4 日 15:59:17 後の現在開かれている最初の 40 イベントが Atom フィード形式で返されます。

```
http://<bsmserver.example.com>/opr-console/rest/9.10/event_
list/?alt=atom&watermark=2010-03-04T15:59:17%2B02:00&page_
size=40&start_index=1
```

URL クエリ言語，クエリ・フィルタ，パラメータの詳細については，283ページ「イベント Web サービス・クエリ言語」を参照してください。

イベント変更の検出方法

イベント変更のリストを返すには，サービス・ドキュメントでイベント変更リストの URL をクリックします。

```
http://<bsmserver.example.com>/opr-console/rest/event_change_list
```

前回イベント・リストを取得した後のすべてのイベント変更のリストを返すこともできます。次の例の URL では，2010 年 3 月 10 日 12:29:54 後のすべてのイベント変更が Atom フィード形式で返されます。

```
http://<bsmserver.example.com>/opr-console/rest/event_change_
list?alt=atom&watermark=2010-03-10T15:59:17%2B01:00
```

イベントの変更方法

Web ブラウザで REST クライアントを使用するか，`RestWsUtil` コマンドライン・ユーティリティを使用すると，イベントを変更(項目の読み取り，作成，更新，削除)できます。

注: 本項では，標準 HTTP 環境について説明します。セキュア環境については，HTTPS を使用してください。

注: BSM 9.10 およびそれ以上では，イベント Web サービスの変更操作は，変更要求(PUT，POST および DELETE)で `X-Secure-Modify-Token` HTTP ヘッダを設定してセキュリティ保護する必要があります。このヘッダは Web アプリケーションの悪意のある検索に対する強力なセキュリティ保護を提供します。詳細は，266ページ「イベント Web サービスのセキュリティ」を参照してください。

REST クライアントを使用したイベントの変更

通常，REST クライアントを使用してイベントを更新する場合は，次の操作を実行します。

- 所定のイベントの URL を入力して REST サービスを呼び出します(URL ではイベント ID を指定する必要があります)。
- HTTP GET 要求を使用してイベントを取得します。
- 変更したい XML ドキュメントの要素を編集します。
- HTTP PUT 要求を使用してイベントの変更部分を戻します。
- XML を再ロードして変更部分を表示します。

イベントの変更で使用できる REST クライアントには，RESTClient および Mozilla Firefox Poster Extension があります。

RESTClient を使用したイベントの変更

ここでは，RESTClient を使用してイベントを変更する方法について説明します。RESTClient は，オープン・ソースのダウンロードとして利用可能です。RESTClient をダウンロードするには，次の場所で情報を参照してください。

```
http://code.google.com/p/rest-client/
```

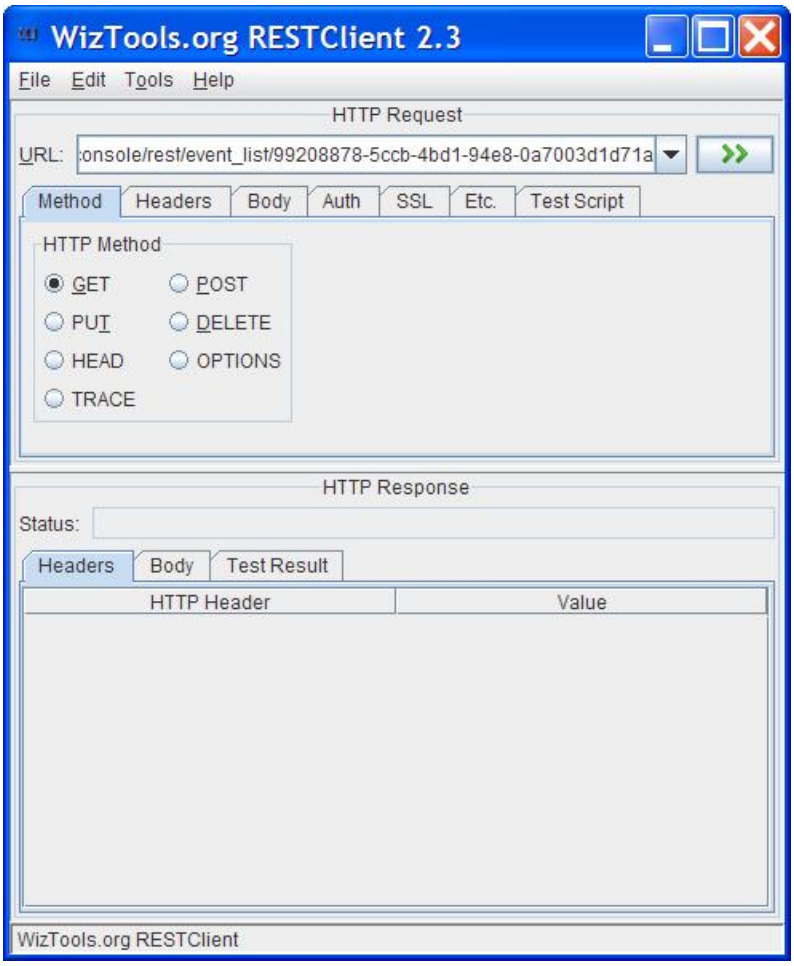
RESTClient ユーザ・インタフェースは 2 つのパートに分割されています。

- HTTP 要求 :ユーザ・インタフェースの上半分。変更するイベントの URL を入力して，HTTP GET 要求でイベントを取得し，HTTP PUT 要求で変更した XML を送信するするために使用されます。
- HTTP 応答 :ユーザ・インタフェースの下半分。応答を返すために使用されます。

ここでは，RESTClient を使用してイベントのタイトルおよび重大度を変更する方法の例を示します。必要な手順は次のとおりです。

1. イベント・リスト・フィードから変更するイベントのイベント ID を取得します。
2. RESTClient ユーザ・インタフェースの[URL]フィールドに変更するイベントの URL(イベント ID の指定) を入力します。次の構文を使用します。

   ```
   http://<bsmserver.example.com>/opr-console/rest/9.10/event_
   list/<event_ID>
   ```



3. [HTTP Method]ボックスの[**GET**]ラジオ・ボタンを選択して，  ボタンをクリックします。
4. 下部の[HTTP Response]セクションに結果が表示されます。[**Body**]タブをクリックして，編集するイベントの応答 XML を読み取ります。
5. XML をコピーします。上部の[HTTP Request]セクションの[**Body**]タブをクリックし，XML をテキスト・ボックスに貼り付けます。
6. 変更するイベント・プロパティを編集して，イベントを変更します。完全な XML をサーバに戻す必要はありません。XML 構造が保持されていれば，イベントに対して更新した XML 要素のみを送信するように選択することができます。
7. 変更が完了したら，[**Method**]タブを再び選択し，[**PUT**]ラジオ・ボタンをクリックします。

8.  ボタンをクリックして，変更を送信します。変更が適用されると，HTTP `200 OK` メッセージが応答領域に表示されます。Atom フィードを確認して，実行した変更を検証します。

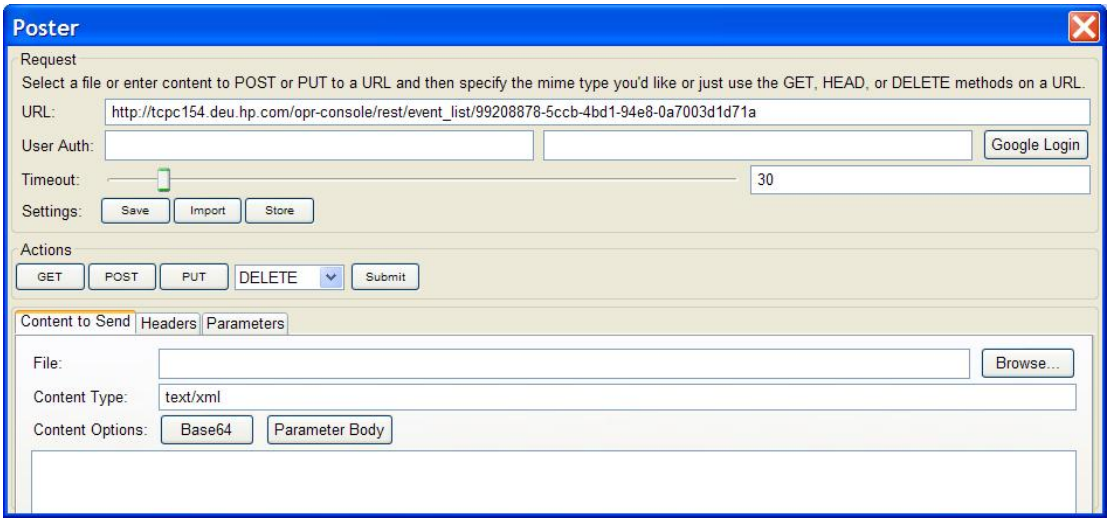
Firefox Poster Extension を使用したイベントの変更

REST クライアントを使用してイベントを変更する方法を説明するために，ここでは，Mozilla Firefox Poster Extension を使用してイベントを変更する方法について説明します。Poster Extension は，シンプルな REST クライアントで，最初に Firefox のプラグインとしてインストールする必要があります。

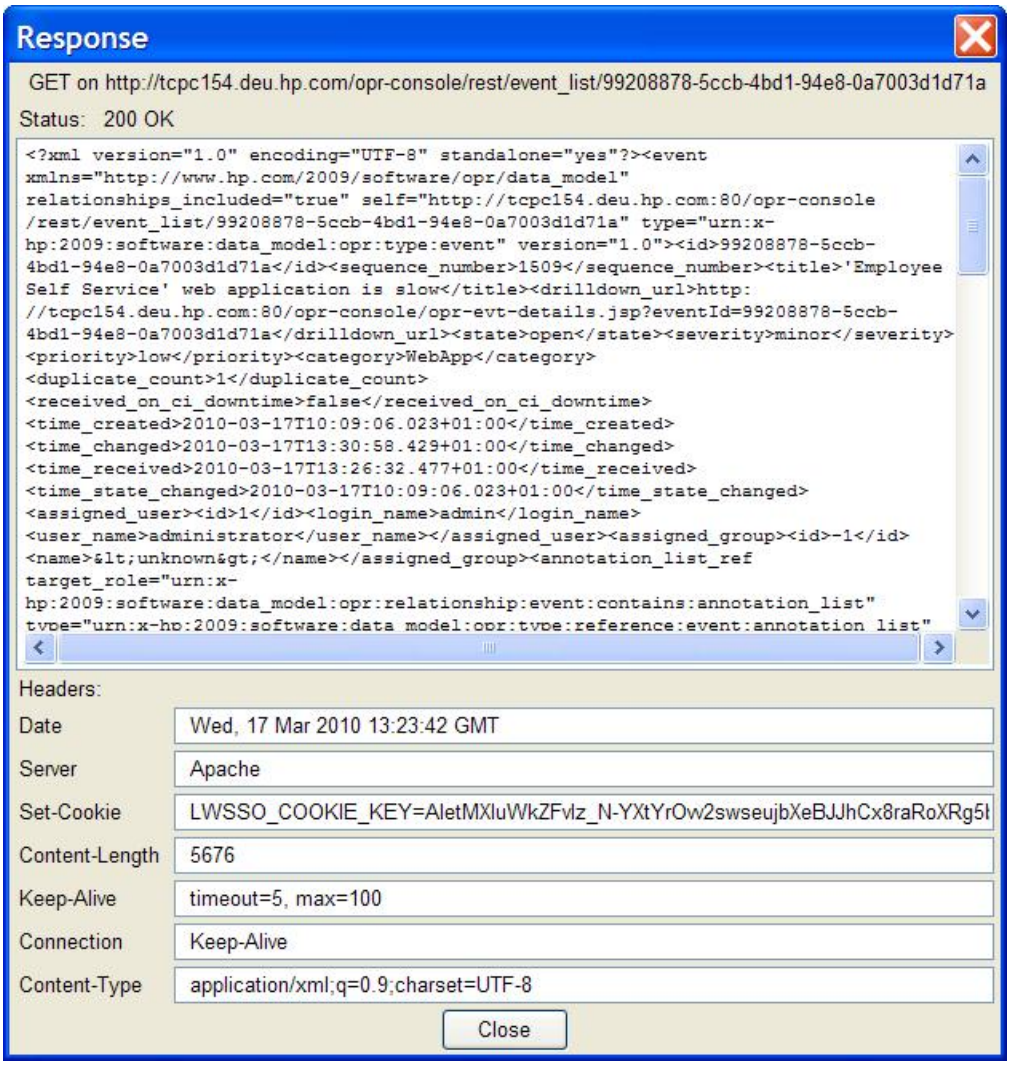
Firefox Poster Extension を使用してイベントのタイトルおよび重大度を変更する方法の例を示します。必要な手順は次のとおりです。

1. 最初に Mozilla Firefox Poster Extension をインストールします(まだインストールされていない場合) 。
2. Firefox ブラウザを開き，[**ツール**] > [**Poster**] の順に選択します。[Poster] ダイアログ・ボックスが表示されます。
3. イベント・リスト・フィードから変更するイベントのイベント ID を取得します。
4. [Poster] ダイアログ・ボックスの URL フィールドに，変更するイベントのイベント ID を含む URL を入力します。次の構文を使用します。

   ```
   http://<bsmserver.example.com>/opr-console/rest/9.10/event_
   list/<event_ID>
   ```



5. [**GET**] をクリックして，イベントを XML として受信します。[Response] という名前のウィンドウが開きます。エラーが発生していない場合，状態が `200 OK` になります。また，変更するイベントの完全な XML が表示されます。



6. 完全な XML を[Response]ウィンドウからコピーし，[Poster]ダイアログ・ボックスの[Content to Send]タブにある内容テキスト・フィールドに貼り付けます。[Response]ウィンドウは使用しないので，ここで閉じます。

7. 次に，イベントに対して実行する変更に基づいて XML を編集できます。完全な XML をサーバに戻す必要はありません。XML 構造が保持されていれば，イベントに対して更新した XML 要素のみを送信するように選択することができます。

> **注:** すべてのプロパティを更新できるわけではありません。編集可能なプロパティのリストについては，298ページ「編集可能なプロパティ」を参照してください。最新の Java API ドキュメントも確認してください。このドキュメントは，次の場所で入手できる zip ファイルに含まれています。
>
> `<HPBSM のルート・ディレクトリ>/opr/api/doc/opr-external-api-javadoc.zip`
>
> この zip ファイルの内容を任意の場所で解凍して，API ドキュメントを参照してください。

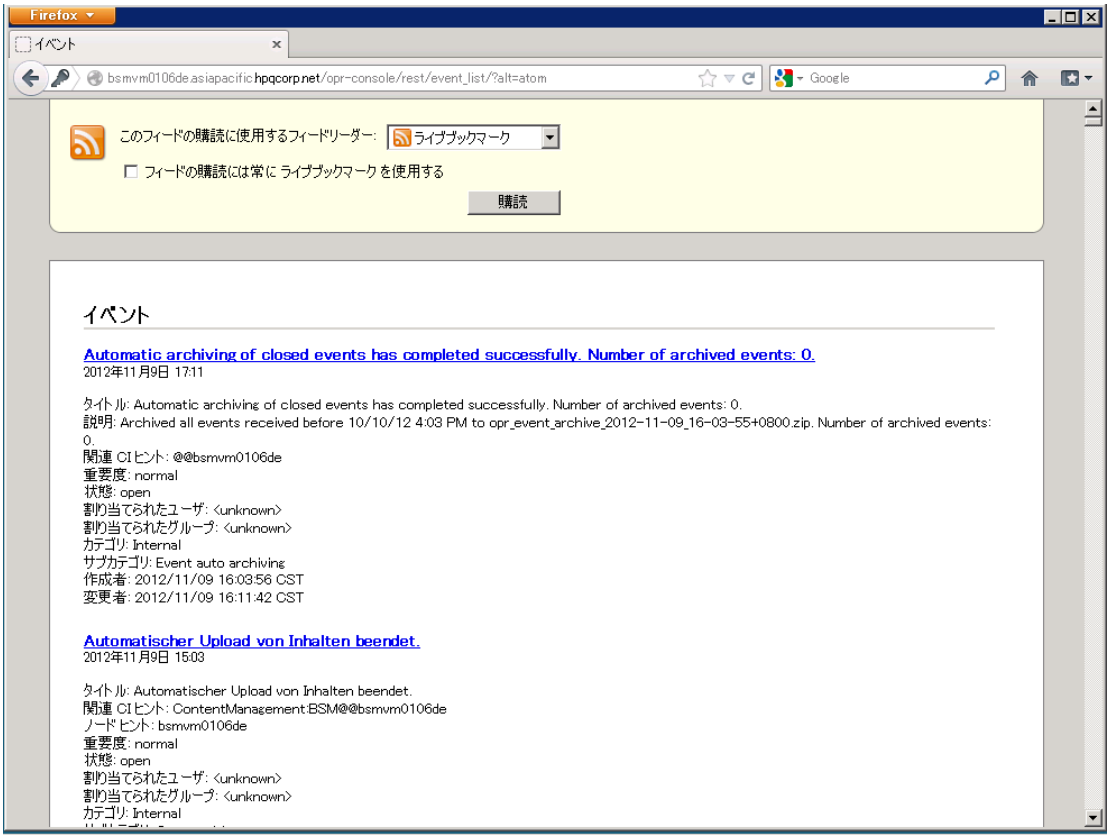
この例では，イベントのタイトルおよび重大度を直接 XML で変更します。たとえば，イベントの元のタイトルが「'Employee Self Service' web application is slow」であると仮定します。「very」という単語を挿入して，イベントのタイトルを編集します。

```
...<title>'Employee Self Service' web application is very
slow</title>...
```

同様に，イベントの重大度を警戒域から注意域に変更します。

```
...<severity>warning</severity>...
```

8. イベントに対する変更を実行したら，[Content Type]フィールドが **application/xml** に設定されていることを確認します(入力が必要)。[**PUT**]をクリックして変更を保存します。[Response]ウィンドウが開きます。エラーが発生していない場合，HTTP `200 OK` メッセージが表示されます。Atom フィードを確認して，実行した変更を検証します。



Firefox Poster Extension についての追加情報

Firefox Poster Extension についての追加情報は，次の場所で参照できます。

`http://code.google.com/p/poster-extension/`

RestWsUtil ユーティリティを使用したイベントの変更

REST Web サービス・ユーティリティを使用すると，コマンドラインからイベント Web サービスに対して REST Web サービス操作を実行できます。このユーティリティを使用すると，次の 4 つの REST Web

サービス操作のいずれかを実行できます。

CRUD 操作	HTTP メソッド
作成	POST
読み取り	GET
更新	PUT
削除	DELETE

このユーティリティは次の場所にあります。

<HPBSM のルート・ディレクトリ>/opr/bin

RestWsUtil コマンドライン・ユーティリティを使用して，イベントを変更できます。

このユーティリティを使用してイベントを変更する方法を示す非常に簡単な例を次に示します。

例 :イベントのタイトルを変更する方法

RestWsUtil ユーティリティを使用してイベントのタイトルを変更するには，次の手順を実行します。

1. 変更するイベントのイベント ID を取得します。
2. 変更するイベントの XML を **<HPBSM のルート・ディレクトリ>/opr/bin** ディレクトリの XML ファイル (たとえば update.xml) に書き込みます。

   update.xml ファイルの内容は次のようになります。

   ```
   <event xmlns="http://www.hp.com/2009/software/opr/data_
   model"><title>
   New title goes here</title></event>
   ```

   タイトルを編集します。
3. 変更を更新および保存するには，コマンド・プロンプトに次のコマンドを入力します。

   <HPBSM のルート・ディレクトリ>/opr/bin>RestWsUtil -update update.xml -username <ログイン名> -password <パスワード> -url http://<bsmserver.example.com>/opr-console/rest/9.10/event_list/<event_ID>

   説明：

   <HPBSM のルート・ディレクトリ> は，BSM がインストールされているディレクトリ，<ログイン名> は認証に必要なユーザ名，<event_ID> は変更するイベントのイベント ID です。

RestWsUtil コマンドライン・ユーティリティの使用方法に関する詳細および例については，274ページ「REST Web サービス・コマンドライン・ユーティリティ」を参照してください。

新規イベントの作成方法

RestWsUtil コマンドライン・ユーティリティを使用すると，新規イベントを作成できます。

注: イベントを作成するには，BSM ユーザ管理設定でイベントの送信権限が付与されている必要があります。この操作方法に関する詳細については，HP ビジネス・サービス管理のオンライン ヘルプを参照してください。

例 :新規イベントを作成する方法

`RestWsUtil` ユーティリティを使用して新規イベントを作成するには，次の手順を実行します。

1. 次のフォルダに XML ファイルを作成します(たとえば，`create.xml` という名前をつけます) 。

   <HPBSM root directory>/opr/bin/create.xml

2. ファイルに次の行を追加して，新規イベントのプロパティを定義します。

   ```
   <event xmlns="http://www.hp.com/2009/software/opr/data_model">
      <title>New event title</title>    <severity>normal</severity>
      <priority>low</priority>    <state>open</state> </event>
   ```

3. コマンド・プロンプトを開いて，次を入力します。

   <HPBSM root directory>/opr/bin>RestWsUtil -create create.xml -username *<login name>* -password *<password>* -url http://<bsmserver.example.com>/opr-console/rest/9.10/event_list

   説明：

   <HPBSM root directory> は，BSM がインストールされているディレクトリです。*<login name>* は，認証に必要なユーザ名です。

`RestWsUtil` コマンドライン・ユーティリティの使用方法に関する詳細および例については，274ページ「REST Web サービス・コマンドライン・ユーティリティ」を参照してください。

イベント・プロパティの高度な変更

イベントには，252ページ「イベント Web サービスにアクセスする方法」の項に記載されているように，サービス・ドキュメントに一覧表示されている OPR イベント・サービスによって変更可能な選択した数のリスト・プロパティが含まれます。

指定したイベントのカスタム属性プロパティの XML リストを生成するには，次の URL を呼び出します。

http://*<bsmserver.example.com>*/opr-console/rest/9.10/event_list/*<event_ID>*/custom-attribute-list/

説明：

<bsmserver.example.com> はゲートウェイ・サーバの名前で，*<event_ID>* は，カスタム属性を一覧表示する対象のイベントの ID です。

このような URL 呼び出しに対する REST 応答 (この場合では ID 26629e00-2d8d-71dd-1aa2-1039228c0111 が付いたイベントに対するもの) は，次のようになります。

```
<custom_attribute_list
xmlns="http://www.hp.com/2009/software/opr/data_model"
```

```
self="http://<bsmserver.example.com>/ws/rest/event_list/26629e00-2d8d-
71dd-1aa2-1039228c0111/custom_attribute_list"       type="urn:x-
hp:2009:software:data_model:opr:type:event:custom_attribute_list"
version="1.0"> <custom_attribute
self="http://<bsmserver.example.com>/ws/rest/event_list/26629e00-2d8d-
71dd-1aa2-1039228c0111/custom_attribute_list/drilldown.url.ci"
type="urn:x-hp:2009:software:data_model:opr:type:event:custom_
attribute" version="1.0"> <name>drilldown.url.ci</name>
<value>http://url.to/drill/down/ci</value> </custom_attribute>
<custom_attribute self="http://<bsmserver.example.com>/ws/rest/event_
list/26629e00-2d8d-71dd-1aa2-1039228c0111/custom-attribute-
list/drilldown.url.event" type="urn:x-hp:2009:software:data_
model:opr:type:event:custom_attribute" version="1.0">
<name>drilldown.url.event</name>
<value>http://url.to/drill/down/event</value> </custom_attribute>
</custom_attribute_list>
```

イベント Web サービスを使用して，一覧の項目を作成および編集できます。また，カスタム属性一覧から項目を削除することも可能です。

注: RestWsUtil コマンドライン・ユーティリティを使用してイベント自体を作成したり削除することはできません。

指定のイベントに対するカスタム属性の項目を作成するには，該当する XML オブジェクトとともに HTTP POST 要求をイベントのカスタム属性一覧の URL に送ります。指定したイベントに対してカスタム属性一覧の URL を呼び出すと，そのイベントのカスタム属性の一覧に新しい項目が追加されていることが確認できます。

カスタム属性の一覧内の項目を編集するには，次のような類似の操作を行います。既存の項目名とその項目の変更された値を指定している HTTP PUT 要求を送信します。イベント Web サービスによって，その値が新しい値に更新されます。

HTTP 応答は次のようになります。

```
<custom_attribute xmlns="http://www.hp.com/2009/software/opr/data_
model" self="http://<bsmserver.example.com>/ws/rest/event_
list/26629e00-2d8d-71dd-1aa2-1039228c0111/custom_attribute_
list/mynewattribute" type="urn:x-hp:2009:software:data_
model:opr:type:event:custom_attribute" version="1.0">
<name>mynewattribute</name> <value>Hello</value> </custom_attribute>
```

同様に，カスタム属性の一覧から項目を削除するには，HTTP DELETE 要求を次の URL に送信します。

```
http://<bsmserver.example.com>/opr-console/rest/9.10/event_
list/<event_ID>/custom_attribute_list/<custom_attribute_name>
```

説明：

<*custom_attribute_name*> は削除用として選択したカスタム属性の名前です。

イベントの一括更新

特定のイベント ID を参照してイベントを個別に更新することに加えて，イベントを一括で更新することが可能です。これは特定のイベントのアドレスを指定するのではなく，クエリでイベント一覧のアドレスを指定することによって(283ページ「イベント Web サービス・クエリ言語」を参照)，および更新を PUT 要求のペイロード内で更新を指定することによって行われます。.

たとえば，

URL:`http://my.host.com/opr-console/rest/9.10/event-list&query=title%20LIKE%20'%25db down%25'`

HTTP メソッド :PUT

ペイロード : `<event xmlns="http://www.hp.com/2009/software/opr/data_model"> <severity>major</severity> </event>`

上記の要求によって，タイトルに `db down` が付いているすべてのイベントの重大度が `major` に設定されます。

HTTP メソッド :PUT

ペイロード : `<event xmlns="http://www.hp.com/2009/software/opr/data_model"> <state>closed</state> </event>`

上記の要求によって，タイトルに `db down` が付いているすべてのイベントは閉じます。

イベントの一括挿入

イベントを個別に挿入することに加えて，イベントの一覧を 1 つの Web サービス・コールで挿入することが可能です。これは，次のプロパティを持つ POST 要求を実行することで行われます。

- 単なる 1 つのイベントというのではない，`event_list` の例は次のとおりです :

```
<event_list xmlns="http://www.hp.com/2009/software/opr/data_model">
   <event>
      <title>Major Event</title>
      <severity>major</severity>
   </event>
   <event>
      <title>Minor Event</title>
      <severity>minor</severity>
   </event>
</event_list>
```

- `"application/xml; type=collection"` または `"text/xml; type=collection"` のコンテンツタイプ。

  `RestWsUtil` ユーティリティを使用する場合，`-content_type` オプションを使用してコンテンツ・タイプを指定できます。

イベント Web サービスのセキュリティ

イベント Web サービスにアクセスするには，有効なユーザであり，ユーザ名とパスワードをユーザ認証の資格情報として提供できる必要があります。許可されたユーザのみがイベントの表示，イベントの変更，アクションの実行が可能です。

イベント Web サービスのクライアントは，`X-Secure-Modify-Token` HTTP ヘッダを設定して変更操作 (PUT，POST，および DELETE) のセキュリティを確保する必要があります。このヘッダは Web アプリケーションの悪意のある検索に対する強力なセキュリティ保護を提供します。詳細は，267ページ「変更操作のセキュリティ保護」を参照してください。

CA SiteMinder Web エージェントが `CssChecking=YES` のように設定されている場合，CA SiteMinder Web エージェント `BadUrlChars` パラメータで設定された文字が CA SiteMinder Web エージェントによって拒否されます。これらの文字を URL で使われないようにする方法の詳細については，272ページ「CA SiteMinder を使用した環境」を参照してください。

エラーの詳細度の変更

管理者は，イベント Web サービスが返すエラー・コードを使用して問題の原因を特定することができます。管理者は，インフラストラクチャ設定マネージャで詳細レベルを変更することで，イベント Web サービスから返されたエラー・コードのタイプを制御できます。次のオプションを利用できます。

オプション	説明
`Standard`	標準設定値です。HTTP 1.1 標準にリストされている適切な HTTP エラー・コードが，エラーを記述する標準エラー・テキスト・メッセージとともに Web サービスの呼び出し側に返されます。
`Verbose`	開発環境に推奨されます。エラーの原因を記述する詳細メッセージが返されます。
`Brief`	エラーのタイプに基づきエラー・コード 400(Bad Request) または 503 (Service Unavailable) およびエラー識別子のみが返されます。詳細なエラー・メッセージは，エラー・ログの ID と詳細なエラー・メッセージを検索することにより，取得できます。

1. インフラストラクチャ設定マネージャの Web サービス設定に移動します。

   [**インフラストラクチャ設定**] > [**アプリケーション**] > [**オペレーション管理**] > [**オペレーション管理 - Web サービス設定**] > [**エラー応答の詳細**]

2. *オプション*。標準設定値の **Standard** を別の値に変更します。

3. BSM ゲートウェイ・サーバでエラー・ログ・ファイルを検証します。

   `<HPBSM のルート・ディレクトリ>/log/opr-ws-response.log`

変更操作のセキュリティ保護

BSM 9.10 およびそれ以上では，イベント Web サービスの変更操作は，変更要求(PUT，POST および DELETE) で `X-Secure-Modify-Token` HTTP ヘッダを設定してセキュリティ保護する必要があります。このヘッダは Web アプリケーションの悪意のある検索に対する強力なセキュリティ保護を提供します。変更操作の保護強化は，インフラストラクチャ設定マネージャの Web サービス設定で標準で有効にしています。下位互換性のためにこの設定を無効にすることができます(272ページ「強化されたセキュリティ保護の無効化」を参照)。

X-Secure-Modify-Token HTTP ヘッダの設定

イベント Web サービス・クライアントは，最初に `secureModifyToken` クッキーを取得し，`X-Secure-Modify-Token` HTTP ヘッダでクッキーの値を設定する必要があります。

1. 変更要求(PUT，POST または DELETE) を実行する前に `secureModifyToken` クッキーを取得します。

   クライアント起動時に `secureModifyToken` クッキーを取得するための推奨されるアプローチとして，`/opr-console/rest` でのイベント Web サービスのサービス・ドキュメントに対して HTTP GET 要求を実行します。

   HTTP GET 操作が完了すると，クッキーが設定されます。クライアントの有効期間およびシングル・サインオン・セッションでは，クッキーの値が変更される場合があります。変更操作を実行する前に，HTTP クライアントは，クライアントから現在のクッキーの値を取得する必要があります。この値は HTTP クライアントの有効期間中に変更される可能性があるため，後で再利用するためにローカル変数に保存することは推奨されません。

2. `X-Secure-Modify-Token` HTTP ヘッダを設定します。

   `X-Secure-Modify-Token HTTP` ヘッダは，変更操作(PUT，POST または DELETE) の実行時にすべてのイベント Web サービス・クライアントによって設定される必要があります。この HTTP ヘッダの設定先の値は，`secureModifyToken` クッキーに指定されます。

注: クライアントはサーバによって返されるすべてのクッキーを後続の要求で設定する必要があります。たとえば，`LWSSO_COOKIE_KEY` および `JSESSIONID` はサーバによって返される追加のクッキーで，そのサーバに対する後続の要求で設定する必要があります。新規セッションが確立すると，GET 要求を介して以前取得された `secureModifyToken` が無効になります。したがって，その他のクッキーも必要になります。

標準的な Java HTTP クライアントを使用した場合のサンプル・コード

次のサンプル・コードでは，最初に `secureModifyToken` クッキーが取得され，次に `X-Secure-Modify-Token` HTTP ヘッダが設定されます。

secureModifyToken クッキーの取得

次のメソッドでは，初期 GET 要求からすべてのクッキーを取得します。標準的な Java HTTP クライアントでは，Apache HttpClient が行うようなユーザのためのクッキーの管理は自動的に行われません。クッキーの取得と管理は別々に行う必要があります。

```
private static List<HttpCookie> getCookies(final String path)
{
```

```
   final URL url = new URL(path);
   final HttpURLConnection connection = url.openConnection();
   final List<HttpCookie> result = new ArrayList<HttpCookie>();

   connection.setRequestMethod("GET");

   // Set the username and password for the request
   byte[] encodedUserPassword = Base64.encodeBase64((username + ":"+
password).getBytes());
   connection.setRequestProperty("Authorization", "Basic " + new
String(encodedUserPassword));

   connection.connect();
   int response = connection.getResponseCode();
   if (response == 200)
   {
      for (int i=1; (final String headerName =
connection.getHeaderFieldKey(i)) != null; i++)
      {
         if (headerName.equals("Set-Cookie"))
         {
            final String cookieString = connection.getHeaderField(i);
            final List<HttpCookie> cookies = HttpCookie.parse
(cookieString);
            if (cookies != null && !cookies.isEmpty())
               result.addAll(cookies);
         }
      }
   }
   return result;
}
```

X-Secure-Modify-Token HTTP ヘッダの設定

次のコードは，`secureModifyToken` クッキーが存在する場合に，POST 要求および HTTP ヘッダの `X-Secure-Modify-Token` にクッキーを追加します。

```
final URL url
final List<HttpCookie> cookies = getCookies("http://" + localHostName
+ ":"+ port + "/opr-console/rest");

final URL url = new URL("http://" + localHostName + ":"+ port + "/opr-
console/rest/9.10/event_list");
final HttpURLConnection connection = url.openConnection();

// Set the cookies and HTTP header for the request
for (HttpCookie cookie :cookies)
{
   // add the cookies to the request
   connection.addRequestProperty("Cookie", cookie.getName() + "=" +
cookie.getValue());
   if (cookie.getName().equalsIgnoreCase("secureModifyToken"))
```

```
    {
       // add the HTTP header
       connection.setRequestProperty("X-Secure-Modify-Token",
cookie.getValue());
    }
}
...
```

Apache HttpClient を使用した場合のサンプル・コード

次のサンプル・コードでは，最初に `secureModifyToken` クッキーが取得され，次に `X-Secure-Modify-Token` HTTP ヘッダが設定されます。

secureModifyToken クッキーの取得

次のメソッドは null を返す場合があります。その場合，ターゲット Web サービスで `X-Secure-Modify-Token` HTTP ヘッダが必要とされないと想定されます(たとえば，BSM 9.10 以前のバージョンはこの HTTP ヘッダを必要としません) 。

```
private static String getSecureModifyToken(final HttpClient client,
final String url)

{
   int rc = -1;

   String secureModifyToken = null;

   // get the service document from the base path
   final HttpMethod getMethod = new GetMethod(url);
   getMethod.setFollowRedirects(true);
   getMethod.setDoAuthentication(true);
   getMethod.setRequestHeader("Accept", "text/plain, text/xml,
application/xml, application/atomsvc+xml");
   getMethod.setRequestHeader("Accept-Language", System.getProperty
("user.language", "en") + "-"
      + System.getProperty("user.country", "US"));
   try
   {
      client.executeMethod(getMethod);
      rc = getMethod.getStatusCode();
   }
   catch (IOException ioe)
   {
      // ignore any errors for backwards compatibility
   }
   if (rc == HttpStatus.SC_OK)
   {
      // look for the secureModifyToken
     Cookie[] cookies = client.getState().getCookies();
     if (cookies != null && cookies.length > 0)
     {
        for (Cookie cookie :cookies)
```

```
            {
               if (SECURE_MODIFY_TOKEN.equalsIgnoreCase(cookie.getName()))
                   secureModifyToken = cookie.getValue();
            }
         }
      }
      return secureModifyToken;
}
```

X-Secure-Modify-Token HTTP ヘッダの設定

```
HttpClient client = new HttpClient();
client.getState().setCredentials(new AuthScope(hostname, port),
       new UsernamePasswordCredentials(username, password));
final String secureModifyToken = getSecureModifyToken(client,
       "http://" + localHostName + ":"+ port + "/opr-console/rest");
String url = "http://" + localHostName + ":"+ port + "/opr-
console/rest/9.10/event_list";
PostMethod method = new PostMethod(url);
if (secureModifyToken != null)
   method.setRequestHeader("X-Secure-Modify-Token",
secureModifyToken);
...
```

Apache Wink RestClient を使用した場合のサンプル・コード

次のサンプル・コードでは，最初に `secureModifyToken` クッキーが取得され，次に `X-Secure-Modify-Token` HTTP ヘッダが設定されます。

初期クッキーの取得

次のメソッドでは，初期 GET 要求からすべてのクッキーを取得します。Apache Wink RestClient では，Apache HttpClient が行うようなユーザのためのクッキーの管理は自動的に行われません。クッキーの取得と管理は別々に行う必要があります。

```
private static Set<Cookie> getCookies(final String url, final
RestClient client)
{
   final Set<Cookie> cookies = new HashSet <Cookie>();
   final Resource resource = client.resource(url);

   // Set the username and password for the request
   byte[] encodedUserPassword = Base64.encodeBase64((username + ":"+
password).getBytes());
   resource.header("Authorization", "Basic " + new String
(encodedUserPassword));
   final ClientResponse response = resource.get();


   final MultivaluedMap<String, String> headers = response.getHeaders
();
   if (headers != null)
   {
```

```
      for (final Map.Entry<String, List<String>> header
:headers.entrySet())
      {
         if ("Set-Cookie".equalsIgnoreCase(header.getKey()))
         {
            for (final String value :header.getValue())
            {
               if (value != null && value.length() > 0)
               try
               {
                  cookies.add(Cookie.valueOf(value));
               }
               catch (IllegalArgumentException e)
               {
                  // ignore this entry
               }
            }
         }
      }
   }
   return cookies;
}
```

X-Secure-Modify-Token HTTP ヘッダの設定

次のコードは，`secureModifyToken` クッキーが存在する場合に，REST リソースおよび HTTP ヘッダの `X-Secure-Modify-Token` にクッキーを追加します。

```
final RestClient client = new RestClient();
final Set<Cookie> cookies = getCookies("http://" + localHostName +
":"+ port + "/opr-console/rest", client);

String url = "http://" + localHostName + ":"+ port + "/opr-
console/rest/9.10/event_list";
final Resource resource = client.resource(url);

// Set the username and password for the request
byte[] encodedUserPassword = Base64.encodeBase64((username + ":"+
password).getBytes());
resource.header("Authorization", "Basic " + new String
(encodedUserPassword));

// Set the cookies and HTTP header for the request
for (Cookie cookie :cookies)
{
   // add the cookies to the resource
   resource.cookie(cookie);
   if (cookie.getName().equalsIgnoreCase("secureModifyToken"))
   {

      // add the HTTP header
      resource.header("X-Secure-Modify-Token", cookie.getValue());
```

```
    }
}
...
```

強化されたセキュリティ保護の無効化

`X-Secure-Modify-Token` HTTP ヘッダを設定するイベント Web サービス・クライアントは，オペレーション管理のイベント Web サービス 9.0x 以下と通信するときに失敗する場合があります。したがって，インフラストラクチャ設定マネージャの Web サービス設定で，強化されたセキュリティ保護を無効にすることができます。

1. インフラストラクチャ設定マネージャの Web サービス設定に移動します。

   [**インフラストラクチャ設定**] > [**アプリケーション**] > [**オペレーション管理**] > [**オペレーション管理 - Web サービス設定**] > [**安全な変更**]

2. 標準設定値の true を **false** に変更します。

3. *オプション*。次の追加対策を実装すると，オペレーション管理の使用時における悪意ある攻撃に対するエンド・ユーザの保護を強化できます。

   - Web ブラウザにユーザ名およびパスワードを記憶させないようにします。
   - 同一の Web ブラウザを使用して，オペレーション管理とインターネットへのアクセスを同時に行わないようにします(タブを使用したブラウジング) 。オペレーション管理にログインしている場合，Web ブラウザでほかの Web サイトを閲覧しないようにします。
   - Web ブラウザを統合する HTML 対応アプリケーション(電子メールまたはニュースリーダー・アプリケーション) では，単に電子メール・メッセージやニュース・メッセージを閲覧するだけでも攻撃の実行につながる場合があるため，更なるリスクが発生します。クライアント・ワークステーションをオペレーション管理および上記のようなアプリケーションに接続する場合には注意する必要があります。

CA SiteMinder を使用した環境

CA SiteMinder Web エージェントが `CssChecking=YES` のように設定されている場合，CA SiteMinder Web エージェント `BadUrlChars` パラメータで設定された文字が CA SiteMinder Web エージェントによって拒否されます。標準設定では，次の文字がこのリストに含まれます。

- シングルの引用符 (‘)
- より大きいを表す記号 (>)
- より小さいを表す記号 (<)

イベント Web サービス・クライアントは，URL のクエリ・パラメータ・セクションで次の文字を使用できません。

- **文字列リテラル :** CA SiteMinder を使用する環境では，シングルの引用符 (') が拒否される場合があるため，二重引用符を使用して文字列リテラルを囲んでください。詳細は，294ページ「値のタイプ」を参照してください。

CA SiteMinder は，% エンコーディングをブロックします。そのため，文字列リテラル自体では，次のようにパーセント記号 (%) の変わりにドル記号 ($) を使用して違反文字をエスケープする必要があります。

- シングル引用符 (') を $60 で置換。
- より大きいを表す記号 (>) を $3E で置換。
- より小さいを表す記号 (<) を $3C で置換

詳細は，296ページ「URL エスケープ・コード」を参照してください。

- **演算子**：より大きいを表す記号 (>) とより小さいを表す記号 (<) は CA SiteMinder エージェントによって拒否される場合があるため，その代わりに GT および LT エイリアスを使用してください。詳細は，294ページ「演算子のエイリアス」を参照してください。

REST Web サービス・コマンドライン・ユーティリティ

提供されている REST Web サービス・コマンドライン・ユーティリティを使用すると，次の操作を実行できます。

- イベント Web サービスの単純なテストの実行。
- 作成，読み取り，更新，削除の 4 つの基本的な操作の実行。

コマンドライン・ユーティリティは次の場所にあります。

Windows :`<HPBSM のルート・ディレクトリ>/opr/bin/RestWsUtil.bat`

Linux :`<HPBSM のルート・ディレクトリ>/opr/bin/RestWsUtil.sh`

`RestWsUtil` ユーティリティを使用すると，作成，読み取り，更新，削除の 4 つの基本的な操作を実行できます。作成および更新では，REST Web サービスに送信するペイロードを含む入力ファイルが必要です。読み取りおよび削除では，ペイロードは設定されません。このユーティリティでは基本的な認証が使用されます。ユーザ名およびパスワードを指定するためのパラメータがあります。

ユーティリティ・ヘルプを呼び出す方法

ユーティリティのヘルプを呼び出すには，次のコマンドを入力します。

```
RestWsUtil -help
```

その使用方法は次のとおりです。

使用方法 :`RestWsUtil (-h | -version |`

`(-r | -d | ((-c | -u) <filename> [-content_type <type>])`

`[-o <filename>]`

`(([-ssl] [-server <server>] [-p <port>]) | [-u <URL>]))`

`[-username <login name>] [-password <password>] [-v]`

`RestWsUtil` ユーティリティのオプションは次のとおりです。

`-c,-create <in_file>`	指定した XML ドキュメントでリソースを作成します。HTTP POST 操作。
`-content_type <type>`	作成および更新操作のために設定された HTTP ヘッダの content-type 値です。標準設定値は `application/xml` です。
`-d,-delete`	指定されたリソースを削除します。URL は削除するリソースを直接指します。HTTP DELETE 操作。
`-h,-help`	このメッセージを印刷して終了します。

-o,-output <out_file>	Web サービス要求から返されたテキストのための出力ファイルの名前です。標準設定値は stdout です。
-p,-port <port>	ポート番号を設定します。HTTP の標準設定のポート番号は 80 で，HTTPS の場合は 443 です。url オプションとともにこのオプションを指定しないでください。
-password <password>	指定されたユーザのパスワード。
-r,-read	指定されたりソースを読み取ります。HTTP GET 操作。
-server <server>	ターゲット・ゲートウェイ・サーバを設定します。値は，ゲートウェイ・サーバのホスト名または IP アドレスです。url オプションとともにこのオプションを指定しないでください。
-ssl	プロトコルを HTTPS に設定します。標準設定は HTTP の使用です。url オプションとともにこのオプションを指定しないでください。
-update <in_file>	リソースを，指定された XML ドキュメントの変更によって更新します。HTTP PUT 操作。
-url <URL>	ゲートウェイ・サーバの URL。このオプションは，ssl オプション，server オプション，port オプションと同時に使用できません。
-username <login name>	認証のためユーザ・ログイン名が必要です。
-v, -verbose	詳細な出力を印刷します。
-version	バージョン情報を印刷して終了します。

終了ステータスは次のうちのいずれかである可能性があります。

0	正常に終了。
1	失敗。
3	出力先変更(300 ～ 399)。
4	クライアント エラー(400 ～ 499)。
5	内部サーバ・エラー(500 ～ 599)。

例

RestWsUtil ユーティリティの使用例を次に示します。

イベントの読み取り

RestWsUtil コマンドライン・ユーティリティを使用して，イベントを読み取り，その結果を test.xml という出力ファイルに送る例を次に示します。

```
RestWsUtil -r -username admin -o test.xml -verbose

Password:******

INFO:Read the resource located at:http://bsmserver.example.com/opr-
console/rest/9.10/event_list
```

```
INFO:Operation successful.
```

イベントのカスタム属性の読み取り

イベントのカスタム属性を読み取るには，属性の読み取りの対象となるイベントの ID を含む完全な URL を指定する必要があります。

次の例では，標準設定の stdout に出力を送信します。

```
RestWsUtil -r -url http://<fully qualified domain name of BSM gateway
server>/opr-console/rest/9.10/event_list/0695624b-93fa-40b1-
8b0fc9b4ea07a4ec/custom_attribute_list -username admin -verbose
Password:****** INFO:Read the resource located
at:http://localhost/opr-console/rest/9.10/event_list/0695624b-93fa-
40b1-8b0f-c9b4ea07a4ec/custom_attribute_list

<?xml version="1.0"encoding="UTF-8"standalone="yes"?> <custom_
attribute_list xmlns="http://www.hp.com/2009/software/opr/data_model"
self="http://localhost:80/opr-console/rest/9.10/event_list/0695624b-
93fa-40b1-8b0f-c9b4ea07a4ec/custom_attribute_list" type="urn:x-
hp:2009:software:data_model:opr:type:event:custom_attribute_
list"version="1.0"> <custom_attribute self="http://localhost:80/opr-
console/rest/9.10/event_list/0695624b-93fa-40b1-
8b0fc9b4ea07a4ec/custom_attribute_list/CiResolverSimilarityMetric"
type="urn:x-hp:2009:software:data_model:opr:type:event:custom_
attribute"version="1.0"> <name>CiResolverSimilarityMetric</name>
<value>100</value> </custom_attribute> </custom_attribute_list>

INFO:Operation successful.
```

イベントの注釈の読み取り

イベントの注釈を読み取るには，注釈の読み取りの対象となるイベントの ID を含む完全な URL を指定する必要があります。

次の例では，標準設定の stdout に出力を送信します。

```
-username admin -verbose Password:****** INFO:Read the resource
located at:http://<fully qualified domain name of BSM gateway
server>/opr-console/rest/9.10/event_list/10d9a54f-54b9-41d6-b933-
ed84b4f5e43e/annotation_list

<?xml version="1.0" encoding="UTF-8"?> <annotation_list
xmlns="http://www.hp.com/2009/software/opr/data_model"
self="http://mambo.mambo.net:80/opr-console/rest/9.10/event_
list/10d9a54f-54b9-41d6-b933-ed84b4f5e43e/annotation_list"
type="urn:x-hp:2009:software:data_model:opr:type:event:annotation_
list" version="1.0"> <annotation self="http://mambo.mambo.net:80/opr-
console/rest/9.10/event_list/10d9a54f-54b9-41d6-b933-
ed84b4f5e43e/annotation_list/74b869a8-399c-42cb-854c-03b9ced975c3"
type="urn:x-hp:2009:software:data_model:opr:type:event:annotation"
version="1.0"> <id>74b869a8-399c-42cb-854c-03b9ced975c3</id> <event_
ref target_role="urn:x-hp:2009:software:data_
model:opr:relationship:event:is_related_to:event" type="urn:x-
```

```
hp:2009:software:data_model:opr:type:reference:event:event_ref"
version="1.0"> <target_id>10d9a54f-54b9-41d6-b933-
ed84b4f5e43e</target_id> <target_type>urn:x-hp:2009:software:data_
model:opr:type:event</target_type> <target_
href>http://mambo.mambo.net:80/opr-console/rest/9.10/event_
list/10d9a54f-54b9-41d6-b933-ed84b4f5e43e</target_href> </event_ref>
<author>admin</author> <time_created>2010-07-
13T14:18:14.860+02:00</time_created> <text>Some annotation text</text>
</annotation> </annotation_list>

INFO:Operation successful.
```

カスタム属性の作成

次の例では，指定したイベントに対して MyNewCA という新規カスタム属性を作成する方法を示します。

```
RestWsUtil -c newca.xml -url http://<fully qualified domain name of
BSM gateway server>/opr-console/rest/9.10/event_list/0695624b-93fa-
40b1-8b0fc9b4ea07a4ec/custom_attribute_list -username admin -verbose
Password:******* INFO:Create the resource located at:newca.xml

<?xml version="1.0"encoding="UTF-8"standalone="yes"?> <custom_
attribute xmlns="http://www.hp.com/2009/software/opr/data_model"
self="http://localhost:80/opr-console/rest/9.10/event_list/0695624b-
93fa-40b1-8b0f-c9b4ea07a4ec/custom_attribute_list/MyNewCA"
type="urn:xhp:2009:software:data_model:opr:type:event:custom_
attribute"version="1.0"> <name>MyNewCA</name> <value>100</value>
</custom_attribute>

INFO:Operation successful.
```

新規のカスタム属性およびその値が stdout に書き込まれます。

newca.xml ファイルの内容は次のようになります。

```
<custom_attribute xmlns="http://www.hp.com/2009/software/opr/data_
model"> <name>MyNewCA</name> <value>100</value> </custom_attribute>
```

注釈の作成

次の例では，注釈の作成方法を示します。新規の注釈は newanno.xml ファイルに書き込まれます。

```
RestWsUtil -c newanno.xml -url http://<fully qualified domain name of
BSM gateway server>/opr-console/rest/9.10/event_list/10d9a54f-54b9-
41d6-b933-ed84b4f5e43e/annotation_list -username admin -verbose
Password:****** INFO:Create the resource located at:newanno.xml

<?xml version="1.0" encoding="UTF-8"?>

<annotation xmlns="http://www.hp.com/2009/software/opr/data_model"
self="http://mambo.mambo.net:80/opr-console/rest/9.10/event_
list/10d9a54f-54b9-41d6-b933-ed84b4f5e43e/annotation_list/62f86310-
```

```
38ce-4793-ab3f-23ecfb3f2a67" type="urn:x-hp:2009:software:data_
model:opr:type:event:annotation" version="1.0"> <id>62f86310-38ce-
4793-ab3f-23ecfb3f2a67</id> <event_ref target_role="urn:x-
hp:2009:software:data_model:opr:relationship:event:is_related_
to:event" type="urn:x-hp:2009:software:data_
model:opr:type:reference:event:event_ref" version="1.0"> <target_
id>10d9a54f-54b9-41d6-b933-ed84b4f5e43e</target_id> <target_
type>urn:x-hp:2009:software:data_model:opr:type:event</target_type>
<target_href>http://mambo.mambo.net:80/opr-console/rest/9.10/event_
list/10d9a54f-54b9-41d6-b933-ed84b4f5e43e</target_href> </event_ref>
<author>admin</author> <time_created>2010-07-
13T14:56:56.642+02:00</time_created> <text>Some new annotation
text</text> </annotation>

INFO:Operation successful.
```

newanno.xml ファイルの内容は次のようになります。

```
<annotation xmlns="http://www.hp.com/2009/software/opr/data_model" >
<author>admin</author> <text>Some new annotation text</text>
</annotation>
```

カスタム属性の更新

次の例では，指定したイベントの MyNewCA というカスタム属性を新規の値を使用して更新する方法を示します。

```
RestWsUtil -update updateca.xml -url http://<fully qualified domain
name of BSM gateway server>/opr-console/rest/9.10/event_list/0695624b-
93fa-40b1-8b0fc9b4ea07a4ec/custom_attribute_list/MyNewCa -username
admin -verbose Password:****** INFO:Update the resource with changes
located at:updateca.xml

<?xml version="1.0"encoding="UTF-8"standalone="yes"?> <custom_
attribute xmlns="http://www.hp.com/2009/software/opr/data_model"
self="http://localhost:80/opr-console/rest/9.10/event_list/0695624b-
93fa-40b1-8b0f-c9b4ea07a4ec/custom_attribute_list/MyNewCA"
type="urn:x-hp:2009:software:data_model:opr:type:event:custom_
attribute"version="1.0"> <name>MyNewCA</name> <value>999</value>
</custom_attribute>

INFO:Operation successful.
```

カスタム属性の更新済みの値が stdout に書き込まれます。

updateca.xml ファイルの内容は次のようになります。

```
<custom_attribute xmlns="http://www.hp.com/2009/software/opr/data_
model"> <value>999</value> </custom_attribute>
```

注釈の更新

次の例では，注釈の更新方法を示します。更新した注釈は updateanno.xml ファイルに書き込まれます。

```
-url http://<fully qualified domain name of BSM gateway server>/opr-
console/rest/9.10/event_list/10d9a54f-54b9-41d6-b933-
ed84b4f5e43e/annotation_list/582f0488-15ad-40ac-907f-fec21041b5c0 -
username admin -verbose Password:****** INFO:Update the resource with
changes located at:updateanno.xml

<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <annotation
xmlns="http://www.hp.com/2009/software/opr/data_model"
self="http://mambo.mambo.net:80/opr-console/rest/9.10/event_
list/10d9a54f-54b9-41d6-b933-ed84b4f5e43e/annotation_list/582f0488-
15ad-40ac-907f-fec21041b5c0" type="urn:x-hp:2009:software:data_
model:opr:type:event:annotation" version="1.0"> <id>582f0488-15ad-
40ac-907f-fec21041b5c0</id> <event_ref target_role="urn:x-
hp:2009:software:data_model:opr:relationship:event:is_related_
to:event" type="urn:x-hp:2009:software:data_
model:opr:type:reference:event:event_ref" version="1.0"> <target_
id>10d9a54f-54b9-41d6-b933-ed84b4f5e43e</target_id> <target_
type>urn:x-hp:2009:software:data_model:opr:type:event</target_type>
<target_href>http://mambo.mambo.net:80/opr-console/rest/9.10/event_
list/10d9a54f-54b9-41d6-b933-ed84b4f5e43e</target_href> </event_ref>
<author>admin</author> <time_created>2010-07-
13T15:56:31.220+02:00</time_created> <text>Updated annotation
text</text> </annotation>

INFO:Operation successful.
```

updateanno.xml ファイルの内容は次のようになります。

```
<annotation xmlns="http://www.hp.com/2009/software/opr/data_model">
<text>Updated annotation text</text> </annotation>
```

イベントのタイトルの更新

次の例では，イベントのタイトルを変更する方法を示します。変更したタイトルは update.xml ファイルに書き込まれます。

```
RestWsUtil -update update.xml -username admin -url http://<fully
qualified domain name of BSM gateway server>/opr-
console/rest/9.10/event_list/d36a157e-7312-4302-a2da-e5b7230b0e21
Password: *******

INFO:Operation successful.
```

update.xml ファイルの内容は次のようになります。

```
<event xmlns="http://www.hp.com/2009/software/opr/data_model">
<title>New title goes here</title> </event>
```

イベントのライフサイクル状態の更新

次の例では，イベントのライフサイクル状態を変更する方法を示します。これは，イベントのタイトルを変更する方法と同じです。変更したライフサイクル状態は update.xml ファイルに書き込まれます。

```
RestWsUtil -update update.xml -username admin -url http://<fully
qualified domain name of BSM gateway server>/opr-
```

```
console/rest/9.10/event_list/d36a157e-7312-4302-a2da-e5b7230b0e21
Password: *******

INFO:Operation successful.
```

update.xml ファイルの内容は次のようになります。

```
<event xmlns="http://www.hp.com/2009/software/opr/data_model">
<state>in_progress</state> </event>
```

イベントのライフサイクル状態および重大度の更新

次の例では，イベントのライフサイクル状態および重大度を変更する方法を示します。これは，イベントのその他の内容を更新する方法と同じです。変更したライフサイクル状態および重大度は update.xml ファイルに書き込まれます。

```
RestWsUtil -update update.xml -username admin -url http://<fully
qualified domain name of BSM gateway server>/opr-
console/rest/9.10/event_list/d36a157e-7312-4302-a2da-e5b7230b0e21
Password: *******

INFO:Operation successful.
```

update.xml ファイルの内容は次のようになります。

```
<event xmlns="http://www.hp.com/2009/software/opr/data_model">
<state>in_progress</state> <severity>critical</severity> </event>
```

接続サーバへのコントロール移転の情報に基づくイベントの更新

次の例では，イベントのコントロールが接続サーバに移転されているという情報を使用してイベントを更新する方法を示します。この情報は update.xml ファイルに書き込まれます。

```
RestWsUtil -update update.xml -username admin -url http://<fully
qualified domain name of BSM gateway server>/opr-
console/rest/9.10/event_list/d36a157e-7312-4302-a2da-e5b7230b0e21
Password:******* INFO:Operation successful.
```

update.xml ファイルの内容は次のようになります。

```
<event xmlns="http://www.hp.com/2009/software/opr/data_model">
<control_transferred_to> <name>logger</name> </control_transferred_to>
</event>
```

一括イベント更新 :イベントの状態および重大度の更新

「DB down」というタイトルに設定されたすべてのイベントが in_progress の重大度に設定されている例を次に示します。更新されたイベントのリストが呼び出し側に返されます。

```
RestWsUtil -update update.xml -username admin -url http://<fully
qualified domain name of BSM gateway server>/opr-
console/rest/9.10/event_list?query=title='DB down' Password:*******
INFO:Operation successful.
```

update.xml ファイルの内容は次のようになります。

```
<event xmlns="http://www.hp.com/2009/software/opr/data_model">
<state>in_progress</state> <severity>critical</severity> </event>
```

カスタム属性の削除

次の例では，選択したイベントのカスタム属性リストから MyNewCA というカスタム属性を削除する方法を示します。

```
RestWsUtil -d -url http://<fully qualified domain name of BSM gateway
server>/opr-console/rest/9.10/event_list/0695624b-93fa-
40b18b0fc9b4ea07a4ec/custom_attribute_list/MyNewCa -username admin -
verbose Password:****** INFO:Deleting resource located
at:http://localhost/opr-console/rest/9.10/event_list/0695624b-93fa-
40b1-8b0f-c9b4ea07a4ec/custom_attribute_list/MyNewCa

<?xml version="1.0"encoding="UTF-8"standalone="yes"?> <custom_
attribute xmlns="http://www.hp.com/2009/software/opr/data_model"
self="http://localhost:80/opr-console/rest/9.10/event_list/0695624b-
93fa-40b1-8b0f-c9b4ea07a4ec/custom_attribute_list/MyNewCa"
type="urn:x-hp:2009:software:data_model:opr:type:event:custom_
attribute"version="1.0"> <name>MyNewCa</name> <value>999</value>
</custom_attribute>

INFO:Operation successful.
```

注釈の削除

次の例では，選択したイベントの注釈リストから注釈を削除する方法を示します。

```
RestWsUtil -d -url http://<fully qualified domain name of BSM gateway
server>/opr-console/rest/9.10/event_list/10d9a54f-54b9-41d6-b933-
ed84b4f5e43e/annotation_list/582f0488-15ad-40ac-907f-fec21041b5c0 -
username admin -verbose Password:****** INFO:Deleting resource located
at:http://mambo.mambo.net/opr-console/rest/9.10/event_list/10d9a54f-
54b9-41d6-b933-ed84b4f5e43e/annotation_list/582f0488-15ad-40ac-907f-
fec21041b5c0

<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <annotation
xmlns="http://www.hp.com/2009/software/opr/data_model"
self="http://mambo.mambo.net:80/opr-console/rest/9.10/event_
list/10d9a54f-54b9-41d6-b933-ed84b4f5e43e/annotation_list/582f0488-
15ad-40ac-907f-fec21041b5c0" type="urn:x-hp:2009:software:data_
model:opr:type:event:annotation" version="1.0"> <id>582f0488-15ad-
40ac-907f-fec21041b5c0</id> <event_ref target_role="urn:x-
hp:2009:software:data_model:opr:relationship:event:is_related_
to:event" type="urn:x-hp:2009:software:data_
model:opr:type:reference:event:event_ref" version="1.0"> <target_
id>10d9a54f-54b9-41d6-b933-ed84b4f5e43e</target_id> <target_
type>urn:x-hp:2009:software:data_model:opr:type:event</target_type>
<target_href>http://mambo.mambo.net:80/opr-console/rest/9.10/event_
list/10d9a54f-54b9-41d6-b933-ed84b4f5e43e</target_href> </event_ref>
<author>admin</author> <time_created>2010-07-
13T15:56:31.220+02:00</time_created> <text>Updated annotation
text</text> </annotation>

INFO:Operation successful.
```

イベントの作成

次の例では，新規イベントの作成方法を示します。

```
RestWsUtil -create create.xml -username admin -password ***** -url
http://<bsmserver.example.com>/opr-console/rest/9.10/event_list
```

`create.xml` ファイルの内容は次のようになります。

```
<event xmlns="http://www.hp.com/2009/software/opr/data_model">
   <title>New event title</title>
   <severity>normal</severity>
   <priority>low</priority>
   <state>open</state>
</event>
```

イベント・リストの作成

次の例では，イベント・リストの作成方法を示します。

```
RestWsUtil -create createlist.xml -content_type "application/xml;
type=collection" -username admin -password ***** -url
http://<bsmserver.example.com>/opr-console/rest/9.10/event_list
```

`createlist.xml` ファイルの内容は次のようになります。

```
<event_list xmlns="http://www.hp.com/2009/software/opr/data_model">
   <event>
      <title>Major Event</title>
      <severity>major</severity>
   </event>
   <event>
      <title>Minor Event</title>
      <severity>minor</severity>
   </event>
</event_list>
```

注: この呼び出しからの HTTP 応答コードは `202 Accepted` になります。これは，POST 処理が正常である可能性があるため要求は送信されますが，そのリソースがまだ作成されていないことを意味します。新規イベントをイベント・パイプラインを介して送信する必要があります。新規イベントは最終的にイベント・データベースに保存される場合または保存されない場合があります。たとえば，イベント・パイプラインでは送信したイベントの重複が排除されます。このインタフェースを介したイベントの送信にはすべての標準制限が適用され，イベント・パイプラインを介して処理される必要があります。

イベント Web サービス・クエリ言語

イベント Web サービスでは，URL クエリ言語に標準クエリ・パラメータが使用されます。

クエリ・パラメータは，何らかの方法でリソースからの応答を変更するために使用されます。すべてのリソースでは，応答メディア・タイプを制御できます。リソースがコレクションの場合，コレクション内の返されたエントリのセットをさまざまな方法でフィルタリングできます。クエリ・パラメータは，応答をさらに制限するために希望の方法で組み合わせることができます。

イベント Web サービスには，特定の条件でイベント・リストをフィルタリングするための数多くのパラメータが用意されています。たとえば，特定のイベント・パラメータを URL クエリ・パラメータの条件と一致させて，結果をフィルタリングすることができます。特定のリスト項目数のみを表示することでリストのサイズを削減し，複数のページに分配できます。指定の日付および時間後に更新されたイベントのみを返す時間および日付パラメータを送信することもできます。

HTTP クエリ・パラメータ

本項では，イベント Web サービスによってサポートされる HTTP クエリ・パラメータについて説明します。

これらのクエリ・パラメータは，コレクション・リソースのみに適用され，HTTP GET メソッドと併用する場合または HTTP PUT メソッドを使用してイベントを一括して更新する場合のみ意味があります。これらのパラメータは，リソースに到達する URL の HTTP クエリ部分内に指定します。

URL の HTTP クエリ部分に指定できるパラメータの一覧を次に示します。クエリ・パラメータは，アンパサンド (&) 演算子を使用して，組み合わせて使用することができます。

クエリ・パラメータ	説明
`watermark`	日付および時間でフィルタリングするためのパラメータ。 詳細は，284ページ「日付および時間によるフィルタリング :watermark」を参照してください。
`query`	イベント属性でフィルタリングするためのパラメータ。 詳細は，285ページ「イベント属性によるフィルタリング :query」を参照してください。
`start_index`	クエリを開始する項目を定義するページング・クエリ・パラメータ。 詳細は，286ページ「ページング」を参照してください。
`page_size`	Atom フィードのページごとに表示する項目数を定義するために使用するページング・クエリ・パラメータ。 詳細は，286ページ「ページング」を参照してください。
`order_by`	指定したフィールドを基準として応答フィードの順序を指定するために使用する順序設定パラメータ。 詳細は，287ページ「順序設定」を参照してください。

クエリ・パラメータ	説明
order_direction	応答フィードを昇順または降順のいずれかで順序変更するかを指定するために使用する順序設定パラメータ。 詳細は, 287ページ「順序設定」を参照してください。
include_closed	イベント・サービス(event_list)をクエリするときに閉じられたイベントを含めるかどうかを指定するために使用するパラメータ。 標準設定値 :false 詳細は, 287ページ「データ包含」を参照してください。
include_relationships	イベント・サービス(event_list)をクエリするときに関係を含めるかどうかを指定するために使用するパラメータ。 標準設定値 :true 詳細は, 287ページ「データ包含」を参照してください。

クライアントは, メタデータまたはリソースのデータが特定の条件と一致するリソースのみを含めることによって応答フィードをフィルタリングするように指定できます。これは, URL の HTTP クエリ部分内でクエリ・パラメータを使用して指定できます。

289ページ「クエリ・フィルタ条件のプロパティ」には, 利用可能なクエリ・フィルタ条件プロパティおよびサポートされる演算子が示されています。

日付および時間によるフィルタリング :watermark

クライアントは応答フィードを時間に基づきフィルタリングするように指定できます。これらのクエリ・パラメータは, コレクション・リソースのみに適用され, HTTP GET メソッドと併用する場合のみ意味があります。

watermark パラメータを使用すると, 時間および日付に基づき項目のクエリを実行できます。watermark パラメータを指定すると, 指定した時間の後に作成または更新されたイベントのみが返されます。

たとえば, watermark=2009-01-01T00:00:00Z は 2009 年の初頭後に更新されたリソースのみを応答フィードに含めることを示します。

watermark クエリ・パラメータを使用すると, コレクションの最近更新されたリソースを取得できます。クエリで指定するすべての時間の場合と同様に, watermark パラメータの値は, XML スキーマの dateTime 形式で指定する必要があります。指定した日付後に更新されたリソースがコレクション内に存在しない場合, 応答フィードは空になります。クエリ・パラメータでの不正な形式の値(XML スキーマの dateTime 形式に準拠しない値など)ではエラー応答が発生します。

XML スキーマの日付タイプの詳細については, XML スキーマのマニュアルを参照してください。このマニュアルは次の場所で入手できます。

http://www.w3.org/TR/xmlschema-2(英語サイト)

さらに, 「Z」を使用して GMT 以外のタイムゾーンを指定する場合は, 「+」記号を指定する必要があります。これは URL 内に指定するため, URL エンコーディングされている必要があります。

エスケープする必要のある文字の URL エスケープ・コード一覧については, 297ページ「エスケープする必要のある文字に対する URL エスケープ・コード」を参照してください。

URL エンコーディングの詳細については，次の場所にある URL(Uniform Resource Locators) の仕様を参照してください。

http://www.rfc-editor.org/rfc/rfc1738.txt

特定の文字をエスケープする必要性として，たとえば，すべての「プラス」記号を「%2B」に置換しなければならない場合が挙げられます。その例を次に示します。

```
query=time_createdGT2009-10-19T17:06:54.453%2B02:00
```

2010 年 3 月 19 日 13:59:17 後のイベントのみをリストする URL 呼び出しの例を次に示します。

```
http://<bsmserver.example.com>/opr-console/rest/9.10/event_
list?alt=atom&watermark=2010-03-19T13:59:17%2B02:00
```

イベント属性によるフィルタリング :query

`query` パラメータは，特定のイベント属性値を使用して要求をフィルタリングします。`query` フィルタの条件は次のとおりです。

- イベント属性を指定するフィルタ・プロパティ
- サポートされる演算子
- プロパティの値

シンプル・フィルタ

簡単な例として，`severity` プロパティが `critical` に等しいすべてのクエリをフィルタリングする場合，Web サービス・クライアントは次の URL を要求する必要があります。

```
http://<bsmserver.example.com>/opr-console/rest/9.10/event_
list?alt=atom&query=severity%20EQ%20"critical"
```

複合フィルタ

複数のフィルタを使用して，より複雑なフィルタ・クエリを作成することができます。フィルタ・クエリ・パラメータは，論理演算子の AND，NOT，OR で区切ることで組み合わせることができます。論理演算子の AND が最初に解決されます。適切なクエリ処理を確実にするため，括弧を使用してください。

複合フィルタのいくつかの例を次に示します。

最初の例では，論理演算子の AND を使用して 2 つのシンプルなフィルタで 1 つの複合フィルタ・クエリを構成します。

```
http://<bsmserver.example.com>/opr-console/rest/9.10/event_
list?query=assigned_user%20EQ%20"admin"%20AND%20title%20EQ%20"My
Title"
```

次の例では，論理演算子の AND を使用してシンプルなフィルタと複雑なフィルタの組み合わせで複合フィルタ・クエリを構成します。

```
http://<bsmserver.example.com>/opr-console/rest/9.10/event_
list?query=assigned_user%20EQ%20"admin"%20AND%20related_ci[target_
id%20EQ%20"ffca22eea15268533029f17b4c01b008"]
```

次の例は，論理演算子の OR が最初に解決される複合フィルタの構成方法を示しています。

```
http://<bsmserver.example.com>/opr-console/rest/9.10/event_
list?query=assigned_user%20EQ%20"admin"%20AND%20(title%20EQ%20"My
Title"%20OR%20state%20EQ%20"closed")
```

次の例は，論理演算子の NOT が最初に解決される複合フィルタの構成方法を示しています。

```
http://<bsmserver.example.com>/opr-console/rest/9.10/event_
list?query=assigned_user%20EQ%20"admin"%20AND%20(title%20EQ%20"My
Title"%20NOT%20state%20EQ%20"closed")
```

ページング

追加の URL パラメータを指定することでクエリ結果を最適化できます。ページング・クエリ・パラメータは，応答フィードのエントリをフィルタリングして，クエリ・パラメータなしで返されるエントリのサブセットを結果として取得します。ページング・クエリ・パラメータは，コレクション・リソースに適用されるため，個別のリソースには影響を与えません。これらのパラメータは，HTTP GET メソッドで使用した場合のみ意味があります。ページング・クエリ・パラメータは，ほかのすべてのフィルタリング・クエリ・パラメータを考慮した後に応答フィードに適用されます。

次の 2 つのクエリ・パラメータを一緒に使用すると，クライアントが大量のエントリを含むフィードでページングするために使用できるインタフェースを提供できます。`start_index` および `page_size`。

start_index

`start_index` クエリ・パラメータを使用すると，応答フィードに返された最初のエントリのインデックスを指定できます。したがって，`start_index` パラメータを使用すると，クエリを開始する項目を定義できます。フィードの最初のエントリのインデックスは 1，2 番目のエントリのインデックスは 2 になり，後続も同じようになります。このクエリ・パラメータが指定されていない場合，`start_index` の標準設定値は常に 1 になります。1 未満の値を指定すると，要求によって fault が返されます。コレクションのエントリ数より大きな値を指定すると，空の応答フィードが返されます。

page_size

`page_size` クエリ・パラメータを使用すると，一度に返されるエントリの数を指定できます。したがって，`page_size` パラメータを使用すると，Atom フィードのページごとに表示する項目の数を定義できます。イベント Web サービスの標準設定値は 20 項目に設定されています。

最小値は 1 で，最大値はサービスが対応できる値に変更できます。`page_size` パラメータが設定されていない場合，すべてのアプリケーションが使用することが期待される標準設定値は存在しません。1 未満の値を指定すると，要求が失敗します。

start_index と page_size の組み合わせ

`start_index` と `page_size` パラメータを組み合わせることで，複数ページにわたって結果を表示できます。`start_index` の値は 0 より大きな値である必要があります。

たとえば，`page_size` を 5 に指定し，`start_index` 値を定義せずに URL を呼び出した場合，最初の 5 つの項目 (項目 1 ～ 5) が返されます。

`page_size` を 5 に，`start_index` 値を 6 に指定して URL を呼び出した場合，6 番目の項目から 5 つの項目 (項目 6 ～ 10) が 2 ページ目に返されます。したがって，クエリの結果として，Atom フィードの 2 ページに渡る 10 項目が返されることになります。

warning に設定した severity パラメータでフィルタリングしたクエリ結果の最初のページ(項目 1 ～ 5) を返すには，次の URL を呼び出します。

```
http://<bsmserver.example.com>/opr-console/rest/9.10/event_
list?alt=atom&query=severity%20EQ%20"warning"&page_size=5
```

クエリ結果の 2 ページ目 (項目 6 ～ 10) を取得するには，次の URL を呼び出します。

```
http://<bsmserver.example.com>/opr-console/rest/9.10/event_
list?alt=atom&query=severity%20EQ%20"warning"&page_size=5&start_
index=6
```

ページングを使用すると，クライアントはすべての要求に一定のページ・サイズを使用したり，要求ごとに異なるページ・サイズを使用することができます。ページを重複させることも可能です。上記の例で start_index を 3，page_size を 5 に設定した場合，要求したページが最初のページと重複します。重複するエントリが表示されたり，複数のページ要求においてエントリが追加または削除されたためエントリが不足する場合があります。

最初または最後のページに移動するリンクがすべてのフィードに設定されます。ページングを使用したためすべてのエントリが返されない場合，次のページおよび前のページに移動するリンクも示されます。これらのリンクは rel 属性を「first」，「last」，「next」または「previous」に設定することで記述します。これらのリンクについては，次の場所でアクセスできる RFC5005 に説明されています。

http://tools.ietf.org/html/rfc5005(英語サイト)

順序設定

クライアントは，特定の順序設定条件で応答フィードが返えられるように指定できます。これらのクエリ・パラメータは，コレクション・リソースのみに適用され，HTTP GET メソッドと併用する場合のみ意味があります。

order_by

order_by クエリ・パラメータを使用すると，指定したフィールドで応答フィードを順序変更するように指定できます。フィールドにはリソースのシンプルなデータまたはメタデータ・プロパティを使用できます。

order_by クエリ・パラメータが時間または sequence_number である場合，標準の順序設定が降順になり，最新のエントリが最初に表示されます。そうでない場合は，昇順になります。

order_direction

order_direction クエリ・パラメータを使用すると，指定した順序で応答フィードを順序変更するように指定できます。このクエリ・パラメータの有効な値は次の 2 つのみです。「ascending」および「descending」。その他の値では，エラー応答が発生します。順序設定クエリ・パラメータが指定されていない場合，標準設定値は降順です(APP 仕様では，フィードが更新時による降順で順序設定されることが示されています) 。

データ包含

データ包含クエリ・パラメータを使用すると，クライアントはクエリによって返される関連データの量を制御できます。その結果，応答の処理性能に直接影響できます。

include_closed

include_closed クエリ・パラメータは，イベント・サービス(event_list)に対するクエリを実行するときに，閉じられたイベントを含めるかどうかを指定するために使用します。このパラメータの標準設定値は false です。true に設定すると，クエリに閉じられたイベントが含まれます。そうでない場合，closed のライフサイクル状態以外のイベントのみがイベント Web サービスから返されます。

include_relationships

include_relationships クエリ・パラメータを使用すると，イベント・サービス(event_list)に対するクエリを実行するときに，関係を含めるかどうかを指定できます。このパラメータの標準設定値は true です。false に設定すると，クエリに関係が含まれません。たとえば，related_ci または source_ci がイベントに設定され，include_relationships クエリ・パラメータが false に設定されている場合，CI ID のみがイベント Web サービスから返されます。include_relationships パラメータを標準設定値の true に設定した場合とは対照的に，コンテナ CI (part_of)を含む主要な属性は解決されたり，返されません。

メディア・タイプ

クライアントは，指定したメディア・タイプで応答を返すように要求できます。

alt

alt クエリ・パラメータを使用すると，クライアントは，指定したメディア・タイプを使用して応答を返すようにサーバに指示することができます。このクエリ・パラメータは，コレクションを含むすべてのリソースに適用されます。また，応答を返すすべての HTTP メソッドで使用できます。パラメータの値は，たとえば application/atom+xml，application/json，application/xml などのメディア・タイプです。通常，サービスでは，所定のリソースをフォーマット化できるメディア・タイプが制限されています。サポートされるメディア・タイプの一覧は，Atom 応答形式(たとえば alt=atom，alt=json，alt=xml)から取得できます。Atom 応答形式はほとんどのリソースでサポートされます(255ページ「新規イベントを検出する方法」を参照)。

alt クエリ・パラメータを使用すると，クライアントは Accept HTTP ヘッダを使用した場合と同じ内容を表現できます。HTTP ヘッダを使用するメリットは，多くのクライアントが Accept ヘッダの組み込みサポートを保有しているという点です。alt クエリ・パラメータを使用するメリットは，Accept ヘッダを特定の値に設定するように指示しなくても，特定の応答形式を含むリンクを電子メール，チャット，ツイッター，ドキュメントなどで配布できる点です。サービスで Accept ヘッダと alt クエリ・パラメータの両方が設定されているメッセージを受信した場合，alt クエリ・パラメータが優先されます。

クエリ・フィルタ条件のプロパティ

289ページ「クエリ・フィルタ条件のプロパティ」には，利用可能なクエリ・フィルタ条件プロパティおよびサポートされる演算子が示されています。

注: 複雑な(ネストされた)属性を使用してフィルタリングを行う場合は，ネスティングを示すために括弧([])を使用する必要があります。複雑な属性を指定する場合，括弧記号([])を次のようにエスケープする必要があります。query=related_ci%5Btarget_id="ffca22eea15268533029f17b4c01b008"%5D。複雑な属性を使用したフィルタリング

に関する詳細については, 298ページ「複雑な属性」を参照してください。

クエリ・フィルタ条件のプロパティ

プロパティ	サポートされる演算子	値のタイプ	order_by のサポート	order_direction の標準設定値
application	EQ, NE, LIKE, IS NULL, IN	文字列リテラル	はい	昇順
assigned_group	EQ, NE, IN	文字列リテラル	はい	ID による昇順
assigned_group[id]	EQ, NE, IN	文字列リテラル	はい	ID による昇順
assigned_group [name]	EQ, NE, IN	文字列リテラル	はい	ID による昇順
assigned_user	EQ, NE, IN	文字列リテラル	はい	ID による昇順
assigned_user[id]	EQ, NE, IN	文字列リテラル	はい	ID による昇順
assigned_user [login_name]	EQ, NE, IN	文字列リテラル	はい	ID による昇順
category	EQ, NE, LIKE, IS NULL, IN	文字列リテラル	はい	昇順
cause[target_id]	EQ, IS NULL, IN	文字列リテラル(UUID)		
close_key_pattern	EQ, NE, LIKE, IS NULL, IN	文字列リテラル		
control_ transferred_to [dns_name]	EQ, NE, LIKE	文字列リテラル		
control_ transferred_to [external_id]	EQ, NE, LIKE, IS NULL, IN	文字列リテラル		
control_ transferred_to[id]	EQ, NE, LIKE, IS NULL, IN	文字列リテラル(UUID)		

プロパティ	サポートされる演算子	値のタイプ	order_by のサポート	order_direction の標準設定値
control_transferred_to[state]	EQ, NE, IS NULL, IN	文字列リテラル(UUID)		
control_transferred_to[state]	EQ, NE, IS NULL, IN	文字列リテラル(UUID)		
ca[<CA 名>]	EQ, NE, LIKE, IN	文字列リテラル		
ca[<CA 名>]	EQ, NE, LIKE, IN	文字列リテラル		
custom_attribute_list[<CA 名>]	EQ, NE, LIKE, IN	文字列リテラル		
description	EQ, NE, LIKE, IS NULL, IN	文字列リテラル	はい	昇順
duplicate_count	EQ, LT, GT, LTE, GTE, NE, IN	整数	はい	昇順
eti_hint	EQ, NE, LIKE, IS NULL, IN	文字列リテラル		
eti_value_id	EQ, NE, LIKE, IS NULL, IN	文字列リテラル		
id	EQ, NE, IN	文字列リテラル(UUID)		
instruction_available		ブール		
key	EQ, NE, LIKE, IS NULL, IN	文字列リテラル		
log_only	EQ	ブール		

プロパティ	サポートされる演算子	値のタイプ	order_by のサポート	order_direction の標準設定値
node[target_global_id]	EQ, NE, IS NULL, IN	文字列リテラル		
node[target_id]	EQ, NE, IS NULL, IN	文字列リテラル		
node_hints[hint]	EQ, NE, LIKE, IS NULL, IN	文字列リテラル		
node_hints[node_core_id]	EQ, NE, LIKE, IS NULL, IN	文字列リテラル		
node_hints[node_dns_name]	EQ, NE, LIKE, IS NULL, IN	文字列リテラル		
node_hints[node_ip_address]	EQ, NE, LIKE, IS NULL, IN	文字列リテラル		
object	EQ, NE, LIKE, IS NULL, IN	文字列リテラル	はい	昇順
om_service_id	EQ, NE, LIKE, IS NULL, IN	文字列リテラル		
om_user	EQ, NE, LIKE, IS NULL, IN	文字列リテラル	はい	昇順
original_data	EQ, NE, LIKE, IS NULL, IN	文字列リテラル		
original_id	EQ, NE, LIKE, IS NULL, IN	文字列リテラル		
priority	EQ, NE, LIKE, IS NULL, IN	文字列リテラル	はい	昇順
received_as_notify	EQ	ブール		

プロパティ	サポートされる演算子	値のタイプ	order_by のサポート	order_direction の標準設定値
received_on_ci_downtime	EQ	ブール		
related_ci[sub_component]	EQ, NE, LIKE, IS NULL, IN	文字列リテラル		
related_ci[sub_component]	EQ, NE, LIKE, IS NULL, IN	文字列リテラル		
related_ci[target_global_id]	EQ, NE, IS NULL, IN	文字列リテラル		
related_ci[target_id]	EQ, NE, LIKE, IS NULL, IN	文字列リテラル		
related_ci_hints[hint]	EQ, NE, LIKE, IS NULL, IN	文字列リテラル		
sequence_number	EQ, LT, GT, GTE, LTE, NE, IN	整数	はい	降順
severity	EQ, NE, IS NULL, IN	文字列リテラル	はい	昇順
skip_duplicate_suppression	EQ	ブール		
solution	EQ, NE, LIKE, IS NULL, IN	文字列リテラル	はい	昇順
source_ci[target_global_id]	EQ, NE, IS NULL, IN	文字列リテラル		
source_ci[target_id]	EQ, NE, IS NULL, IN	文字列リテラル		

プロパティ	サポートされる演算子	値のタイプ	order_by のサポート	order_direction の標準設定値
source_ci_hints [hint]	EQ, NE, LIKE, IS NULL, IN	文字列リテラル		
source_ci_hints [node[core_id]]	EQ, NE, LIKE, IS NULL, IN	文字列リテラル		
source_ci_hints [node[dns_name]]	EQ, NE, LIKE, IS NULL, IN	文字列リテラル		
source_ci_hints [node[ip_address]]	EQ, NE, LIKE, IS NULL, IN	文字列リテラル		
sourced_from[dns_name]	EQ, NE, LIKE, IS NULL, IN	文字列リテラル		
sourced_from [external_id]	EQ, NE, LIKE, IS NULL, IN	文字列リテラル		
sourced_from[id]	EQ, NE, IS NULL, IN	文字列リテラル(UUID)		
state	EQ, NE, IS NULL, IN	文字列リテラル	はい	昇順
sub_category	EQ, NE, LIKE, IS NULL, IN	文字列リテラル	はい	昇順
time_changed	LT, GT, GE, LE	日時	はい	降順
time_created	LT, GT, GE, LE	日時	はい	降順
time_received	LT, GT, GE, LE	日時	はい	降順
time_state_changed	LT, GT, GE, LE	日時	はい	降順
title	EQ, NE, LIKE, IS NULL, IN	文字列リテラル	はい	昇順
type	EQ, NE, LIKE, IS NULL, IN	文字列リテラル	はい	昇順

演算子のエイリアス

294ページ「演算子のエイリアス」には演算子の詳細なエイリアスの一覧が記載されています。

演算子のエイリアス

演算子	エイリアス
=	EQ
!=	NE
<	LT
<=	LTE
>	GT
>=	GTE
LIKE	n/a

値のタイプ

289ページ「クエリ・フィルタ条件のプロパティ」にリストされるクエリ・フィルタ条件のプロパティでは，次の値のタイプを使用できます。

値のタイプ

値のタイプ	説明
文字列リテラル	文字列リテラルを使用してフィルタリングする場合，文字列リテラルを二重引用符 ("") で閉じる必要があります。 文字列リテラルでエスケープしなければならない文字のいずれかを使用する必要がある場合 (297ページ「エスケープする必要のある文字に対する URL エスケープ・コード」を参照)，パーセント記号 (%) の代わりにドル記号 ($) を使用してその文字をエスケープする必要があります。たとえば，`query=title EQ '%3CMy title%3E'` の代わりに `query=title EQ "$3CMy title$3E"` を使用します。 **注:** OMi 9.0x およびそれ以下では，シングルの引用符 (') およびパーセント記号 (%) のエスケープ・コードのみがサポートされます。OMi 9.0x では，ドル記号 ($) を使用する文字列リテラルのエスケープ・コードはサポートされません。OMi 9.10 およびそれ以上では，シングル (') およびダブル (") の引用符がサポートされます。イベント Web サービス・クライアントでは二重引用符を使用することをお勧めします。

値のタイプ	説明
日時	XML スキーマの dateTime 形式で指定する必要があります。XML スキーマの日付タイプの詳細については，XML スキーマのマニュアルを参照してください。このマニュアルは次の場所で入手できます。 http://www.w3.org/TR/xmlschema-2 (英語サイト) さらに，「Z」を使用して GMT 以外のタイムゾーンを指定する場合は，「+」記号を指定する必要があります。これは URL 内に指定するため，URL エンコーディングされている必要があります。297ページ「エスケープする必要のある文字に対する URL エスケープ・コード」を参照してください。
整数	0 を含む正および負の自然数。

POST メソッドを使用したクエリ

サーバによって受け入れられる URL では，その最大長が制限される場合があります。この長さを超えると，`Request-URI Too Long` を示す 414 ステータスが返されます。より一般的に，このタイプの制限は媒介手段において発生する場合があります。クライアントが応答でこのステータスを受信した場合，長いクエリ表現がその原因となっている可能性が高いです。クライアントは，次の変更を行い要求を再試行する必要があります。

- HTTP メソッドを `GET` から `POST` に変更する
- Content-Type ヘッダを `application/x-www-form-urlencoded` に変更する
- `query` クエリ・パラメータを URL から削除し，要求の本文を `query` クエリ・パラメータに設定する。たとえば，メッセージの本文を次のようにすることができます。`query=severity='critical'`
- 要求を再送信する

URL エスケープ・コード

297ページ「エスケープする必要のある文字に対する URL エスケープ・コード」には，URL でエスケープする必要のある文字の一覧が示されています。

文字列リテラルの文字をエスケープする必要がある場合，パーセント記号 (%) の代わりにドル記号 ($) を使用します。たとえば，`query=title EQ '%3CMy title%3E'` の代わりに `query=title EQ "$3CMy title$3E"` を使用します。

エスケープする必要のある文字に対する URL エスケープ・コード

文字	URL エスケープ・コード	文字列リテラルのエスケープ・コード
空白	%20	$20
<	%3C	$3C
>	%3E	$3E
#	%23	$23
%	%25	$25
+	%2B	$2B
{	%7B	$7B
}	%7D	$7D
|	%7C	$7C
\	%5C	$5C
^	%5E	$5E
~	%7E	$7E
[	%5B	$5B
]	%5D	$5D
`	%60	$60
;	%3B	$3B
/	%2F	$2F
?	%3F	$3F
:	%3A	$3A
@	%40	$40
=	%3D	$3D
&	%26	$26
$	%24	$24

URL 内の空白

URL のクエリの部分では，次の文字または文字列を使用して空白を表すことができます。

- 空白( )
- プラス記号(+)

- URL エスケープ・コード (%20)
- 文字列リテラル・エスケープ・コード ($20)

複雑な属性

複雑な (ネストされた) 属性を使用してフィルタリングする場合，ネストされていることを示すのに角括弧 ([ ]) を使う必要があります。

複雑な属性のフィルタは，下部属性に対する複数のネストを許可し，他のフィルタと組み合わせることも可能です。

複雑な属性を含んでいる URL の呼び出しの例は次のとおりです。

```
http://<bsmserver.example.com>/opr-console/rest/9.10/event_
list?query=related_ci%5Btarget_
id="ffca22eea15268533029f17b4c01b008"%5D
```

```
http://<bsmserver.example.com>/opr-console/rest/9.10/event_
list?query=custom_attribute_list%5BMyCaName%20NE%20""%5D
```

注: 複雑な属性を指定する際，角括弧文字 ([ ]) をエスケープ処理する必要があります。URL のエスケープ・コードの詳細については，296ページ「URL エスケープ・コード」を参照してください。

編集可能なプロパティ

イベント Web サービス内で編集可能なイベント・プロパティは 298ページ「編集可能なイベント・プロパティの一覧」で一覧表示されます。

注: これは，com.hp.opr.ws.model.event Java API ドキュメンテーションのサマリです。編集可能なプロパティに関する最新の情報については，Java API ドキュメンテーションを参照してください。

Java API ドキュメンテーションのアクセス情報については，302ページ「ファイルの場所」を参照してください。

編集可能なイベント・プロパティの一覧

タグの名前	タイプ	説明	コメント
title	文字列	イベントのタイトル	
description	文字列	イベントの説明	
solution	文字列	ソリューション	

タグの名前	タイプ	説明	コメント
state	文字列	イベントのライフサイクル状態	値は次のうちの 1 つになります。closed, in_progress, open, resolved
severity	文字列	イベントの重要度	値は次のうちの 1 つになります。major, minor, critical, warning, normal, unknown
priority	整数	イベントの優先度	
assigned_user	ユーザ	イベントに割り当てられたユーザ	例 :<name>User</name>
assigned_group	グループ	イベントに割り当てられたユーザ・グループ	例 :<name>Users</name>
cause	ID	原因の識別子	
control_transferred_to	ID または名前	接続サーバの ID または名前	control_transferred_to 構造で設定
symptom_list	一覧	症状の一覧	症状は Symptom Web サービスを使用して作成，更新，削除のみが可能です。
custom_attribute_list	一覧	カスタム属性の一覧	カスタム属性は Custom Attribute Web サービスを使用して作成，更新，削除のみが可能です。
annotation_list	一覧	注釈の一覧	症状は Annotation Web サービスを使用して作成，更新，削除のみが可能です。
auto_action	なし	自動アクション	Automatic Action Web サービスのみを使用して，POST でアクションの実行，DELETE でアクションの停止ができます。XML の本文は空であることに注意してください。

タグの名前	タイプ	説明	コメント
user_action	なし	オペレータのアクション	Operator Action Web サービスのみを使用して，POST でアクションの実行，DELETE でアクションの停止ができます。XML の本文は空であることに注意してください。

履歴行

履歴行では，イベントの変更を追跡できます。履歴行には，タイプ，名前，特定のイベントに対して変更されたイベント・プロパティの以前および現在の値に関する情報が提供されます。同時実行の変更は同じ履歴行として表されます。履歴行は，自動的に生成され，読み取り専用モードで使用できます。

記録されたプロパティの変更

履歴行でのイベント・プロパティの変更時に記録されるイベント・プロパティについては，300ページ「記録されたプロパティの変更」を参照してください。通常，各変更では次のプロパティが記録されます。プロパティ名，以前の値，現在の値，変更時間。これらのプロパティのそれぞれの詳細については，com.hp.opr.api.ws.model.event.property パッケージに一覧表示されている適切な変更のタイプの Javadoc を参照してください。

記録されたプロパティの変更

名前	ChangeType
annotation	OprAnnotationPropertyChange
application	OprStringPropertyChange
assigned_group	OprGroupPropertyChange
assigned_user	OprUserPropertyChange
category	OprStringPropertyChange
cause	OprStringPropertyChange
control_transfer_to	OprContrlTransferredToPropertyChange
correlation_rule	OprCorrelationRulePropChange
custom_attribute	OprCustomAttributePropertyChange
description	OprStringPropertyChange
duplicate_count	OprIntegerPropertyChange
eti_hint	OprStringPropertyChange
key	OprStringPropertyChange

名前	ChangeType
key_pattern	OprStringPropertyChange
node_hint	OprStringPropertyChange
object	OprStringPropertyChange
om_service_id	OprStringPropertyChange
om_user	OprStringPropertyChange
original_data	OprStringPropertyChange
original_id	OprStringPropertyChange
priority	OprPriorityPropertyChange
related_ci_hint	OprStringPropertyChange
severity	OprSeverityPropertyChange
solution	OprStringPropertyChange
source_ci_hint	OprStringPropertyChange
state	OprStatePropertyChange
sub_category	OprStringPropertyChange
time_created	OprTimePropertyChange
time_received	OprTimePropertyChange
title	OprStringPropertyChange
type	OprStringPropertyChange

イベント変更リスト

履歴行は特定のイベントに対して提供されています。そのため，この Web サービス・エンドポイントではフィルタリングがサポートされていません。

イベント変更リストでは，使用可能なすべての履歴行が返されます。イベント変更リストは，次の URL を呼び出すことでアクセスできます。

`http://<bsmserver.example.com>/opr-console/rest/event_changes_list`

変更リストは，標準の HTTP クエリ・パラメータを使用してフィルタリングできます。このリストは，発生したイベントの変更を検出するために使用します。

特定の日付から履歴行を受信する場合は，ウォーターマーク・クエリを指定する必要があります。ウォーターマークおよび日付と時間の指定に関する詳細については，284ページ「日付および時間によるフィルタリング :watermark」を参照してください。

ファイルの場所

イベント Web サービス・インタフェースに関する参照資料ファイルの場所は次のとおりです。

- イベント Web サービス Java API ドキュメント:

  <HPBSM のルート・ディレクトリ>/opr/api/doc/opr-external-api-javadoc.zip

- イベント Web サービス・スキーマ:

  <HPBSM のルート・ディレクトリ>/opr/api/schema/OprDataModel.xsd

第13章

外部イベント・プロセスの統合

本項では，外部プロセスをイベント処理に統合することを目的とした外部アプリケーションとの統合を可能にするインタフェースについて説明します。このインタフェースでは，転送されたイベントおよびそれ以降のイベント変更に関する通知をプログラムで受信できます。

主に，このインタフェースは，HP Service Manager または BMC Remedy Service Desk といった ITIL インシデント・マネージャなどの外部マネージャとイベントおよびイベント変更を同期するために使用します。

本項の内容

イベントの転送およびイベント変更の同期

本項では，外部イベント処理アプリケーション(たとえば，HP Service Manager または BMC Remedy Service Desk などのインシデント・マネージャ) にイベントがどのように転送されるか，転送されたイベントおよび後続のイベント変更がその外部アプリケーションからどのように逆同期されるかについて説明します。

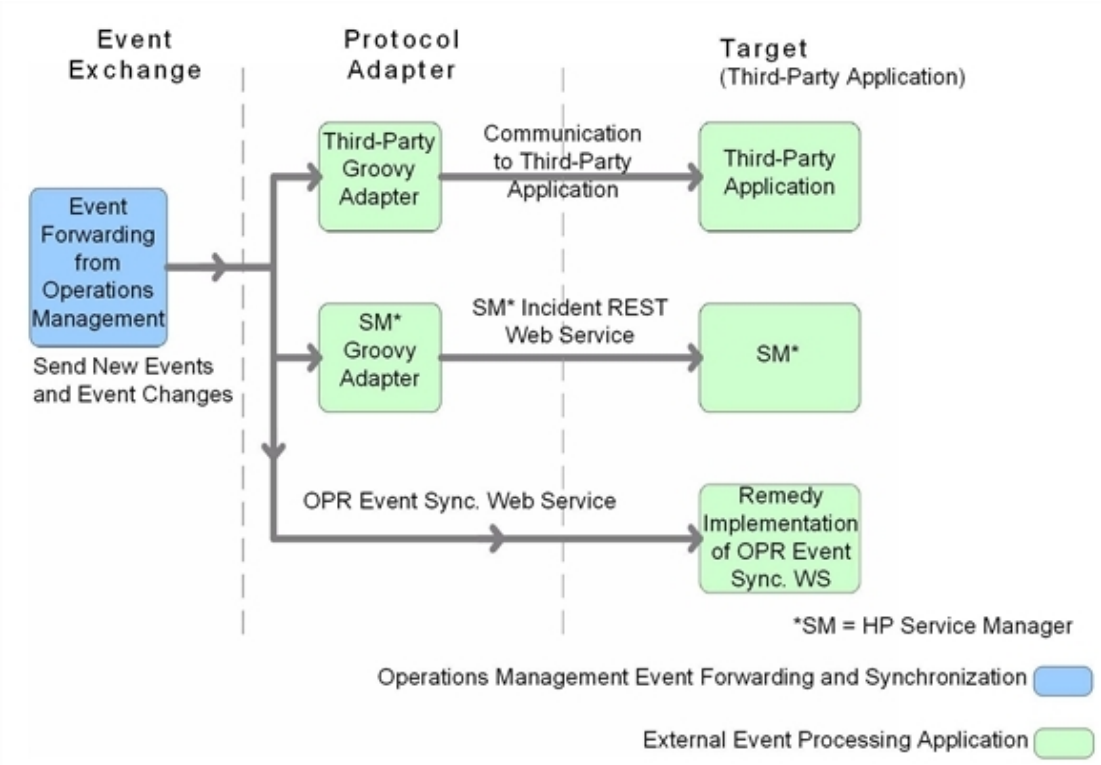
イベント転送および同期プロセス

アプリケーション間のイベントとイベントの変更の同期は次によって異なります。

- 外部イベント・プロセスへのイベントおよびそれ以降の変更の転送
- 外部プロセスからのイベントの変更の逆同期

イベントおよびイベントの変更を外部イベント・プロセスに転送

次の図はイベント転送アーキテクチャの概要と，転送されたイベントがインシデント・マネージャに到達するために取ることが可能な多様なルートを示しています。



転送されたイベントおよびそれ以降のイベントの変更を受信することになるターゲット・サーバは，接続サーバ・マネージャを使用して設定されている必要があります。どのイベントがフィルタに基づいて転送されるか，およびどの接続サーバにイベントが転送されるかを設定することも可能です。イベント転送マネージャでフィルタを設定可能です。

接続サーバの設定方法および転送ルールの設定方法の詳細については，オペレーション管理のオンライン・ヘルプを参照してください。

フィルタに一致するイベントは，イベントに対するそれ以降のすべての変更とともに，ターゲット接続サーバにプッシュされます。次の 4 つの転送モードがサポートされています。

- 通知 :フィルタに一致するイベントは指定した接続サーバに転送されます。
- 通知および更新 :通知と同様ですが，イベントに対するそれ以降の変更もターゲット接続サーバに転送されます。
- 同期 :通知および更新と同様ですが，イベントに対して加えられた変更を逆同期することを期待されていたターゲット接続サーバとの双方向の同期をサポートします。
- 同期してコントロールを移す :同期と同様ですが，イベントのコントロールは接続サーバに移されます。特別な権限があるオペレーション管理ユーザ(管理者など) のみが，コントロールが移された後にイベントをクローズすることが許されます。外部イベント (インシデント) がクローズしたときに接続サーバがそのクローズされた状態を逆同期することが期待されます。転送ルールのオプションとして利用可能なことに加えて，オペレータはイベント・ブラウザのショートカット・メニューを介して手動でコントロールを移すことが可能です。

転送されたイベントおよびそれ以降のイベントの変更の配信は保証されています。イベントが転送されていたり変更が発生するときにターゲット接続サーバがダウンしている場合，要求はキューに格納されて，ターゲット接続サーバが再び利用可能状態になると配信されます。

外部イベント・プロセスを統合するには次の 2 つの方法があります。

- Groovy スクリプト・アダプタの使用
- イベント同期 Web サービスの使用

Groovy スクリプト・アダプタ

標準設定のアダプタをカスタマイズしたり新規アダプタを作成できるように，Groovy スクリプトを作成したり変更できます。標準設定状態で提供される Groovy スクリプト・アダプタは次のとおりです。

- Service Manager アダプタ
- サンプルのログファイル・アダプタ(309ページ「サンプル Groovy スクリプト :ログファイル・アダプタ」を参照)

Groovy スクリプトが接続サーバに対して設定されている場合，一致したイベントおよびそれらのイベントに対するそれ以降の変更をターゲット接続サーバに転送するために Groovy スクリプトが呼び出されます。このスクリプトは，イベントおよびイベントの変更をターゲット接続サーバに配信するために最適な API を使用するように設計されています。たとえば，Service Manager アダプタに対して使用される Groovy スクリプトは Apache Wink REST クライアント API を使用して，HP Service Manager に対する REST Web サービス呼び出しを実行します。

統合のために Groovy スクリプトを使用する方法の詳細については，309ページ「Groovy スクリプトを使用した外部イベント・プロセスの統合」を参照してください。

イベント同期 Web サービス

Groovy スクリプトを実装する代わりに，インテグレータはイベント同期 Web サービスのエンドポイントを実装して，イベント転送要求やそれ以降の更新を直接オペレーション管理から受信することが可能です。Groovy スクリプトではなくイベント REST Web サービスを呼び出すよう接続サーバがオペレーショ

ン管理内で設定されている場合，OPR イベント準拠の REST Web サービスがターゲット・サーバによって実装され，接続サーバのエンドポイントで利用可能であることも同時に期待されます。

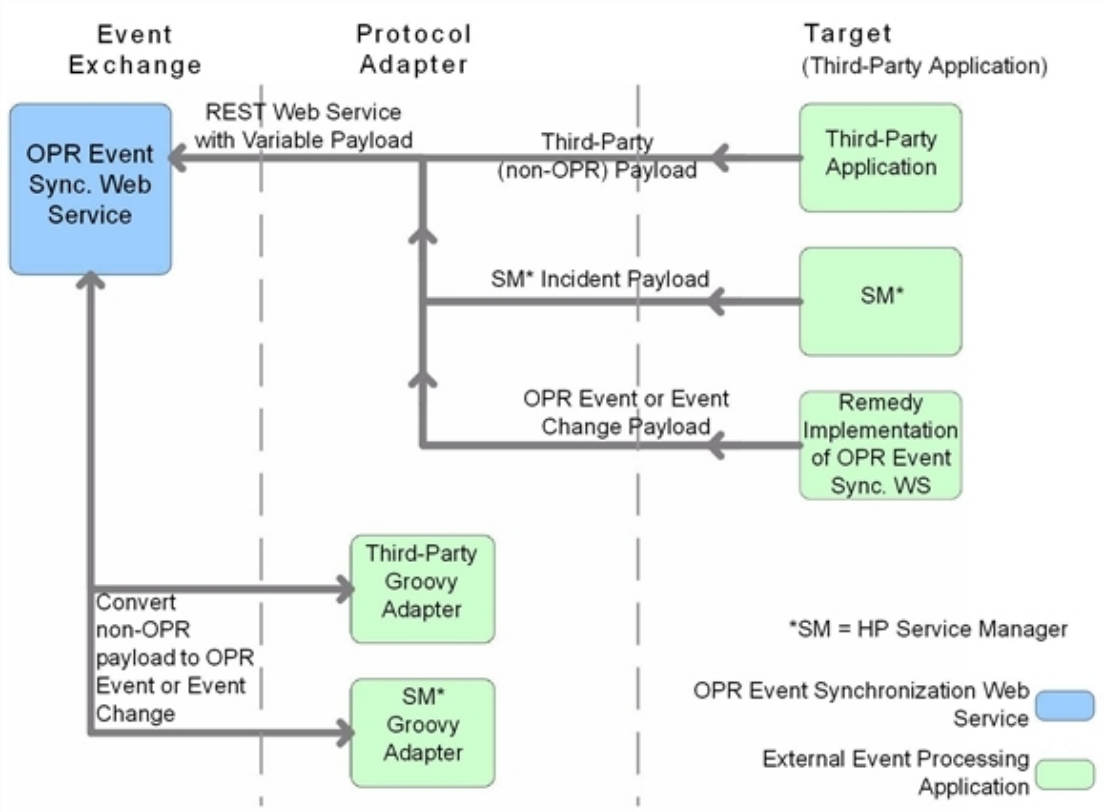
イベントおよびそれ以降のイベントの変更を転送するには，次の標準 REST Web サービスの HTTP メソッド呼び出しがオペレーション管理によって実行されます。

- POST :POST 呼び出しは OPR イベントのペイロードとともにイベントを転送します(ペイロードは要求の本文部分です) 。`/event` パラメータを付加された，ターゲット接続サーバに対して設定されているベース URL は，エンドポイントのアドレスを指定するために使用できます。作成された新規イベントは応答ペイロード内にあることが期待されます。
- POST :POST 呼び出しは，OPR イベント・リストのペイロードとともにイベントのリストを転送します。`/event` パラメータを付加された，ターゲット接続サーバに対して設定されているベース URL は，エンドポイントのアドレスを指定するために使用できます。このエンドポイントはオプションです。指定されると，ユーザは接続サーバの設定で[**バルク転送をサポート**]を選択できます。リスト内の各イベントは一意の `sequence_number` を持ちます。作成されるイベントを含んでいる OPR イベント・リスト内に期待される応答ペイロードがあります。対応するイベントは，どのイベントが作成されたかを識別するよう `sequence_number` が設定されている必要があります。応答で識別されないイベントは，再試行されます。
- POST :POST 呼び出しはイベント変更を OPR イベント変更のペイロードとともに転送します。ターゲット接続サーバに対して設定されたベース URL は，`/event_change/<external_event_ID>` パラメータを付加され，エンドポイントのアドレスを指定するために使用されます。期待される応答ペイロードは OPR イベント変更オブジェクトです。
- POST :POST 呼び出しは一括イベント変更を OPR イベント変更リストのペイロードとともに受信します。このエンドポイントはオプションです。これは接続サーバの設定で[**バルク転送をサポート**]オプションが選択された場合にのみ必要です。`/event_change` パラメータを付加された，ターゲット接続サーバに対して設定されているベース URL は，エンドポイントのアドレスを指定するために使用できます。期待される応答ペイロードは OPR イベント変更リストです。リスト内の各 OprEventChange アイテムは，`target_id` が OPR イベント ID に設定され，`global_target_id` が外部イベント ID に設定されている event ref のイベント参照を持っています。また，各 OprEventChange アイテムは，リストのシーケンスに設定されたシーケンス番号を持っています。応答によって，どのイベント変更が正常に適用されているかを示すようはシーケンス番号が設定されている必要があります。リストから欠落しているイベント変更(対応するシーケンス番号によって識別) は再試行されます。
- GET :GET 呼び出しが外部イベントの現在の状態を取得するために使用されます。`/event/<external_event_ID>` パラメータを付加された，ターゲット接続サーバに対して設定されているベース URL は，エンドポイントのアドレスを指定するために使用できます。期待される応答ペイロードは OPR イベント・オブジェクトです。
- HEAD :HEAD 呼び出しはサービスを PING するために使用されます。これはエンド・ユーザが指定した Web サービス資格情報をチェックするために接続サーバ・マネージャによって使用されます。ターゲット接続サーバに対して設定されたベース URL は，エンドポイントのアドレスを指定するために使用できます。

イベント同期 Web サービスの使用方法の詳細については，327ページ「イベント同期 Web サービス・インタフェース」を参照してください。

外部イベント・プロセスから戻されたイベント変更の受信

次の図はターゲット・アプリケーションがどのように OPR イベント同期 Web サービスと変更を逆同期するかの概要を示しています。



接続サーバの転送タイプを設定するとき，ターゲット・サーバがあらゆる変更をオペレーション管理と逆同期するようにするべきか検討する必要があります。逆同期が必要な場合，転送ルールの設定時に，[同期]転送タイプを指定するか，[同期してコントロールを移す]転送タイプを指定する必要があります。

コントロールを移すことをサポートするよう，ターゲット接続サーバを設定できます。ターゲット・サーバの設定中に[同期およびコントロールの転送をサポート]が選択されると，サーバがイベント・ブラウザのショットカット・メニューで利用可能になり，オペレータによってイベントのコントロールをターゲット接続サーバに移すことが可能になります。コントロールの転送が選択されていない場合，ターゲット・サーバはショットカット・メニューに表示されません。

ターゲット接続サーバがあらゆる変更と逆同期することが期待されている場合，イベント同期 Web サービスを使用してこれらの変更を受信できます。イベント転送の設定時と同様に，Groovy スクリプト・アダプタを使用して Web サービスのペイロードをカスタマイズできます。

HP OMi は，外部のイベント変更を逆同期するための外部アプリケーションによる使用向けに，次の WS PUT メソッドのエンドポイントを追加的に提供します。

PUT :PUT 呼び出しは，外部イベントの一括更新のために使用されます。外部イベントの一括更新用のエンドポイントのアドレスを指定するために，オペレーション管理によって使用される URL は次のようになります。
`http://gatewayhost/opr-gateway/rest/synchronization/event/`

接続サーバに Groovy スクリプトが設定されている場合，Groovy スクリプト `receiveChanges()` メソッドによってペイロードが定義されます。

接続サーバに Groovy スクリプトが設定されていない場合，ペイロードは `OprEventList` オブジェクトである必要があります。

イベント同期 Web サービスの使用方法の詳細については，327ページ「イベント同期 Web サービス・インタフェース」を参照してください。

外部アプリケーションからイベント・ブラウザの URL 起動を実行する

イベント・ブラウザの URL 起動を使用して，外部アプリケーションからオペレーション管理ユーザ・インタフェースにドリルダウンする必要のあるオペレータは，次の項目を満たす必要があります。

- 有効なオペレーション管理ユーザであること。
- 呼び出し側のアプリケーションにログオンし，呼び出しを実行するオペレータの名前がオペレーション管理で設定されたユーザ名と同じであること。

  シングル・サインオン(SSO) 認証が設定されている場合，呼び出し側のアプリケーションにログオンし，URL 呼び出しを実行するためにオペレータが使用するユーザ名と同じユーザ名でオペレーション管理でオペレータを設定します(オペレーション管理オペレータのパスワードは，空または任意の文字列に設定できます) 。呼び出し側アプリケーションへのログインが成功したら，オペレータは追加の認証なしで オペレーション管理のイベント・ブラウザを起動できます。

  呼び出し側アプリケーションが SSO 認証を使用するように設定されていない場合，呼び出し側アプリケーションのオペレータが使用するユーザ名と同じユーザ名を使用してオペレーション管理でオペレータを設定し，有効なパスワードを指定します。オペレータは，オペレーション管理のイベント・ブラウザを起動する際にユーザ名とパスワードを入力する必要があります。
- 必要なアクションを含む「ユーザに割り当てられたイベント」の権限が付与されていること。オプションとして，ユーザに割り当てられていないイベントを表示する権限を付与することができます。

この有効なユーザ名を保有していない場合，必要な表示権限を保有していない場合，呼び出し側アプリケーションからオペレーション管理のイベント・ブラウザで URL 起動を実行しようとすると，空白のブラウザ・ウィンドウが表示されます。

オペレーション管理のイベント・ブラウザの URL 起動の設定方法に関する詳細については，241ページ「イベント・ブラウザの URL 起動」を参照してください。

Groovy スクリプトを使用した外部イベント・プロセスの統合

Groovy スクリプトがサポートされます。Groovy スクリプト・アダプタを使用して，イベントおよびイベント変更をターゲット・サーバに転送できます。本項では，外部イベント処理アプリケーションとの統合に向けて Groovy スクリプトを使用および開発する際に考慮すべき主な統合ポイントに関する情報を提供します。

Groovy および Groovy 言語を説明するドキュメントに関する詳細については，次の URL にアクセスしてください。

http://groovy.codehaus.org(英語サイト)

348ページ「Groovy スクリプトおよびプログラミング」の項で FAQ の回答も参照できます。

サンプル Groovy スクリプト :ログファイル・アダプタ

テンプレートとしてすぐに使用できるサンプル Groovy スクリプトが用意されています。`LogfileAdapter.groovy` というこのサンプル・スクリプトは次のディレクトリにあります。

`<HPBSM のルート・ディレクトリ>/conf/opr/integration/sample`

ログファイル・アダプタを Groovy スクリプト・アダプタとして使用して接続サーバ・マネージャでターゲット接続サーバを設定することができます。ログファイル・アダプタを使用する場合，この設定では任意のホスト名およびポート番号を使用することができます。ただし，Operations Manager i サーバの DNS 名は使用できないことに注意してください。別の名前(`localhost` など) を使用する必要があります。このアダプタに転送されたイベントおよびイベント変更は次のログ・ファイルに記録されます。

`<HPBSM のルート・ディレクトリ>/log/opr/integration/LogfileAdapter.log`

開発者およびインテグレータは，このアダプタをテストに使用したり，テンプレートとして使用して別のアダプタを作成できます。

OMi での外部イベント統合用に新規 Groovy スクリプトを開発する際の開始ポイントとして利用可能なテンプレートの Groovy スクリプトが次の場所に用意されています。

注： `<HPBSM のルート・ディレクトリ>/opr/examples/external-event-adapter`

新規統合を開発するための，「TODO」と明記されたセクションおよび手順が提供されています。

ログファイル・アダプタの使用による接続サーバの設定

オペレーション管理 インスタンスとサード・パーティのイベント・プロセス間のイベントとイベントの変更の同期は，外部イベント処理アプリケーションにイベントを転送するオペレーション管理 によって異なります。これらのイベントおよびイベントの変更は，外部アプリケーションから送り返されることになります。これを成し遂げるための最初の手順は，接続サーバ・マネージャでターゲットの接続サーバを設定することです。

接続サーバの設定方法の完全な詳細情報については，オペレーション管理のオンライン ヘルプを参照してください。

Groovy スクリプト・アダプタとしてログファイル・アダプタを使用してターゲット接続サーバを構成するには，次の手順を行います。

1. オペレーション管理のユーザ・インターフェイスの接続サーバ・マネージャに移動します。

   [**管理**]>[**オペレーション管理**]>[**セットアップ**]>[**接続サーバ**]

2. 新規 ボタンをクリックすると[サーバ接続の新規作成]ダイアログ・ボックスが開きます。

3. [**表示名**]フィールドで，ターゲット接続サーバに名前を入力します。[名前]フィールドは自動的に入力されます。たとえば，ターゲットの HP Service Manager・サーバの表示名として `Logger Example` と入力すると，`Logger_Example` が[名前]フィールドに自動的に挿入されます。

   > **注:** 新規ターゲット・サーバの名前を書き留めます(この例では，`Logger_Example`)。この名前は後で HP Service Manager サーバを設定してオペレーション管理をホストしているサーバとの通信を行えるようにするときに，`username` として提供する必要があります。

   オプション:新規ターゲット・サーバの説明を入力します。

   [アクティブ]チェックボックスを選択していることを確認します。

   [**次へ**]をクリックします。

4. [**外部イベント処理**]を選択して外部イベント処理アプリケーションに適したサーバの種類を選びます。

   [**次へ**]をクリックします。

5. ログファイルのターゲット・サーバの完全修飾 DNS 名を，たとえば `localhost` のように入力します

   [**次へ**]をクリックします。

6. 次に，統合のタイプを確立する必要があります。[統合タイプ]ダイアログ・ボックスで，Groovy スクリプト・アダプタを使用するか，イベント同期 Web サービスを使用するかを選べます。この例では，Groovy スクリプト・アダプタを選択することにします。

   a. [**Groovy スクリプト・アダプタの呼び出し**]を選択します。

   b. [外部イベント処理タイプ]フィールドで[**sample**]を選択します。

   c. [Groovy スクリプトのファイル名]フィールドで，**LogfileAdapter.groovy** を選択します。

      (この場合，外部リソースが必要でないため，[Groovy クラスパス(Groovy Classpath)]フィールドは空白のまま残します。)

   d. [**次へ**]をクリックします。

7. [送信接続]ダイアログ・ボックスは，ターゲット・サーバとの接続を可能にする認証情報(ユーザ名，パスワード，ポート番号)を提供するためのものです。また，そのサーバにイベントを転送します。ログファイル・アダプタの場合には，ユーザ名，パスワード，ポート番号を提供する必要はありません。また，HTTP 設定を選択しなくても構いません。この実演ではこれらのフィールドを空白のままにすることも可能です。

   [送信接続]ダイアログ・ボックスで，次の手順を行います。

   a. [同期してコントロールを移すを有効化(Enable Synchronize and Transfer Control)]チェックボックスが選択されていることを確認します。[同期してコントロールを移す]フラグがオンに設定されると，オペレーション管理のオペレータはイベントの所有権をターゲット接続サーバに移転できます。このフラグが設定されていないと，ルールの転送の設定時に[同期してコント

ロールを移す]オプションが転送タイプのリストに表示されません。

b. [**次へ**]をクリックします。

8. [イベントドリルダウン]ダイアログが開きます。.イベントを純粋にターゲット接続サーバに追加することに加えて，外部イベント処理アプリケーションにドリルダウンできるようにする場合は，完全修飾 DNS 名を指定し，イベント・ドリルダウンを実行するオペレーション管理 システムのポートを指定することが必要になります。この例では，次のように入力します。

   - 完全修飾 DNS 名 :**test.host.com**
   - Port : **80**
   - [**次へ**]をクリックします。

9. 次に行うべきことは，接続サーバから返されるイベントの変更を受信できるようにすることです。このためには，接続サーバ用の認証情報を提供して，オペレーション管理 をホストしているサーバにアクセスする必要があります。

   a. [受信接続]ダイアログ・ボックスで，外部アプリケーションがオペレーション管理 をホストしているサーバに接続するために必要なパスワードを入力します。この例では，`HPpasswd1_` になります。

   b. [**終了**]をクリックします。

   ターゲットの Logger Example サーバは，接続サーバの一覧に表示されます。

イベント転送ルールの設定

次の手順では，どのイベントが `Logger Example` サーバに転送されるかを決めるイベント転送ルールを設定することになります。

フィルタの設定に関する完全な詳細情報については，オペレーション管理 のオンラインヘルプを参照してください。

ルールの転送を設定するには，次の手順を実行します。

1. オペレーション管理 のユーザ・インターフェイスのイベント転送マネージャに移動します。

   [**管理**]>[**オペレーション管理**]>[**イベントの自動化**]>[**イベント転送**]

2. 新規 ボタンをクリックすると[転送ルールの新規作成(New Correlation Rule)]ダイアログ・ボックスが開きます。

3. 表示名フィールドで，転送ルールの名前を入力します。この例では `Forward Major (Sync and Transfer Control)` になります。

   オプション。作成している転送ルールの説明を入力します。

   [アクティブ]チェックボックスが選択されていることを確認します。ルールのステータスがターゲット接続サーバで使用可能になるためには，ルールがアクティブである必要があります。

4. イベント・フィルタ・フィールドの隣の参照ボタンをクリックします。[イベント フィルタを選択]ダイアログが開きます。

5. [イベント フィルタを選択]ダイアログで，新規 ボタンをクリックして[フィルタ構成]ダイアログを開きます。

6. [フィルタ表示名]フィールドで，新規フィルタの名前を入力します。この例では，`FilterMajor` になります。

   重大以外のすべての重大度レベルに対するチェックボックスの選択を解除します。

   [**OK**]をクリックします。

7. 新規フィルタが[イベント フィルタを選択]ダイアログに表示されていることが確認できます(ハイライト表示されていない場合は選択します)。

   [**OK**]をクリックします。

8. ターゲット・サーバのところで，309ページ「サンプル Groovy スクリプト :ログファイル・アダプタ」の項で設定した接続ターゲット・サーバを選択します。この例では，`Logger Example` になります。

   ターゲット・サーバの選択フィールドの隣にある[追加]()ボタンをクリックします。これにより，接続サーバの詳細を確認できるようになります。

   [**OK**]をクリックします。

これで，新規の転送ルールが使用可能になります。

Groovy スクリプト・インタフェース

Groovy スクリプトを使用して，インシデント・マネージャなどの外部イベント・プロセスとの統合を行う場合，次のインタフェースによって定義されるメソッドを実装する Groovy スクリプトを実装する必要があります。

```
com.hp.opr.api.ws.adapter.ExternalProcessAdapter
```

これは，`opr-external-api.jar` という JAR ファイルで提供されます。

すべての引数およびタイプの完全なドキュメントを含む Groovy スクリプト・インタフェースについては，製品に付属する Javadoc ドキュメントを参照してください。Javadoc API ドキュメントの場所については，313ページ「Javadoc API ドキュメント」を参照してください。

重要なファイルの場所

本項では，API ライブラリ，Javadoc API ドキュメント，OPR イベント・スキーマの場所を示します。

API ライブラリ

API ライブラリには，OPR イベントおよびイベント変更オブジェクトの Java Architecture for XML Binding(JAXB)注釈クラスが含まれています。Groovy または Java でプログラミングする場合は，これらのクラスを直接使用することができます。そうでない場合は，OPR イベント・スキーマを使用しなければならない場合があります(313ページ「OPR イベント・スキーマ」を参照)。

API ライブラリは，次のインストール場所にあります。

<HPBSM のルート・ディレクトリ>/lib/opr-external-api.jar

Javadoc APIドキュメント

外部 API インタフェースの詳細については，Javadoc APIドキュメントを参照してください。このドキュメントは次のインストール場所にあります。

<HPBSM のルート・ディレクトリ>/opr/api/doc/opr-external-api-javadoc.zip

OPR イベント・スキーマ

Java 以外のプログラミング言語を使用する場合は，OPR イベント・スキーマを使用して，イベントおよびイベント変更オブジェクトをマーシャリングおよびマーシャリング解除するためのクラスを生成しなければならない場合があります。

OPR イベント・スキーマは次のインストール場所にあります。

<HPBSM のルート・ディレクトリ>/opr/api/schema/OprDataModel.xsd

イベント統合 Groovy スクリプト

イベント統合 Groovy スクリプトを次のインストール場所に保存する必要があります。

<HPBSM のルート・ディレクトリ>/conf/opr/integration/<ディレクトリ>

説明：

上記パスの最後にある <ディレクトリ> は，接続サーバ・マネージャに表示されるアダプタ・タイプを示します。このアダプタ・タイプのすべての Groovy スクリプトは，適切なディレクトリに存在しなければなりません。次の場所には，すぐに使用が可能なスクリプトが用意されています。

ログファイル・アダプタ(タイプ sample):
<HPBSM のルート・ディレクトリ>
/conf/opr/integration/sample/LogfileAdapter.groovy

Service Manager アダプタ(タイプ sm):
<HPBSM のルート・ディレクトリ>
/conf/opr/integration/sm/ServiceManagerAdapter.groovy

Groovy スクリプト・メソッド

Groovy スクリプトを使用して外部イベント処理を統合する場合，その Groovy スクリプトは ExternalProcessAdapter インターフェイスによって定義されたメソッドを実装する必要があります。

Groovy スクリプトによって実装される必要のあるメソッドを以下のセクションに示します。

init

init(def args) メソッドは，Groovy スクリプトが最初に読み込まれたときに呼び出されます。読み込まれたスクリプトはキャッシュされ，何度も呼び出されます。起動時およびスクリプトまたは構成が接続サーバ・マネージャで変更された場合のみ再度読み込まれます。このスクリプトを使用するそれぞれの接続サーバに対して個別のインスタンスが作成されます。

`init(def args)` メソッドには 1 つの引数があります。そのプロパティの一覧については，314ページ「init() メソッドのプロパティ」を参照してください。

init() メソッドのプロパティ

プロパティ	説明
読み取りプロパティ	
`installDir`	BSMオペレーション管理 ルート・ディレクトリ。
`logger`	タイプ: `org.apache.commons.logging.Log`. ログにアクセスするために使用されます。
`connectedServerId`	ターゲット接続サーバの ID。
`connectedServerName`	ターゲット接続サーバの名前。
`connectedServerDisplayName`	ターゲット接続サーバの表示名。
`node`	ターゲット接続サーバの DNS 名。
`port`	タイプ Integer。存在する場合，ターゲット接続サーバの Web サービスのポート番号。
`nodeSsl`	タイプ Boolean。ターゲット接続サーバの Web サービスで SSL が必要かどうかを示します。
`drilldownNode`	ドリルダウンが実行されている可能性のあるターゲット接続サーバの DNS 名。
`drilldownPort`	タイプ Integer。存在する場合，ターゲット接続サーバのドリルダウン・ノード上のドリルダウン・サービスのドリルダウン・ポート。
`drilldownNodeSsl`	タイプ Boolean。ドリルダウン・サービスで SSL が必要かどうかを示します。
`maxTimeout`	接続タイムアウトで使用するミリ秒の数。これは接続サーバ・マネージャで設定します。
書き込みプロパティ	
なし	このメソッドには書き込みプロパティはありません。

destroy

`destroy()` メソッドは，Groovy スクリプトが不要になったときに呼び出されます。引数はありません。

ping

`ping(def args)` メソッドは，設定パラメータが正しいかどうかを判別するために接続サーバ・マネージャによって呼び出されます。指定した読み取りプロパティを使用して接続サーバに達することができる場合，`ping` メソッドによって true が返され，そうでない場合は false が返されます。

315ページ「ping() メソッドのプロパティ」には，ping() メソッドのプロパティの一覧が記載されています。

ping() メソッドのプロパティ

プロパティ	説明
読み取りプロパティ	
`installDir`	BSMオペレーション管理 ルート・ディレクトリ。
`logger`	タイプ: `org.apache.commons.logging.Log.` ログにアクセスするために使用されます。
`connectedServerName`	ターゲット接続サーバの名前。
`connectedServerDisplayName`	ターゲット接続サーバの表示名。
`node`	ターゲット接続サーバの DNS 名。
`port`	タイプ Integer。存在する場合，ターゲット接続サーバの Web サービスのポート番号。
`nodeSsl`	タイプ Boolean。ターゲット接続サーバの Web サービスで SSL が必要かどうかを示します。
`credentials`	タイプ: `java.net.PasswordAuthentication` 存在する場合，ターゲット接続サーバの資格情報。
`locale`	タイプ:Locale 設定されているプロパティの目的のロケール。
書き込みプロパティ	
`outputDetail`	ping 処理の詳細結果。[Connected Servers configuration]ダイアログ・ボックスに目的のロケールで表示されます。

イベントを接続サーバに転送するメソッド

ターゲット接続サーバとして設定される外部イベント処理アプリケーションにイベントを転送するメソッドの一覧を以下のセクションに示します。

forwardEvent

`forwardEvent(def args)` メソッドは，ターゲット接続サーバにイベントを転送するために呼び出されます。このメソッドには 1 つの引数があります。そのプロパティの一覧については，315ページ「forwardEvent() メソッドのプロパティ」を参照してください。

forwardEvent() メソッドのプロパティ

プロパティ	説明
読み取りプロパティ	

プロパティ	説明
credentials	タイプ: java.net.PasswordAuthentication 存在する場合，ターゲット接続サーバの資格情報。
event	タイプ: com.hp.opr.api.ws.model.event.OprEvent 転送されるイベントです。
info	タイプ: com.hp.opr.api.ws.model.event.ci.OprForwardingInf-o 次の転送のタイプに関する情報を保持します。Notify，Notify and Update，Synchronize または Synchronize and Transfer Control。
causeExternalRefI-d	これが症状である場合に，要因が設定され，ターゲット・サーバに転送されていると，このプロパティは要因の外部参照 ID を保持します。そうでない場合は，null が返されます。
読み取りメソッド	
credentials	タイプ: java.net.PasswordAuthentication 存在する場合，ターゲット接続サーバの資格情報。
getEvent(id, includeRefs)	指定したイベントを取得できます。イベントが存在する場合に，アプリケーションによってアクセスされると，そのイベントが返されます。そうでない場合は，null が返されます。呼び出し側が参照の解決を求める場合，includeRefs を true に設定します。そうでない場合は，false に設定します。このフラグの詳細については，イベント WS を参照してください。
getCi(id)	すべてのプロパティを含む，設定項目の詳細を取得します。引数は String タイプです。このメソッドは，すべてのプロパティを含む，指定の設定項目を返します。戻り値は OprConfigurationItem タイプの値です。
書き込みプロパティ	
drilldownUrlPath	オプション。設定すると，イベント・ブラウザで外部アプリケーションへのドリルダウンが有効になります。これは，URL のベース・パスで，ノードまたはポートを含みません。
externalRefId	転送が正常な場合に設定する必要があります。あらゆる文字列値に設定できます。

forwardEvents

`forwardEvents(def args)` メソッドは，イベントのリストをターゲット接続サーバに転送するために呼び出されます。このメソッドには，転送するイベントのリストおよびいくつかのユーティリティ・メソッドを含む単一の引数が渡されます。リスト内の各イベントの転送結果を設定するメソッドもあります。イベントのいずれかに転送結果が設定されていない場合，そのイベントは転送キューに残り，groovy スクリプトの次回の呼び出しでリトライされます。個別のイベントの結果は，成功または失敗に設定できます。このメソッドはオプションです。

`forwardEvent()` メソッドには，groovy スクリプトで容易に `OprForwardingInfo` を取得することを可能にするユーティリティ・メソッドがあります。`BulkForwardEventArgs` は，このユーティリティ・メソッドを直接提供しません。各イベントの `OprForwardingInfo` を取得するには，各イベントで `getForwardingInfo(final String serverId)` メソッドを呼び出すことができます。`serverId` は，この接続サーバの ID であり，`InitArgs` で groovy スクリプトに渡されます。

このメソッドには，1 つの引数 (`BulkForwardEventArgs`) があります。そのプロパティについては，317ページ「forwardEvents() メソッドのプロパティ」を参照してください。

forwardEvents() メソッドのプロパティ

プロパティ	説明
読み取りプロパティ	
`credentials`	タイプ: `java.net.PasswordAuthentication` 存在する場合，ターゲット接続サーバの資格情報。
`events`	タイプ: `com.hp.opr.api.ws.model.event.OprEventList` 転送されるイベントのリストです。
読み取りメソッド	
`getEvent(id, includeRefs)`	指定したイベントを取得できます。イベントが存在する場合に，アプリケーションによってアクセスされると，そのイベントが返されます。そうでない場合は，null が返されます。呼び出し側が参照の解決を求める場合，`includeRefs` を true に設定します。そうでない場合は，false に設定します。このフラグの詳細については，イベント WS を参照してください。
`getCi(id)`	すべてのプロパティを含む，設定項目の詳細を取得します。引数は String タイプです。このメソッドは，すべてのプロパティを含む，指定の設定項目を返します。戻り値は `OprConfigurationItem` タイプの値です。
`getCauseExternalRefId(symptomId)`	外部参照 ID を取得します。引数は String タイプです。このメソッドは，指定の症状に `cause external reference ID` が存在する場合，それを返します。
書き込みメソッド	

プロパティ	説明
`setForwardSuccess (eventId, externalRefId, crossLaunchUrlPath)`	このメソッドでは，個別のイベントについて，成功，外部参照 ID およびその外部イベントのクロス起動 URL を設定できます。
`setForwardFailed (reason)`	一括転送が失敗した場合，このメソッドを呼び出して転送の失敗の原因を記述する `Throwable` を渡すことができます。原因は null にすることが可能です。リスト全体が失敗としてマークされ，リトライが実行されません。
`setForwardFailed (eventId, reason)`	リスト内の個別のイベントを失敗に設定します。このイベントはリトライされません。オプションの要因を指定するか，null に設定できます。 特定のイベントに対して `setSuccess()` または `setFailed()` が呼び出されない場合，そのイベントは後でリトライされます。

forwardChange

`forwardChange(def args)` メソッドは，イベントの変更をターゲット接続サーバに転送するために呼び出されます。このメソッドには 1 つの引数があります。そのプロパティの一覧については，318ページ「forwardChange() メソッドのプロパティ」を参照してください。

forwardChange() メソッドのプロパティ

プロパティ 読み取りプロパティ	説明
`changes`	タイプ: `com.hp.opr.api.ws.model.event.OprEventChange` 転送するイベントの変更。
`credentials`	タイプ: `java.net.PasswordAuthentication` 存在する場合，ターゲット接続サーバの資格情報。
`externalRefId`	タイプ:String 更新する必要のある外部イベントの ID。
`info`	タイプ: `com.hp.opr.api.ws.model.event.ci.OprForwardingInfo` 次の転送のタイプに関する情報を保持します。`Notify`，`Notify and Update`，`Synchronize` または `Synchronize and Transfer Control`。

プロパティ	説明
event	現在の状態でイベントのコピーを返します。イベントの属性には，変更が発生した時点の状態ではなく，この呼び出しが実行された時点のイベントの状態が反映されます。
causeExternalRefId	これが症状である場合に，要因が設定され，ターゲット・サーバに転送されていると，このプロパティは要因の外部参照 ID を保持します。そうでない場合は，null が返されます。
読み取りメソッド	
getEvent(id, includeRefs)	指定したイベントを取得できます。イベントが存在する場合に，アプリケーションによってアクセスされると，そのイベントが返されます。そうでない場合は，null が返されます。呼び出し側が参照の解決を求める場合，includeRefs を true に設定します。そうでない場合は，false に設定します。このフラグの詳細については，イベント WS を参照してください。
getCi(id)	すべてのプロパティを含む，設定項目の詳細を取得します。引数は String タイプです。このメソッドは，すべてのプロパティを含む，指定の設定項目を返します。戻り値は OprConfigurationItem タイプの値です。
書き込みプロパティ	
なし	このメソッドには書き込みプロパティはありません。

forwardChanges

forwardChanges(def args) メソッドは，イベントの変更をターゲット接続サーバに転送するために呼び出されます。このメソッドには，転送するイベント変更のリストおよびいくつかのユーティリティ・メソッドを含む単一の引数が渡されます。リスト内の各イベント変更項目の転送結果を設定するメソッド セットもあります。イベント変更のいずれかに転送結果が設定されていない場合，そのイベント変更は転送キューに残り，次回の呼び出しでリトライされます。個別のイベント変更の結果は，成功または失敗に設定できます。このメソッドはオプションです。

このメソッドには，1 つの引数 (BulkForwardChangeArgs) があります。そのプロパティについては，319ページ「forwardChanges() メソッドのプロパティ」を参照してください。

forwardChanges() メソッドのプロパティ

プロパティ	説明
読み取りプロパティ	
changes	タイプ: com.hp.opr.api.ws.model.event.OprEventChangeList 転送するイベントの変更。

プロパティ	説明
credentials	タイプ: java.net.PasswordAuthentication 存在する場合，ターゲット接続サーバの資格情報。
読み取りメソッド	
getCi(id)	すべてのプロパティを含む，設定項目の詳細を取得します。引数は String タイプです。このメソッドは，すべてのプロパティを含む，指定の設定項目を返します。戻り値は OprForwardingInfo タイプの値です。
getEvent(id, includeRefs)	指定したイベントを取得できます。イベントが存在する場合に，アプリケーションによってアクセスされると，そのイベントが返されます。そうでない場合は，null が返されます。呼び出し側が参照の解決を求める場合，includeRefs を true に設定します。そうでない場合は，false に設定します。このフラグの詳細については，イベント WS を参照してください。
getCauseExternalRefId(symptomId)	外部参照 ID を取得します。引数は String タイプです。このメソッドは，指定の症状に cause external reference ID が存在する場合，それを返します。
OprForwardingInfo getForwardingInfo (change)	すべてのプロパティを含む，転送情報の詳細を取得します。引数は String タイプです。このメソッドは，指定のイベント変更の OprForwardingInfo を取得します。 外部参照 ID は，この転送情報から取得できます。
書き込みメソッド	
setForwardSuccess()	リスト内のすべての変更に対して成功を設定します。
setForwardSuccess (changesId)	リスト内の指定の変更に対して成功を設定します。
setForwardFailed (reason)	リスト内のすべての変更を失敗に設定します。要因は例外の形式で指定できます。または null を指定できます。失敗に設定された変更はリトライされません。
setForwardFailed (changeId, reason)	指定した変更を失敗に設定します。この変更はリトライされません。 特定の変更に対して setSuccess() または setFailed() が呼び出されない場合，その変更は後でリトライされます。

接続サーバから同期データを受信するメソッド

ターゲット接続サーバとして設定される外部イベント処理アプリケーションから同期データを受信するメソッドの一覧を以下のセクションに示します。

注: これらのメソッドは，接続サーバが接続サーバからイベントの変更を同期しなおす機能をサポートしている場合のみ必要です。

receiveChange

`receiveChange(def args)` メソッドは，接続サーバによってイベント同期 Web サービスにイベント変更が送信されたときに呼び出されます。このメソッドには 1 つの引数があります。そのプロパティの一覧については，321ページ「receiveChange() メソッドのプロパティ」を参照してください。

receiveChange() メソッドのプロパティ

プロパティ	説明
読み取りプロパティ	
`event`	タイプ:OprEvent 変更が適用される前の現在の状態におけるイベントの読み取り専用コピー。読み取り専用です。イベントを更新するには，引数の書き込みプロパティのいずれかを設定します。
`externalEventChange`	タイプ:String 変更が接続サーバから同期しなおされた場合に，その接続サーバから受信する Web サービス呼び出しのペイロード(PUT または POST)。
`info`	タイプ: `com.hp.opr.api.ws.model.event.ci.OprForwardingInfo` 次の転送のタイプに関する情報を保持します。`Notify`，`Notify and Update`，`Synchronize` または `Synchronize and Transfer Control`。
`locale`	タイプ:Locale Web サービスの呼び出し側の目的のロケール。
`externalRefId`	タイプ:String 更新する必要のある外部イベントの ID。
`causeExternalRefId`	これが症状である場合に，要因が設定され，ターゲット・サーバに転送されていると，このプロパティは要因の外部参照 ID を保持します。そうでない場合は，null が返されます。
読み取りメソッド	
`getEvent(id, includeRefs)`	指定したイベントを取得できます。イベントが存在する場合に，アプリケーションによってアクセスされると，そのイベントが返されます。そうでない場合は，null が返されます。呼び出し側が参照の解決を求める場合，`includeRefs` を true に設定します。そうでない場合は，false に設定します。このフラグの詳細については，イベント WS を参照してください。
`getCi(id)`	すべてのプロパティを含む，設定項目の詳細を取得します。引数は String タイプです。このメソッドは，すべてのプロパティを含む，指定の設定項目を返します。戻り値は `OprConfigurationItem` タイプの値です。

プロパティ 書き込みプロパティ	説明
eventId	イベントの ID。
title	イベントのタイトル。
description	イベントの説明。
solution	イベントのソリューション。
severity	イベントの重要度。
priority	イベントの優先度。
state	イベントの状態。 closed または resolved に設定すると，イベントの状態が closed に設定されます。イベントをその他の値に設定すると，イベントの状態が open に設定されます。
cause	要因イベントの ID。
assignedUser	タイプ: com.hp.opr.api.ws.model.event.OprUser イベントの根本的な問題の解決を担当するユーザの名前。
assignedGroup	タイプ: com.hp.opr.api.ws.model.event.OprGroup イベントの割り当て済みユーザが属しているグループの名前。

追加のメソッド

イベントのコントロール，注釈，カスタム属性，設定項目を移行したり，HTTP 要求および応答にアクセスするための追加のメソッドが提供されています。

コントロールの移行

イベントのコントロールの移行に使用できるメソッドを次に示します。

- requestControl

  requestControl メソッドは，イベントのコントロールに対する要求です。いったん要求が実行されると，そのイベントは別のサーバによって所有されることができなくなります。要求がキューされます。要求が完了すると，呼び出し側は，control_transferred_to イベント・プロパティの変更通知を受信します。

- returnControl

  returnControl メソッドは，イベントのコントロールを外部イベント プロセスからオペレーション管理に戻します。コントロールを戻すには，呼び出し側がイベントのコントロールを所有している必要があります。要求がキューされます。要求が完了すると，呼び出し側は，control_transferred_to イベント・プロパティの変更通知を受信します。

注釈

注釈に使用できるメソッドを次に示します。注釈を追加したり更新することはできますが，削除することはできません。

- `addAnnotation(def author, def text)`

  `addAnnotation(def author, def text)` メソッドは注釈を追加します。引数は String タイプです。

- `updateAnnotations(def id, def author, def text)`

  `updateAnnotation(def id, def author, def text)` メソッドは注釈を更新します。引数は String タイプです。

カスタム属性

カスタム属性に使用できるメソッドを次に示します。

- `addCustomAttribute(def name, def value)`

  `addCustomAttribute(def name, def value)` メソッドはカスタム属性を追加します。引数は String タイプです。

- `updateCustomAttribute(def name, def value)`

  `updateCustomAttribute(def name, def value)` メソッドはカスタム属性を更新します。引数は String タイプです。

HTTP 要求および応答

3 つのメソッドが，HTTP 要求または応答にアクセスするために提供されます。

- `getHttpRequestHeader(def name)`

  `getHttpRequestHeader(def name)` メソッドはヘッダの値のリストを返します。引数は String タイプです。

- `setHttpResponseStatus(def httpResponseCode, def httpResponseMessage)`

  `setHttpResponseStatus(def httpResponseCode, def httpResponseMessage)` メソッドは，応答状態およびペイロードを制御するために呼び出されます。デフォルトは 202 およびペイロードなしです。コードは Integer タイプのコードで，メッセージは String タイプのメッセージです。

- `setHttpResponseHeader(def name, def value)`

  `setHttpResponseHeader(def name, def value)` メソッドは，指定した HTTP 応答ヘッダの設定を有効にします。引数は String タイプです。

receiveChanges

`receiveChanges(def args)` メソッドは，接続サーバによってイベント同期 Web サービスに複数のイベント変更が送信されたときに呼び出されます。このメソッドには，外部処理サーバによってイベント同期 WS エンドポイント (/opr-gateway/rest/9.10/event_change) に送信されたペイロード，およびいくつかのユーティリティ・メソッドを含む単一の引数が渡されます。ペイロールには，変更のリストが含ま

れていることが期待されます。変更のリストの解釈およびイベントへの適用は，groovy スクリプト・メソッドによって決定されます。これは，変更される各イベントの ReceiveChangeArgs オブジェクトを作成し，そのオブジェクトで変更を設定するユーティリティ・メソッドを介して実行されます。

このメソッドには，1 つの引数 (`BulkReceiveChangeArgs`) があります。そのプロパティについては，324ページ「receiveChanges() メソッドのプロパティ」を参照してください。

receiveChanges() メソッドのプロパティ

プロパティ	説明
読み取りプロパティ	
`externalEventChanges`	タイプ:String 1 つまたはそれ以上の変更が接続サーバから同期しなおされた場合に，その接続サーバから受信する Web サービス呼び出しのペイロード (PUT または POST)。
`locale`	タイプ:Locale Web サービスの呼び出し側の目的のロケール。
読み取りメソッド	
`getEvent(id, includeRefs)`	指定したイベントを取得できます。イベントが存在する場合に，アプリケーションによってアクセスされると，そのイベントが返されます。そうでない場合は，null が返されます。呼び出し側が参照の解決を求める場合，`includeRefs` を true に設定します。そうでない場合は，false に設定します。このフラグの詳細については，イベント WS を参照してください。
`getCi(id)`	すべてのプロパティを含む，設定項目の詳細を取得します。引数は String タイプです。このメソッドは，すべてのプロパティを含む，指定の設定項目を返します。戻り値は `OprConfigurationItem` タイプの値です。
`getgetHttpRequestHeader(name)`	指定したヘッダを取得します。ヘッダは複数回指定することができます。指定したすべてのインスタンスが返されます。
`getReceiveChangeArgs(id)`	指定したイベントの `ReceiveChangeArgs` オブジェクトを取得します。このオブジェクトは，変更されるイベントのリストに自動的に追加されます。`ReceiveChangeArgs` によって提供されるメソッドを呼び出して，指定のイベントに適切な変更を適用することができます。

接続サーバからデータを受信するメソッド

接続サーバからデータを受信するメソッドの一覧を以下のセクションに示します。

getExternalEvent

`getExternalEvent()` メソッドは，オペレータが外部イベントの現在の表現を要求したときに呼び出されます。オペレータがイベント・ブラウザで[イベント詳細]を開き，[外部情報]タブを選択するか，イベント・ブラウザのコンテキスト・メニューを使用すると，`getExternalEvent()` メソッドが呼び出され，外部イベントの最新のコピーが取得され，選択したフィールドがイベント・ブラウザに表示されま

す。このメソッドには 1 つの引数があります。そのプロパティの一覧については，325ページ「getExternalEvent() メソッドのプロパティ」を参照してください。

getExternalEvent() メソッドのプロパティ

プロパティ	説明
読み取りプロパティ	
externalRefId	取得する外部イベントを識別する外部参照 ID。
credentials	タイプ: java.net.PasswordAuthentication 存在する場合，ターゲット接続サーバの資格情報。
locale	タイプ:Locale 設定されているプロパティの目的のロケール。
書き込みプロパティ(すべて String タイプ)	
title	オプションの外部イベントのタイトル。
description	オプションの外部イベントの説明。
assignedUser	オプションの外部イベントに割り当てられたユーザ。
assignedGroup	オプションの外部イベントに割り当てられたユーザ・グループ
severity	オプションのイベントの重要度。
priority	オプションのイベントの優先度。
state	オプションのイベントの状態。

接続サーバにデータを供給するメソッド

ターゲット接続サーバにデータを供給するメソッドも存在します。

toExternalEvent

toExternalEvent() メソッドは，OPR イベントを外部イベント・オブジェクトに変換します。このメソッドは，接続サーバが OPR イベントの最新コピーについてイベント同期 Web サービスにクエリを実行したときに呼び出されます。このメソッドにより，インテグレータはイベントを外部イベント表現に変換できます。すなわち，GET 形式 (.../event/<event id>) の応答ペイロードを作成できます。

このメソッドには 1 つの入力パラメータがあります。
com.hp.opr.api.ws.model.event.OprEvent

このメソッドの戻り値は文字列です。これは，Web サービスを呼び出したアプリケーションに返されるペイロードです。

Groovy スクリプト・メソッドの管理

Groovy スクリプト メソッドは，別途の Java セキュリティ・マネージャで実行されます。Java セキュリティ・マネージャでは，次の権限が拒否されます。

- `System.exit()` 呼び出しへのアクセス
- ターゲット exitVM の `java.lang.RuntimePermission` のみ

  http://java.sun.com/javase/6/docs/api/java/lang/RuntimePermission.html

イベント同期 Web サービス・インタフェース

イベント同期 Web サービス・インタフェースを使用すると，HP Service Manager のようなインシデント・マネージャなどの外部 (サードパーティ) イベント・プロセスと統合できます。このインタフェースは，オペレーション管理 OPR クライアントからサードパーティ・アプリケーションにイベントおよびイベント変更を転送したり，イベントおよび外部アプリケーションによるイベント変更をサードパーティ・クライアントから逆同期するために使用します。

外部イベント・プロセスは，Groovy スクリプトを実装する代わりに OPR 準拠イベント同期 Web サービスを実装することで直接 オペレーション管理 イベント処理と統合することが可能です。外部アプリケーションは，OPR イベントおよびイベント変更を受信するための Web サービス・エンドポイント，および外部アプリケーションが同期をサポートする場合は，イベント同期 Web サービスにイベントを送信したり，イベント変更を逆同期するための REST Web サービス・クライアントを実装する必要があります。

OPR クライアントからのイベントおよびイベント変更の転送

OPR クライアントでイベントおよびイベント変更をサードパーティのイベント処理アプリケーションに転送するには，そのサードパーティ・アプリケーションで OPR 準拠イベント同期 Web サービスが実装されている必要があります。

イベントの転送

OPR クライアントからサードパーティ・アプリケーションにイベントを転送する場合，次のデータが関連します。

- HTTP メソッド :POST
- ベース URL :`http://<接続サーバの設定されたベース URL>/event`
- 期待されるペイロード :期待される応答ペイロードは `OprEvent` オブジェクトです。

イベント変更の転送

OPR クライアントからサードパーティ・アプリケーションにイベント変更を転送する場合，次のデータが関連します。

- HTTP メソッド :POST
- ベース URL :`http://<接続サーバの設定されたベース URL>/event_change/<external_event_ID>`
- 期待されるペイロード :期待される応答ペイロードは `OprEventChange` オブジェクトです。

イベントの転送

OPR クライアントからサードパーティ・アプリケーションにイベントを一括して転送する場合，次の

データが関連します。

- HTTP メソッド :POST
- ベース URL :`http://<接続サーバの設定されたベース URL>/event`
- 期待されるペイロード :ペイロードは `OprEventList` オブジェクトです。
- 返されるペイロード :成功した場合，ターゲット・サービスは ID が外部イベント ID に設定された状態で，送られてきたイベントのリストを返す必要があります。`sequence_number` は，POST で送信されたイベントの `sequence_number` に一致している必要があります。一致するイベントが返されていないイベントについては，後続の要求で再試行されます。1 つのイベントが拒否された場合，ペイロード全体に対して HTTP エラーが返される必要があります。HTTP エラー 500 ～ 599 が返された場合，各イベントは一括要求としてではなく個別に再試行されます。HTTP エラー 400 ～ 499 の場合，要求のすべてのイベントが失敗としてマークされ，再試行は実行されません。

イベント変更の一括転送

OPR クライアントからサード パーティ・アプリケーションにイベント変更を一括して転送する場合，次のデータが関連します。

- HTTP メソッド :POST
- ベース URL :`http://<接続サーバの設定されたベース URL>/event_change/`
- 期待されるペイロード :期待される応答ペイロードは `OprEventChangeList` オブジェクトです。リスト内の各 `OprEventChange` 項目には，`event_ref` プロパティが設定されている必要があります。`global_id` はターゲット・システムの ID に設定されることが期待されます。たとえば，オペレーション管理 がターゲット・システム上 のこの Web サービスを呼び出すと，`target_id` は オペレーション管理のイベント ID に設定され，`target_global_id` はターゲット・オブジェクト ID に設定されます。
- 返されるペイロード :成功した場合，ターゲット・システムは，そこに送られてきたイベント変更のリストを同じ `sequence_number` で返す必要があります。`sequence_number` は，POST で送信されたイベント変更の `sequence_number` と一致する必要があります。一致するイベント変更が返されていないイベント変更については，後続の要求で再試行されます。1 つのイベント変更が拒否された場合，ペイロード全体に対して HTTP エラーが返される必要があります。HTTP エラー 500 ～ 599 が返された場合，各イベント変更は一括要求としてではなく個別に再試行されます。HTTP エラー 400 ～ 499 の場合，要求のすべてのイベント変更が失敗としてマークされ，再試行は実行されません。

Web サービス GET 要求

外部イベントをサード パーティ・アプリケーションから取得する必要がある場合 (たとえば，イベント・ブラウザから[外部情報]タブを選択した場合)，次のデータが関連します。

- HTTP メソッド :GET
- ベース URL : `http://<接続サーバの設定されたベース URL>/event/<external_event_ID>`
- 期待されるペイロード :ペイロードは，`OprEvent` の形式の外部イベントです。

Web サービス ping 要求

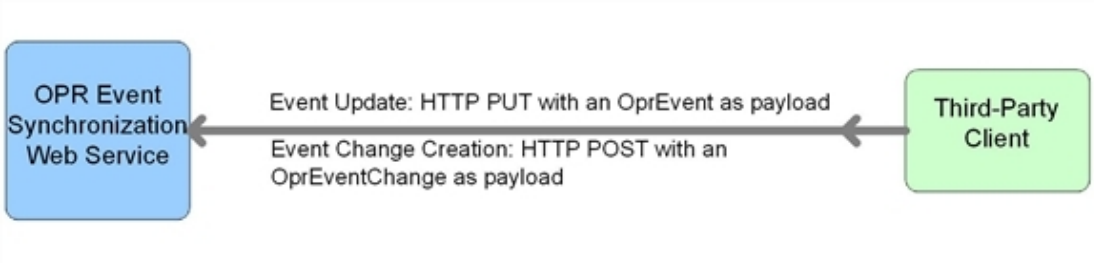
サード パーティ・アプリケーションに対して ping 要求を送信できます。サード パーティ・アプリケーションに ping 要求を送信する場合，次のデータが関連します。

- HTTP メソッド :ベース URL への HEAD
- ベース URL :
  - イベントを転送するためのベース URL :`http://<接続サーバの設定されたベース URL>`
- 期待されるペイロード :なし

外部クライアントからのイベント変更の逆同期

外部アプリケーションが同期をサポートする場合，REST Web サービス・クライアントはイベント変更をイベント同期 Web サービスに対して逆同期する必要があります。この逆同期のペイロードは，ネイティブ OPR オブジェクトに準拠することが期待されます。接続サーバに Groovy スクリプトが設定されている場合，その Groovy スクリプトによって，クライアントが定義したペイロードが解釈されます。

クライアントが Groovy スクリプトを呼び出さずに新規イベントまたはイベントに対する変更を送信する場合，`event_list` および `event_change_list` のサブパスを使用する必要があります。これらのパスは，ネイティブ OPR オブジェクト (たとえば，`OprEvent` または `OprEventChangeList`) を必要とします。`event_list` および `event_change_list` サブパスを使用すると，クライアントは 1 つまたは複数のイベントまたはイベント変更を送信できます。



イベントの更新

イベントの取得

イベントの取得では，次のデータが関連します。

- HTTP メソッド :GET
- ベース URL :`http://<bsmserver.example.com>/opr-gateway/rest/9.10/synchronization/event/<event_id>`
- 返されるペイロード :
  - 接続サーバに Groovy スクリプトが設定されている場合，Groovy スクリプトの `toExternalEvent()` メソッドによってペイロードが定義されます。
  - 接続サーバに Groovy スクリプトが設定されていない場合，ペイロードは `OprEvent` オブジェクトです。

イベントの更新

イベントの更新では，次のデータが関連します。

- HTTP メソッド :PUT
- ベース URL :`http://<bsmserver.example.com>/opr-gateway/rest/9.10/synchronization/event/<event_id>`
- Ping 要求 (サードパーティ・クライアントから) :ベース URL への HTTP HEAD
- 受信ペイロード :
  - 接続サーバに Groovy スクリプトが設定されている場合，Groovy スクリプトによってペイロードが定義されます。
  - 接続サーバに Groovy スクリプトが設定されていない場合，ペイロードは `OprEvent` オブジェクトである必要があります。

333ページ「イベント更新 :ログファイル・アダプタの例」も参照してください。

イベント変更の作成

イベント変更では，次のデータが関連します。

- HTTP メソッド :POST
- ベース URL : `http://<bsmserver.example.com>/rest/9.10/synchronization/event_change/<event_id>`
- Ping 要求 (サードパーティ・クライアントから) :ベース URL への HTTP HEAD
- 受信ペイロード :
  - 接続サーバに Groovy スクリプトが設定されている場合，Groovy スクリプトによってペイロードが定義されます。
  - 接続サーバに Groovy スクリプトが設定されていない場合，ペイロードは `OprEventChange` オブジェクトである必要があります。

335ページ「イベント変更の作成 :ログファイル・アダプタの例」も参照してください。

Web サービス ping 要求

サードパーティ・アプリケーションからも ping 要求を送信できます。サードパーティ・アプリケーションから ping 要求を送信する場合，次のデータが関連します。

- HTTP メソッド :HEAD
- ベース URL :`http://<bsmserver.example.com>/opr-gateway/rest/9.10/synchronization`
- 期待されるペイロード :なし

イベント・リストの更新

次の URL は，`event_list` サブパスをポイントします。このサブパスに対する要求では，Groovy スクリプトは呼び出されません。この要求は，`OprEvent` および `OprEventList` などの ORP データ構造の入力または出力を常に保有します。

イベントの取得

イベントの取得では，次のデータが関連します。

- HTTP メソッド :GET
- ベース URL :`http://<bsmserver.example.com>/opr-gateway/rest/9.10/synchronization/event_list/<event_id>`
- 返されるペイロード :
  - ペイロードは常に `OprEvent` オブジェクトです。
  - `event_list` サブパスに対して Groovy スクリプトは呼び出されません。

イベントの更新

イベントの更新では，次のデータが関連します。

- HTTP メソッド :PUT
- ベース URL :`http://<bsmserver.example.com>/opr-gateway/rest/9.10/synchronization/event_list/<event_id>`
- 受信ペイロードおよび返されるペイロード :
  - 受信ペイロードは `OprEvent` オブジェクトである必要があります。
  - 返されるペイロードは常に `OprEvent` オブジェクトです。
  - `event_list` サブパスに対して Groovy スクリプトは呼び出されません。

341ページ「event_list に対するイベント更新の例」も参照してください。

イベントの送信

サードパーティ・アプリケーションからイベントを送信することもできます。サードパーティ・アプリケーションからイベントを送信する場合，次のデータが関連します。

- HTTP メソッド :POST
- ベース URL :`http://<bsmserver.example.com>/opr-gateway/rest/9.10/synchronization/event_list`
- 受信ペイロード :
  - ペイロードは `OprEvent` または `OprEventList` オブジェクトである必要があります。
  - `OprEventList` オブジェクトの場合，メディア・タイプは `text/xml;type=collection` または `application/xml;type=collection` でなくてはなりません。
  - `event_list` サブパスに対して Groovy スクリプトは呼び出されません。

341ページ「イベントの送信の例」も参照してください。

イベント・リストの変更

次の URL は，`event_change_list` サブパスをポイントします。このサブパスに対する要求では，Groovy スクリプトは呼び出されません。この要求は，`OprEvent` および `OprEventList` などの ORP データ構造の入力または出力を常に保有します。

イベント変更の作成

`event_change_list` サブパスに対するイベント変更では，次のデータが関連します。

- HTTP メソッド :POST
- ベース URL :`http://<bsmserver.example.com>/opr-gateway/rest/9.10/synchronization/event_change_list/`
- 受信ペイロード :
  - 受信ペイロードは `OprEventChange` または `OprEventChangeList` オブジェクトである必要があります。
  - `OprEventChangeList` オブジェクトの場合，メディア・タイプは `text/xml;type=collection` または `application/xml;type=collection` でなくてはなりません。
  - `event_change_list` サブパスに対して Groovy スクリプトは呼び出されません。

343ページ「event_change_list のイベント変更作成の例」も参照してください。

接続サーバの設定

イベント同期 Web サービスの使用時に行う一般的な手順をまとめると，次のようになります。

1. ターゲット・サーバを設定します。これは接続サーバ・マネージャで行います。サーバの名前はイベント同期 Web サービスの認証時に使用されなくてはならない名前です。接続サーバの設定中にこのユーザに対するパスワードを指定できます。
2. イベントをターゲット接続サーバに転送します。これは，手動でイベントの転送を行うためにイベント・ブラウザのコンテキスト・メニューを使うことで実行可能です。または，転送ルールを設定し，そのルールをトリガすることでも可能です。
3. イベントがターゲット接続サーバに転送されるまで，サーバは変更内容を同期し戻すことはできません。認証されたオペレーション管理のユーザがアクセス可能なあらゆるイベントについて，それらのユーザに表示および更新を許可する，イベント Web サービス(252ページ「イベント Web サービス・インタフェースを使用したオペレータ機能の自動化」を参照)と対照的に，接続サーバはまずサーバに転送されてきたイベントに対する変更内容を同期し戻すことのみが可能です。
4. 接続サーバはイベントに対する変更内容を同期し戻します。

接続サーバの設定方法の詳細については，オペレーション管理のオンラインヘルプを参照してください。

逆同期をサポートするイベント属性

サポートされた属性の更新

	サポートされている操作		
属性名	**更新**	**作成**	**削除**
annotation	o	o	
assigned_user	o		NULL に設定
assigned_group	o		NULL に設定
cause	o		NULL に設定

属性名	サポートされている操作		
	更新	作成	削除
custom attribute	o	o	
description	o	o	NULL に設定
priority	o		
severity	o		
solution	o		NULL に設定
state	o		
title	o		NULL に設定
control_transferred_to	o		NULL に設定

イベント更新 :ログファイル・アダプタの例

注: 本項のイベント更新を示している例はログファイル・アダプタ(製品に付属のサンプルの Groovy スクリプト・アダプタ)特有のもので，ログファイル・アダプタに対してのみ，またはターゲット接続サーバに対して設定された Groovy スクリプトなしで直接イベント同期 Web サービスにアクセスする場合のみに有効です。

イベント更新とともにイベント変更を送信できます。この場合の HTTP メソッドは PUT で，ログファイル・アダプタの期待されるペイロードは `OprEvent` です。各 Groovy スクリプト・アダプタは自身の期待されるペイロードを定義し，期待されるペイロードがほかの Groovy スクリプト・アダプタと異なるようにします。

イベント同期 Web サービスは 1 つのペイロードでの複数のプロパティの送信をサポートします。次の例はそれぞれ 1 つの変更されたプロパティを示しています。最後のサンプルだけが例外で，2 つの変更したプロパティを示しています(335ページ「1 つの呼び出しによる複数のプロパティの変更」を参照)。

製品に付属の，REST Web サービスのコマンドライン・ユーティリティ `RestWsUtil` を使用してテスト・ペイロードをイベント同期 Web サービスに送信できます。このユーティリティを使用するときに，更新しようとしているイベントがまず，イベント同期 Web サービスで認証を行うのに使用しているターゲット接続サーバに転送されたことを確認します。

`RestWsUtil` コマンドライン・ユーティリティの詳細については，274ページ「REST Web サービス・コマンドライン・ユーティリティ」を参照してください。

イベント更新の呼び出しの例を次に示します。次のイベント更新のサンプル呼び出しは，名前が `logger` でパスワードが `Password1` に設定された接続サーバを対象とし，ID `0695624b-93fa-40b1-8b0fc9b4ea07a4ec` のイベントの更新を試みます。サンプル呼び出しは次のようになります。

```
RestWsUtil -update update.xml-url http://<bsmserver.example.com>/opr-gateway/rest/9.10/synchronization/event/0695624b-93fa-40b1-8b0fc9b4ea07a4ec -username logger -password Password1 -verbose
```

次のサンプルは呼び出しに対して有効な XML ペイロードを示します。ペイロードは `RestWsUtil` コマンドライン・ユーティリティに対する呼び出しで参照される `update.xml` ファイルに含まれています。

注: イベント REST WS のサンプルは次の場所にあります。

```
<HPBSM>/opr/examples/event-ws
```

これは，J2EE コンテナへのデプロイメント用にコンパイルして .war ファイルにパッケージ化することが可能です。この Web アプリケーションをビルドする方法およびテスト用にデプロイする方法については，このディレクトリの README.txt を参照してください。

イベント REST WS のテンプレートは次の場所にあります。

```
<HPBSM>/opr/examples/synchronization-ws
```

これは独自のイベント同期 Web サービスを開発しようとする開発者が開始点として使えるものです。この Web アプリケーションをビルドする方法については，このディレクトリの README.txt を参照してください。「TODO」セクションは開発者が完成させる必要があります。

イベントの説明の設定および設定解除

イベント同期 Web サービスはイベントの説明に対する設定または設定解除の操作のみをサポートします。挿入，更新，削除の操作はサポートされません。

説明の設定

これは説明を設定するサンプル・ペイロードです。

```
<event xmlns="http://www.hp.com/2009/software/opr/data_model">
<description>This is a new description</description> </event>
```

説明の設定解除

これは説明を設定解除するサンプル・ペイロードです。

```
<event xmlns="http://www.hp.com/2009/software/opr/data_model">
<description/> </event>
```

イベント・タイトルの設定および設定解除

イベント同期 Web サービスはイベントのタイトルに対する設定または設定解除の操作のみをサポートします。挿入，更新，削除の操作はサポートされません。

タイトルの設定

これはタイトルを設定するサンプル・ペイロードです。

```
<event xmlns="http://www.hp.com/2009/software/opr/data_model">
<title>This is a new title</title> </event>
```

タイトルの設定解除

これはタイトルを設定解除するサンプル・ペイロードです。

```
<event xmlns="http://www.hp.com/2009/software/opr/data_model">
<title/> </event>
```

1 つの呼び出しによる複数のプロパティの変更

イベント同期 Web サービスは 1 つのペイロードでの複数のプロパティの送信をサポートします。ここでは，2 つの変更したプロパティだけを示します。

タイトルと説明の設定

タイトルと説明は 1 つの呼び出しで設定可能です。属性の順序は重要ではなく，指定されていない属性は現在の値に影響をおよぼしません。

これは，1 つの更新の呼び出しでイベントのタイトルと説明を設定するサンプル・ペイロードです。

```
<event xmlns="http://www.hp.com/2009/software/opr/data_model">
<title>This is a new title</title> <description>This is a new
description</description> </event>
```

イベント変更の作成 :ログファイル・アダプタの例

イベント変更の作成とともにイベント変更を送信できます。この場合の HTTP メソッドは POST で，ログファイル・アダプタの期待されるペイロードは `OprEventChange` です。本項の例で，サポートされる更新それぞれに対するサンプルの XML ペイロードを示します。Groovy スクリプトが接続サーバに対して設定されていない場合，イベント同期 Web サービスは `OprEventChange` オブジェクトを期待します。後述の各サンプルは OprEventChange オブジェクトを表しています。

イベント同期 Web サービスは 1 つのペイロードでの複数のプロパティの送信をサポートします。次の例はそれぞれ 1 つの変更されたプロパティを示しています。最後のサンプルだけが例外で，2 つの変更したプロパティを示しています(340ページ「1 つの呼び出しによる複数のプロパティの変更」を参照)。

これらの例では `531d2673-683f-429f-a742-b8680ee01a76` のイベント ID を使用します。このイベント ID を，変更オブジェクトの作成対象となる特定のイベントの ID に変更します。イベント変更の作成のための正しい HTTP メソッドは POST メソッドです。

イベント変更リストの Web サービスを使用してもイベント変更オブジェクトの例を取得できます。次の URL を使用してイベント変更リストの Web サービスにアクセスできます。

```
http://<bsmserver.example.com>/opr-console/rest/9.10/event_change_list
```

イベント変更作成の呼び出しの例を次に示します。

次に示す，イベント変更作成のサンプル呼び出しは，パスワードが `Password1` に設定された `logger` という名前の接続サーバ用のもので，ID `531d2673-683f-429f-a742-b8680ee01a76` を持つイベントに対するイベント変更の作成を試みます。イベント変更の場合，イベント ID はペイロードで渡されている必要があり，URL の一部ではありません。サンプル呼び出しは次のようになります。

```
RestWsUtil -create create.xml-url http://<bsmserver.example.com>/opr-
gateway/rest/9.10/synchronization/event_change/<イベント ID>-username
logger -password Password1 -verbose
```

次のサンプルは呼び出しに対して有効な XML ペイロードを示します。ペイロードは `RestWsUtil` コマンドライン・ユーティリティに対する呼び出しで参照される `create.xml` ファイルに含まれています。

注釈

イベント同期 Web サービスは，注釈に対しては挿入と更新の操作のみをサポートします。削除操作はサポートされません。

注釈の挿入

これは注釈の挿入を示すサンプル・ペイロードです。

```
<event_change xmlns="http://www.hp.com/2009/software/opr/data_model"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema"> <event_ref> <target_
id>531d2673-683f-429f-a742-b8680ee01a76</target_id> </event_ref>
<changed_properties> <annotation_property_change> <property_
name>annotation</property_name> <current_value
xsi:type="xs:string">This is a new Annotation.</current_value>
<change_operation>insert</change_operation> </annotation_property_
change> </changed_properties> </event_change>
```

注釈の更新

これは ID `1c108839-9584-45f4-93ab-798bf49797c7` の注釈に対する更新を示すサンプル・ペイロードです。

```
<event_change xmlns="http://www.hp.com/2009/software/opr/data_model"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema"> <event_ref> <target_
id>531d2673-683f-429f-a742-b8680ee01a76</target_id> </event_ref>
<changed_properties> <annotation_property_change> <property_
name>annotation</property_name> <annotation_id>1c108839-9584-45f4-
93ab-798bf49797c7</annotation_id> <current_value
xsi:type="xs:string">This is an updated Annotation.</current_value>
<change_operation>update</change_operation> </annotation_property_
change> </changed_properties> </event_change>
```

要因

イベント同期 Web サービスはイベントの要因に対する設定または設定解除の操作のみをサポートします。挿入，更新，削除の操作はサポートされません。症状イベントに要因を設定したり設定解除します。

イベントの要因の設定

これはイベントの要因を ID `9915e504-f5a2-471a-ab98-6184a7382d32` のイベントに設定する方法を示すサンプル・ペイロードです。

```
<event_change xmlns="http://www.hp.com/2009/software/opr/data_model"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema"> <event_ref> <target_
id>531d2673-683f-429f-a742-b8680ee01a76</target_id> </event_ref>
<changed_properties> <string_property_change> <property_
name>cause</property_name> <current_value
xsi:type="xs:string">9915e504-f5a2-471a-ab98-6184a7382d32</current_
value> </string_property_change> </changed_properties> </event_change>
```

イベントの要因の設定解除

これはイベントの要因を設定解除する方法を示すサンプル・ペイロードです。

```
<event_change xmlns="http://www.hp.com/2009/software/opr/data_model"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema"> <event_ref> <target_
id>531d2673-683f-429f-a742-b8680ee01a76</target_id> </event_ref>
<changed_properties> <string_property_change> <property_
name>cause</property_name> <current_value xsi:type="xs:string"/>
</string_property_change> </changed_properties> </event_change>
```

カスタム属性

イベント同期 Web サービスは，カスタム属性に対しては挿入と更新の操作のみをサポートします。削除操作はサポートされません。

カスタム属性の挿入

これは，カスタム属性 `custom_attribute` を値 `Some CA value` で挿入する方法を示すサンプル・ペイロードです。

```
<event_change xmlns="http://www.hp.com/2009/software/opr/data_model"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema"> <event_ref> <target_
id>531d2673-683f-429f-a742-b8680ee01a76</target_id> </event_ref>
<changed_properties> <custom_attribute_property_change> <property_
name>custom_attribute</property_name> <key>TestCA</key> <current_value
xsi:type="xs:string">Some CA value</current_value> <change_
operation>insert</change_operation> </custom_attribute_property_
change> </changed_properties> </event_change>
```

カスタム属性の更新

これは，カスタム属性 `custom_attribute` を値 `Some updated CA value` で更新する方法を示すサンプル・ペイロードです。

```
<event_change xmlns="http://www.hp.com/2009/software/opr/data_model"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema"> <event_ref> <target_
id>531d2673-683f-429f-a742-b8680ee01a76</target_id> </event_ref>
<changed_properties> <custom_attribute_property_change> <property_
name>custom_attribute</property_name> <key>TestCA</key> <current_value
xsi:type="xs:string">Some updated CA value</current_value> <change_
operation>update</change_operation> </custom_attribute_property_
change> </changed_properties> </event_change>
```

説明

イベント同期 Web サービスはイベントの説明に対する設定または設定解除の操作のみをサポートします。挿入，更新，削除の操作はサポートされません。

説明の設定

これは，イベント変更の作成の説明を `Some description of the event` に設定する方法を示すサンプル・ペイロードです。

```
<event_change xmlns="http://www.hp.com/2009/software/opr/data_model"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema">   <event_ref>
    <target_id>531d2673-683f-429f-a742-b8680ee01a76</target_id>
  </event_ref>   <changed_properties>     <string_property_change>
      <property_name>description</property_name>       <current_value
xsi:type="xs:string">Some description of the event.      </current_
value>     </string_property_change>   </changed_properties> </event_
change>
```

説明の設定解除

これはイベント変更の作成の説明を設定解除する方法を示すサンプル・ペイロードです。

```
<event_change xmlns="http://www.hp.com/2009/software/opr/data_model"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema"> <event_ref> <target_
id>531d2673-683f-429f-a742-b8680ee01a76</target_id> </event_ref>
<changed_properties> <string_property_change> <property_
name>description</property_name> <current_value xsi:type="xs:string"/>
</string_property_change> </changed_properties> </event_change>
```

状態

イベントの状態に対して有効な操作は，状態に新しい値を与えることのみです。イベントの状態の変更で，挿入，更新，削除の操作は適用されません。

イベントの状態として有効な値は `open`, `in_progress`, `resolved`, `closed` です。

状態の設定

これはイベント変更の作成の状態を `in_progress` の値に設定する方法を示すサンプル・ペイロードです。

```
<event_change xmlns="http://www.hp.com/2009/software/opr/data_model"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema"> <event_ref> <target_
id>531d2673-683f-429f-a742-b8680ee01a76</target_id> </event_ref>
<changed_properties> <state_change> <property_name>state</property_
name> <current_value xsi:type="xs:string">in_progress</current_value>
</state_change> </changed_properties> </event_change>
```

優先度

イベントの優先度に対して有効な操作は，優先度に新しい値を与えることのみです。優先度の変更で，挿入，更新，削除の操作は適用されません。

優先度は `none`, `lowest`, `low`, `medium`, `high`, `highest` のいずれかの値に設定できます。

優先度の設定

これはイベント変更の作成の優先度を low に設定する方法を示すサンプル・ペイロードです。

```
<event_change xmlns="http://www.hp.com/2009/software/opr/data_model"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema"> <event_ref> <target_
id>531d2673-683f-429f-a742-b8680ee01a76</target_id> </event_ref>
<changed_properties> <priority_property_change> <property_
name>priority</property_name> <current_value
xsi:type="xs:string">low</current_value> </priority_property_change>
</changed_properties> </event_change>
```

重大度

イベントの重大度に対して有効な操作は，重大度に新しい値を与えることのみです。重大度の変更で，挿入，更新，削除の操作は適用されません。

重大度は critical, major, minor, warning, normal のいずれかの値に設定できます。

重大度の設定

これはイベント変更の作成の重大度を normal に設定する方法を示すサンプル・ペイロードです。

```
<event_change xmlns="http://www.hp.com/2009/software/opr/data_model"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema"> <event_ref> <target_
id>531d2673-683f-429f-a742-b8680ee01a76</target_id> </event_ref>
<changed_properties> <severity_property_change> <property_
name>severity</property_name> <current_value
xsi:type="xs:string">normal</current_value> </severity_property_
change> </changed_properties> </event_change>
```

ソリューション

イベント同期 Web サービスはイベントのソリューションに対する設定または設定解除の操作のみをサポートします。ソリューションの変更で，挿入，更新，削除の操作は適用されません。

ソリューションの設定

これは，イベント変更の作成のソリューションを Some solution to the event に設定する方法を示すサンプル・ペイロードです。

```
<event_change xmlns="http://www.hp.com/2009/software/opr/data_model"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema"> <event_ref> <target_
id>531d2673-683f-429f-a742-b8680ee01a76</target_id> </event_ref>
<changed_properties> <string_property_change> <property_
name>solution</property_name> <current_value xsi:type="xs:string">Some
solution to the event.</current_value> </string_property_change>
</changed_properties> </event_change>
```

ソリューションの設定解除

これはイベント変更の作成のソリューションを設定解除する方法を示すサンプル・ペイロードです。

```
<event_change xmlns="http://www.hp.com/2009/software/opr/data_model"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema"> <event_ref> <target_
id>531d2673-683f-429f-a742-b8680ee01a76</target_id> </event_ref>
<changed_properties> <string_property_change> <property_
name>solution</property_name> <current_value xsi:type="xs:string"/>
</string_property_change> </changed_properties> </event_change>
```

タイトル

イベント同期 Web サービスはイベントのタイトルに対する設定または設定解除の操作のみをサポートします。タイトルの変更で，挿入，更新，削除の操作は適用されません。

タイトルの設定

これは，イベント変更の作成のタイトルを `Some title for the event` に設定する方法を示すサンプル・ペイロードです。

```
<event_change xmlns="http://www.hp.com/2009/software/opr/data_model"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema"> <event_ref> <target_
id>531d2673-683f-429f-a742-b8680ee01a76</target_id> </event_ref>
<changed_properties> <string_property_change> <property_
name>title</property_name> <current_value xsi:type="xs:string">Some
title for the event.</current_value> </string_property_change>
</changed_properties> </event_change>
```

タイトルの設定解除

これはイベント変更の作成のタイトルを設定解除する方法を示すサンプル・ペイロードです。

```
<event_change xmlns="http://www.hp.com/2009/software/opr/data_model"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema"> <event_ref> <target_
id>531d2673-683f-429f-a742-b8680ee01a76</target_id> </event_ref>
<changed_properties> <string_property_change> <property_
name>title</property_name> <current_value xsi:type="xs:string"/>
</string_property_change> </changed_properties> </event_change>
```

1 つの呼び出しによる複数のプロパティの変更

イベント同期 Web サービスは 1 つのペイロードでの複数のプロパティの送信をサポートします。ここでは，2 つの変更したプロパティを示します。

タイトルと説明の設定

これは，1 つの更新の呼び出しでイベントのタイトルと説明を設定するサンプル・ペイロードです。

```
<event_change xmlns="http://www.hp.com/2009/software/opr/data_model"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema"> <event_ref> <target_
id>531d2673-683f-429f-a742-b8680ee01a76</target_id> </event_ref>
<changed_properties> <string_property_change> <property_
name>title</property_name> <current_value xsi:type="xs:string">Some
title for the event.</current_value> </string_property_change>
<string_property_change> <property_name>description</property_name>
<current_value xsi:type="xs:string">Some description for the
event.</current_value> </string_property_change> </changed_properties>
</event_change>
```

event_list に対するイベント更新の例

OPR データ構造のみを使用してイベントを更新可能です(Groovy スクリプトは呼び出しません) 。OPR データ構造を使用してイベントを更新するには, `event_list` サブパスを使用して更新を呼び出します。ペイロードは, たとえば標準出力と同じです。

タイトルと説明は, 次のように 1 つの呼び出しで設定できます。属性の順序は重要ではなく, 指定されていない属性は現在の値に影響をおよぼしません。

```
<event xmlns="http://www.hp.com/2009/software/opr/data_model">
   <title>This is a new title</title>
   <description>This is a new description</description>
</event>
```

イベントの送信の例

URL `http://<bsmserver.example.com>/opr-console/rest/9.10/event_list` を使用して新規イベントを送信できます。この場合の HTTP メソッドは POST メソッドで, 期待されるペイロードは `OprEvent` または `OprEventList` です。`OprEventList` オブジェクトの場合, メディア・タイプは `text/xml;type=collection` または `application/xml;type=collection` でなくてはなりません。

送信側によって `OprForwardingInfo` エントリがイベント内で追加されない場合, `ForwardingType` が notify に設定された状態で, 自動的に 1 つのエントリが追加されます。これにより, 呼び出し側は以降, イベント同期 Web サービスからイベントを読み取ることが可能になります。

次のサンプルは, XML ペイロードを示すものです。設定する正しい HTTP メソッドは POST メソッドです。

新規イベントの送信

これは 1 つのイベントの送信を示しているサンプル・ペイロードです。

```
<event xmlns="http://www.hp.com/2009/software/opr/data_model">
   <title>This is a new event</title>
   <description>This is a description</description>
</event>
```

同期の要求を含む新規イベントの送信

これは同期の要求とともに送信される 1 つのイベントを示しているサンプル・ペイロードです。

```
<event xmlns="http://www.hp.com/2009/software/opr/data_model">
   <title>This is a new event</title>
   <description>This is a description</description>
   <forwarding_info_list>
      <forwarding_info>
         <dns_name>extclt.example.com</dns_name>
         <external_id>IM10219</external_id>
         <external_url>http://extclt.example.com:8081/webtier-
9.20/index.do?ctx=docEngine&file=probsummary&query=number%3D%22IM10219-
%22</external_url>
         <forwarding_type>synchronize</forwarding_type>
         <state>originated</state>
      </forwarding_info>
   </forwarding_info_list>
</event>
```

同期してコントロールを移す要求を含む新規イベントの送信

これは同期してコントロールを移す要求とともに送信される 1 つのイベントを示しているサンプル・ペイロードです。

```
<event xmlns="http://www.hp.com/2009/software/opr/data_model">
   <title>This is a new event</title>
   <description>This is a description</description>
   <forwarding_info_list>
      <forwarding_info>
         <dns_name>extclt.example.com</dns_name>
         <external_id>IM10219</external_id>
         <external_url>http://extclt.example.com:8081/webtier-
9.20/index.do?ctx=docEngine&file=probsummary&query=number%3D%22IM10219-
%22</external_url>
         <forwarding_type>synchronize_and_transfer_
control</forwarding_type>
         <state>originated</state>
      </forwarding_info>
   </forwarding_info_list>
</event>
```

新規イベントのリストの送信

`OprEventList` オブジェクトのメディア・タイプは `text/xml;type=collection` または `application/xml;type=collection` である必要があります。

これはイベントのリストの送信を示しているサンプル・ペイロードです。

```
<event_list xmlns="http://www.hp.com/2009/software/opr/data_model">
   <event><title>e0</title><severity>critical</severity></event>
   <event><title>e1</title><severity>normal</severity></event>
   <event><title>e2</title><severity>major</severity></event>
   <event><title>e3</title><severity>minor</severity></event>
   <event><title>e4</title><severity>warning</severity></event>
</event_list>
```

event_change_list のイベント変更作成の例

URL `http://<bsmserver.example.com>/opr-gateway/rest/9.10/synchronization/event_change_list` を使用して新規イベント変更を送信できます。この場合の HTTP メソッドは POST メソッドで，期待されるペイロードは `OprEventChange` または `OprEventChangeList` です。`OprEventChangeList` オブジェクトの場合，メディア・タイプは `text/xml;type=collection` または `application/xml;type=collection` でなくてはなりません。

次のサンプルは，XML ペイロードを示すものです。設定する正しい HTTP メソッドは POST メソッドです。

新規イベント変更の送信

これはタイトル変更の送信を示しているサンプル・ペイロードです。

```
<event_change xmlns="http://www.hp.com/2009/software/opr/data_model"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <event_ref>
      <target_id>531d2673-683f-429f-a742-b8680ee01a76</target_id>
   </event_ref>
   <changed_properties>
      <string_property_change>
         <property_name>title</property_name>
         <current_value xsi:type="xs:string">Some title for the
event.</current_value>
      </string_property_change>
   </changed_properties>
</event_change>
```

新規イベント変更のリストの送信

メディア・タイプは `text/xml;type=collection` または `application/xml;type=collection` である必要があります。

これは 1 つのイベントのタイトル変更，および別のイベントの状態変更の送信を示しているサンプル・ペイロードです。

```
<event_change xmlns="http://www.hp.com/2009/software/opr/data_model">
<event_change xmlns="http://www.hp.com/2009/software/opr/data_model"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

```
   <event_ref>
      <target_id>531d2673-683f-429f-a742-b8680ee01a76</target_id>
   </event_ref>
   <changed_properties>
      <string_property_change>
         <property_name>title</property_name>
         <current_value xsi:type="xs:string">Some title for the
event.</current_value>
      </string_property_change>
   </changed_properties>
   </event_change>
<event_change xmlns="http://www.hp.com/2009/software/opr/data_model"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <event_ref>
      <target_id>ad726f4f-ba51-409f-b429-b445b791ac9d</target_id>
   </event_ref>
   <changed_properties>
      <state_change>
         <property_name>state</property_name>
         <current_value xsi:type="xs:string">in_progress</current_
value>
      </state_change>
   </changed_properties>
</event_change>
</event_change_list>
```

外部イベント・プロセスの統合 :FAQ

本項では，外部イベント・プロセスの統合に関する FAQ を示します。この FAQ は，外部イベント処理に対して設定されている接続サーバ用のスクリプトを作成する開発者を支援することを目的としています。

本項の内容

はじめに

本項では，作業を開始するにあたって必要とされる基本情報に関する FAQ を示します。

利用可能なドキュメントはありますか?

はい。本ガイド(『Operations Manager i 拡張性ガイド』)をお読みください。

- イベントの転送および同期処理に関する情報については，304ページ「イベントの転送およびイベント変更の同期」を参照してください。
- イベント同期 Web サービスに関する情報については，327ページ「イベント同期 Web サービス・インタフェース」を参照してください。
- Groovy スクリプトを使用した統合に関する情報については，309ページ「Groovy スクリプトを使用した外部イベント・プロセスの統合」を参照してください。

API の Javadoc はありますか?

はい。Javadoc は次の場所にあります。

`<HPBSM のルート・ディレクトリ>/opr/api/doc/opr-external-api-javadoc.zip`

このファイルは解凍する必要があります。

OprEvent および OprEventChange オブジェクトで利用できるスキーマはありますか?

はい。スキーマは次の場所にあります。

`<HPBSM のルート・ディレクトリ>/opr/api/schema/OprDataModel.xsd`

スキーマは必要ですか?

Groovy スクリプトを使用して外部アプリケーションを統合する場合，OprEvent スキーマは通常必要ありません。Groovy スクリプトの介入なしで `OprEvent` および `OprEventChange` オブジェクトを使用してイベント同期 Web サービスのエンドポイントと直接通信するために統合を行う場合に，スキーマが必要になります。たとえば，(スキーマに記述された) `OprEvent` オブジェクトの HTTP POST メソッド呼び出しを受け入れる REST Web サービスを実装し，`OprEventChange` オブジェクトの HTTP

POST メソッド呼び出しまたは OprEvent オブジェクトの HTTP PUT メソッド呼び出しのいずれかを実行して，変更されたこれらのオブジェクトをイベント同期 Web サービスに送信する REST クライアントを実装する場合です。この場合，接続サーバは，Groovy スクリプト統合ではなく，Web サービス統合向けに設定されます。

さらに，クライアントまたはサービスが Java で記述されている場合，opr-external-api.jar ファイルの Java Architecture XML Binding(JAXB) 注釈オブジェクトを直接使用して，XML をマーシャリングまたはマーシャリング解除できます。別の言語でプログラミングしている場合のみ，スキーマを直接使用する必要があります。

利用可能な WSDL(Web サービス記述言語) はありますか?

いいえ。イベント同期 Web サービスはシンプルな REST ベースの Web サービスです。一般的に，REST ベースの Web サービスには WSDL はありません。

REST に関する追加情報はどこを参照すればよいですか?

次の Wikipedia エントリを参照してください。

http://ja.wikipedia.org/wiki/REST

REST クライアントまたはサーバを作成するために役に立つツールキットはありますか?

次の場所で入手可能な Apache Wink ツールキットを使用してください。

http://incubator.apache.org/wink/index.html(英語サイト)

OprEvent および OprEventChange ですでに利用可能な JAXB 注釈クラスはありますか?

はい。<HPBSM のルート・ディレクトリ>/lib/opr-external-api.jar ファイルに JAXB 注釈クラスが含まれています。Java でプログラミングしている場合は，スキーマからクラスを生成する代わりに，これらのクラスを直接使用できます。提供されているクラスの詳細については，Javadoc API ドキュメントを参照してください。

サンプルの実装はありますか?

はい。次の場所にあるサンプルを参照してください。

<HPBSM のルート・ディレクトリ>
/conf/opr/integration/sample/LogfileAdapter.groovy

これは，転送要求を受け入れ，それをログ・ファイルに書き込むサンプルの Groovy スクリプト・アダプタです。

どのようにログファイル・アダプタのサンプルを設定および使用したらよいですか?

このアダプタを設定するには，次のパラメータを使用して接続サーバを設定します。

- ノード : localhost(DNS 名を使用しないでください)
- ポート : 標準 http の場合は 80，SSL が有効な場合は 443
- ドリルダウン・ノード : <使用しているオペレーション管理ノードの DNS 名>
- ドリルダウン・ポート : 標準 http の場合は 80，SSL が有効な場合は 443
- [**同期および転送コントロールを有効化**]を選択

接続サーバを設定した後に，イベント・ブラウザからログファイル・アダプタに手動でイベントを転送できます。イベント・ブラウザのショットカット・メニューから[**コントロールを次に移す**]を選択してください。イベント転送マネージャを使用して自動転送ルールを設定することもできます。

イベントが転送されると，転送済みイベントの[イベント詳細]に[外部情報]という新しいタブが表示されます。このタブを選択すると，`LogfileAdapter.groovy` スクリプトの `getExternalInfo()` メソッドが呼び出され，このイベントの外部情報が取得され，タブに表示されます。呼び出しが成功すると，左側のフィールドにデータが表示されます。

イベント同期 Web サービスにアクセスしてイベントを更新するクライアントを作成しましたが，常に「HTTP 401, Unauthorized」エラー・メッセージを受信します。どのように認証したらよいですか?

接続サーバにパスワードが設定されていることを確認します。このパスワードは，ウィザードの最後のページまたは[受信接続]ページ上に表示されるパスワードです。このページには，ユーザ名も表示されます。

更新対象のイベントが，認証しようとしている接続サーバに正常に転送されていることを最初に確認してください。

Groovy 統合スクリプトを作成しました。どのようにインストールおよびテストしたらよいですか?

- 次の場所に新しいディレクトリを作成します。

  `<HPBSM のルート・ディレクトリ>/conf/opr/integration`

- その新しいディレクトリにスクリプトを追加します。
- 新規の外部処理接続サーバを設定し，ドロップダウン・メニューからスクリプトを選択します。
- 「`my_forward_test`」というタイトルの転送ルールを作成し，ターゲット接続サーバにイベントを自動的に転送させます。
- `sendEvent` ツールを使用してテスト・イベントを作成します。

  `<HPBSM のルート・ディレクトリ>/opr/support/sendEvent.bat -t "my_forward_test"`

- イベントがサーバに転送されたかどうかを確認します。

接続サーバの[送信接続]画面で[**同期および転送コントロールを有効化**]を選択することもできます。その場合は，イベント・ブラウザからイベントを手動で転送できます。次に，[外部情報]タブを選択して，外部イベントを確認することもできます。

自動的に外部サーバにイベントを転送しましたが，イベント・ブラウザに[外部情報]タブが表示されません。何が起きたのでしょうか?

[外部情報]タブは，コントロールを外部サーバに転送したイベントのみに対して表示されます。たとえば，[通知]，[通知して転送]，[同期]など，ほかのタイプの転送のいずれかを使用してイベントを転送した場合，[外部情報]タブで外部の状態を取得することができません。現時点では，イベント・ブラウザでほかのタイプの転送を表示するインタフェースはありません。

複数の外部システムにイベントを転送することは可能ですか?

はい。任意の数のシステムにイベントを転送できますが，コントロールは 1 つのシステムにしか転送できません。

外部システムにコントロールを転送しました。それを取り戻すことはできますか?

現時点では不可能です。

外部システムにコントロールを転送しました。システムはそれを戻すことができますか?

はい。外部システムは，イベント同期 Web サービスを使用して更新を生成する必要があります。Groovy スクリプトの `receiveChange()` メソッドが呼び出されたときに，`args` プロパティを次のように設定できます。

`args.returnControl`

接続サーバがコントロールを保有し，ログインしていること場合，コントロールがローカルのオペレーション管理インスタンスに返されます。

Groovy スクリプトおよびプログラミング

このセクションでは，Groovy スクリプトおよびプログラミングに関する FAQ を示します。

Groovy プログラミングに関する追加の情報はどこにありますか。

次の場所では，初心者用のチュートリアルにアクセスできます。

http://groovy.codehaus.org/Beginners+Tutorial

ログファイル・アダプタに「?」記号が使用されているのを見かけます。どういう意味ですか。

次のコードを見てください。

```
def username = args.credentials?.userName
```

`credentials` の値が null の場合，ランタイムにおけるその参照解除は実行されません。Null は，単に `username` に割り当てられます。`credentials` が null でない場合，`userName` が `username` に割り当てられます。

接続サーバによって送信されるペイロードが XML です。どのようにしたら Groovy スクリプトから解析できますか。

Groovy から XML を解析する簡単な方法は，XmlSlurper を使用する方法です。次のサンプル・コードおよび `LogfileAdapter.receiveChange()` メソッドを参照してください。

次のサンプルでは，OPR イベント(XML)を含むペイロードが想定されています。このペイロードは，更新を送信する接続サーバによって異なります。

```
  def receiveChange(def args) { def timestamp = new Date() def
externalEvent = args.externalEventChange def msg = """###
${timestamp.toString()}:receiveChange() called ### parameter
externalEvent:${externalEvent}\n\n""" m_logfile.append(msg) if
((externalEvent == null) || (externalEvent.length() == 0)) return
false; // check if this is an event or event_change def xmlNode = new
XmlSlurper().parseText(externalEvent); if (xmlNode.name().equals
("event")) return handleEvent(args, xmlNode) else if (xmlNode.name
().equals("event_change")) return handleEventChange(args, xmlNode)
else { def err = "Unexpected object type:${obj.getClass
().canonicalName}" m_logfile.append("${err}\n\n") m_logger.error(err);
return false } } def handleEvent(def args, def event) { m_logger.debug
("Change request received with ${EVENT_TAG} record.")// Update the
event properties if present in XML if (event."title".size())
args.title = event."title".text() if (event."description".size())
args.description = event."description".text() if
(event."solution".size()) args.solution = event."solution".text() if
(event."severity".size()) { def text = event."severity".text() def
severity = severityMap."${text}" if (severity) args.severity =
severity else { args.setHttpResponseStatus(400, "Invalid
severity:${text}") return false } } ...
```

外部処理の重大度をオペレーション管理イベントの重大度にマッピングしたいのですが，Groovy でそれを行う簡単な方法はありますか。

次のようなマップを使用してください。

```
// Map for severity mapping static def severityMap =
["0":OprSeverity.unknown, "1":OprSeverity.normal,
"2":OprSeverity.warning, "3":OprSeverity.minor, "4":OprSeverity.major,
"5":OprSeverity.critical] ... def externalSeverity = "2" def
oprSeverity = severityMap."${externalSeverity}"
```

統合スクリプト・メソッド

本項では，統合スクリプト・メソッドに関する FAQ を示します。

実装しなければならないスクリプト・メソッドは何ですか?

統合スクリプトでは次のメソッドを実装する必要があります。

- `init()`
- `destroy()`
- `forwardEvent()`

スクリプト・メソッドの詳細については，313ページ「Groovy スクリプト・メソッド」を参照してください。

任意のスクリプト・メソッドは何ですか?

次のスクリプト・メソッドは任意です。

- `forwardChange()`:スクリプト・アダプタが次の転送モードをサポートする場合にのみ必要です。[通知して更新]，[同期]または[同期してコントロールを移す]。
- `receiveChange()`:アダプタが次の転送モードをサポートする場合にのみ必要です。[同期]または[同期してコントロールを移す]。
- `getExternalEvent()`:スクリプト・アダプタがイベント・ブラウザの[外部情報]タブの入力をサポートしている場合にのみ必要です。
- `toExternalEvent()`:接続サーバがイベント同期 Web サービスで GET HTTP メソッド呼び出しを実行して，オペレーション管理を発生元とするイベントの現在のプロパティを取得する場合にのみ必要です。

クラスで EventProcessAdapter インタフェースを実装する必要はありますか?

いいえ。スクリプトでは必要なメソッドのみを実装する必要があります。このインタフェースは，ドキュメントの目的，および Eclipse または IntelliJ などの統合開発環境(IDE)を使用するユーザ向けに提供されています。

init() および destroy() はいつ呼び出されますか?

`init()` メソッドは，スクリプトが最初に読み込まれたときに呼び出されます。接続サーバごとに 1 つのインスタンスが読み込まれます。この読み込みは接続サーバへの最初のアクセス時に実行されます。接続サーバの設定が更新されたり，スクリプトが変更されるまで読み込まれた状態が続きます。その際に，`destroy()` メソッドが呼び出され，スクリプトが再び読み込まれます。読み込まれたスクリプトは，複数の呼び出し間でその状態を維持します。たとえば，リモート・サーバに対して接続が開かれたままになり，その後の呼び出しでもその接続が再利用される場合があります。

init() メソッドの引数で利用可能なプロパティは何ですか?

`init()` メソッドで利用可能なプロパティについては, 314ページ「init() メソッドのプロパティ」を参照してください。

例:

```
def init(def args) { m_logger = args.logger m_initArgs = args def
logfileDir = new File("${args.installDir}${File.separator}${LOG_DIR_
REL}") if (!logfileDir.exists()) logfileDir.mkdirs() m_logfile = new
File(logfileDir, LOGFILE_NAME) if (!m_logfile.exists()) m_
logfile.createNewFile() m_logger.debug("Logfile Adapter
initalized.INSTALL_DIR=${args.installDir}") def timestamp = new Date()
def msg = """### ${timestamp.toString()}:init() called ### parameter
connected server ID:${m_initArgs.connectedServerId} parameter
connected server name:${m_initArgs.connectedServerName} parameter
connected server display name:${m_initArgs.connectedServerDisplayName}
parameter node:${m_initArgs.node} parameter port:${m_initArgs.port}
parameter ssl:${m_initArgs.nodeSsl} parameter drilldown node:${m_
initArgs.drilldownNode} parameter drilldown port:${m_
initArgs.drilldownPort} parameter drilldown ssl:${m_
initArgs.drilldownNodeSsl}\n\n""" m_logfile.append(msg) }
```

詳細については, `com.hp.opr.api.ws.adapter.InitArgs` Javadoc を参照してください。

ping() メソッドの引数で利用可能なプロパティは何ですか?

`ping()` メソッドで利用可能なプロパティについては, 315ページ「ping() メソッドのプロパティ」を参照してください。

例:

```
def ping(def args) { args.outputDetail = "success" return true }
```

詳細については, `com.hp.opr.api.ws.adapter.PingArgs` Javadoc を参照してください。

forwardEvent() メソッドの引数で利用可能なプロパティは何ですか?

forwardEvent() メソッドで利用可能なプロパティについては, 315ページ「forwardEvent() メソッドのプロパティ」を参照してください。

例:

```
def forwardEvent(def args) { def timestamp = new Date() def extId =
"urn:uuid:${args.event.getId()}" def msg = """### ${timestamp.toString
()}:forwardEvent() called ### event.id:${args.event.id}
event.title:${args.event.title} event.state:${args.event.state}
event.external.id:${extId}\n\n""" m_logfile.append(msg) // Set the
external reference ID args.externalRefId = extId // Make a drilldown
to the original event as an example args.drilldownUrl = new URL
("http://${m_initArgs.drilldownNode}:${m_initArgs.drilldownPort}
${ROOT_DRILLDOWN_PATH}${args.event.getId()}") return true }
```

`forwardEvent()` メソッドに渡される引数の詳細については, `com.hp.opr.api.ws.adapter.ForwardEventArgs` Javadoc を参照してください。

forwardChange() メソッドの引数で利用可能なプロパティは何ですか?

`forwardChange()` メソッドで利用可能なプロパティについては, 318ページ「forwardChange() メソッドのプロパティ」を参照してください。

例:

```
def forwardChange(def args) { def timestamp = new Date() StringBuffer
buff = new StringBuffer() buff.append("### ${timestamp.toString
()}:forwardChange() called ###\n") buff.append("    parameter
externalRefId:${args.externalRefId}\n") buff.append("    change
headline:${args.changes.headline}\n")
args.changes.changedProperties.each { def propertyChange ->
buff.append("    changed property:${propertyChange.propertyName}
=${propertyChange.currentValue}\n") } buff.append("\n") m_
logfile.append(buff.toString()) return true }
```

`forwardChange()` メソッドに渡される引数の詳細については, `com.hp.opr.api.ws.adapter.ForwardChangeArgs` Javadoc を参照してください。

receiveChange() メソッドの引数で利用可能なプロパティは何ですか?

`receiveChange()` メソッドで利用可能なプロパティについては, 321ページ「receiveChange() メソッドのプロパティ」を参照してください。

例:

```
def receiveChange(def args) { def timestamp = new Date() def msg =
"""### ${timestamp.toString()}:receiveChange() called ### parameter
externalEvent:${args.getExternalEventChange()}\n\n""" m_logfile.append
(msg) def jc = javax.xml.bind.JAXBContext.JAXBContext.newInstance(
com.hp.opr.api.ws.model.event.OprEvent.class) def event =
jc.createUnmarshaller().unmarshal( new CharArrayReader
(args.externalEventChange.toCharArray())) if (event instanceof
com.hp.opr.api.ws.model.event.OprEvent) { if (event.titleUpdated)
args.title = event.title if (event.descriptionUpdated)
args.description = event.description if (event.solutionUpdated)
args.solution = event.solution return true } else { def err =
"Unexpected object type:${obj.getClass().canonicalName}" m_
logfile.append("${err}\n\n") m_logger.error(err); return false } }
```

`receiveChange()` メソッドに渡される引数の詳細については, `com.hp.opr.api.ws.adapter.ReceiveChangeArgs` Javadoc を参照してください。

receiveChange() を処理する際に特定の応答を Web サービスの呼び出し側に送り戻すには, どのようにしたらよいですか?

`args` には, 応答を制御できるメソッドがあります。

```
args.setHttpResponseStatus(400, "My response message")
```

HTTP 状態およびペイロードを任意に設定できます。値が 300 未満である場合, receiveChange() メソッドが呼び出された後にペイロードが処理されます。次に, 状態とメッセージが Web サービスの呼び出し側に返されます。値が 300 以上の場合, HTTP 状態とメッセージが Web サービスの呼び出し側に即座に返されます。

getExternalEvent() メソッドの引数で利用可能なプロパティは何ですか?

`getExternalEvent()` メソッドで利用可能なプロパティについては, 325ページ「getExternalEvent() メソッドのプロパティ」を参照してください。

例:

```
def getExternalEvent(def args) { def timestamp = new Date() def msg =
"""### ${timestamp.toString()}:getExternalEvent() called ###\n\n""" m_
logfile.append(msg) args.assignedUser = "logger" args.assignedGroup =
"logging group" args.state = "open" args.severity = "normal"
args.priority = "none" return true }
```

`getExternalEvent()` メソッドに渡される引数の詳細については, `com.hp.opr.api.ws.adapter.GetExternalEventArgs` Javadoc を参照してください。

イベント・プロパティ

本項にはイベント・プロパティに関連したよくある質問についての記載が含まれます。

どのイベント・プロパティが存在していますか?

イベント・ブラウザで利用可能なすべてのプロパティ, またはイベント Web サービスで見つかるプロパティが `OprEvent` オブジェクトで利用可能です。詳細については, `OprEvent` の Javadoc を参照してください。

例については, 次のイベント Web サービスに移動し, イベントを一覧表示します。

```
http://<bsmserver.example.com>/opr-console/rest/9.10/event_list
```

XML 出力によって利用可能なプロパティについての有益な情報が得られます。この XML 出力は, `OprEvent` オブジェクトから直接生成されます。

このイベントの関連 CI が何であるか理解したいのですが, その情報はどのようにしたら得られますか?

`event.relatedCi` は関連 CI を記述しているプロパティを持つオブジェクトを返します。これはタイプ `OprRelatedCi` です。`event.relatedCi.configurationItem` は CI の主要プロパティを含んでおり, その CI が別の CI の一部である場合は, 所属の CI を示す`event.relatedCi.configurationItem.partOf` を含むことになります。"`partOf`" はタイプ `OprRelatedCi` で, ほかの部分がない状態になるまで続行します。これで, 外部システムの CI を特定するための詳細情報が十分に提供されたことになります。

OprConfigurationItem オブジェクトはすべての CI の主要プロパティすべてを定義していません。ほかの主要プロパティはどのように取得できますか?

`OprConfigurationItem` ユーティリティ・メソッドを使用してほかのプロパティを取得する (`getProperty(name)`) か, `getProperties()` を呼び出してすべてのプロパティのマップを取得します。

イベント内の OprConfigurationItem には主要プロパティしかありません。ほかのプロパティはどのようにして取得できますか?

必要な CI のためにユーティリティ・メソッド `getCi(id)` を呼び出します。すべてのプロパティがこのメソッドによって返される CI 内に設定されます。このユーティリティ・メソッドは `forwardEvent()`, `forwardChange()`, および `receiveChanges()` の `args` で利用可能です。

OprConfigurationItem.getAny() によって返された JAXBElements で返すことが可能なクラス・タイプは何ですか?

可能なクラス・タイプは次のとおりです。

- String
- Boolean
- Integer
- Long
- Float
- Double
- Date

OprConfigurationItem.getAny() は同じ名前を持つ複数のオブジェクトを返します。これはどうして生じるのですか?

CI プロパティがリストの場合，複数のエントリを取得することになります。

トラブルシューティング

本項では，接続サーバのトラブルシューティングに関する FAQ を示します。

接続サーバを設定しましたが，イベント・ブラウザのショットカット・メニュー項目の[コントロールを次に移す]にその接続サーバが表示されません。どうしてですか。

次の項目を確認してください。

- 接続サーバが「アクティブ」であること。これは[一般]タブで確認できます。
- 接続サーバが「所有権の移転」をサポートしていること。これは[送信接続]タブで確認できます。

イベントを外部接続サーバに転送しました。スクリプトが呼び出されたかどうかをどのようにしたら確認できますか?

次の操作を実行してみてください。

- ログ記録をデバッグ・レベルに切り替えます。
- ログファイルを確認します。

ログ記録

本項では，ログ・ファイルに関する FAQ を示します。

スクリプト実行のログ・ファイルはどこで確認できますか?

次の場所でログ・ファイルを確認できます。

```
<HPBSM のルート・ディレクトリ>/log/opr-event-sync-adapter.log
```

ログ記録のレベルはどのようにしたら変更できますか?

`getExternalEvent()` メソッドの呼び出しの場合，次のプロパティ・ファイルを編集する必要があります。

<HPBSM のルート・ディレクトリ>/conf/core/Tools/log4j/EJB/opr-event-sync-adapter.properties

ほかのメソッド呼び出しについては，次のプロパティ・ファイルを編集します。

<HPBSM のルート・ディレクトリ>/conf/core/Tools/log4j/wde/opr-event-sync-adapter.properties

<HPBSM のルート・ディレクトリ>/conf/core/Tools/log4j/opr-ctxm-server/opr-event-syncadapter.properties

ファイルの最上部分にある loglevel パラメータを設定する必要があります。このファイルには，可能な値が含まれています。

どのようにしたらスクリプトからログ記録できますか?

init() メソッドに渡される args には，logger というプロパティがあります。このロガーをログ記録に使用します。たとえば，

```
def logger = args.logger logger.info("This is an info log")
logger.warn("This is a warning log") logger.error("This is a error
log") logger.debug("This is a debug log") logger.error("This is a
error log with an exception", exception)
```

WSDL によって定義された外部イベント処理サービスを統合する

本項で説明する手順を実行すると，標準 Web サービスおよび Web サービス記述言語 (WSDL) を使用してインタフェースを公開する外部イベント処理システムを統合することができます。

HP は，4 つの定義済みの転送タイプに対応する段階別に統合を実装することをお勧めします。各段階では，その前の段階に基づいて，より完全に機能する統合を作成します。

- **すべての転送タイプ**：すべての転送タイプでは，Groovy スクリプトの `init()` メソッドを実装する必要があります。このメソッドは，Groovy スクリプトが初期化されるときに必ず呼び出されます。
- **通知**：最小限の転送タイプです。ターゲット・サーバにイベントを転送するには，Groovy スクリプトの `forwardEvent()` メソッドを実装する必要があります。
- **通知して更新**：ターゲット・サーバにイベント変更を転送するには，Groovy スクリプトの `forwardChange()` メソッドを実装する必要があります。
- **同期**：ターゲット・サーバから変更を受信するには，Groovy スクリプトの `receiveChange()` メソッドを実装する必要があります。

  ターゲット・サーバで変更が発生した場合，ターゲット・サーバは OPR イベント同期 Web サービスを呼び出して，その変更をオペレーション管理にポストできなければなりません。Web サービス要求のペイロードは Groovy スクリプトの `receiveChange()` メソッドに渡され解釈されます。Web サービス呼び出しには，SOAP または REST ベースなど，任意のタイプを使用できます。
- **同期してコントロールを移す**：この実装では，追加の Groovy スクリプト・メソッドは必要ありません。
- *オプション*。**Ping サポート**：Groovy スクリプトの `ping()` メソッドを実装する必要があります。

Groovy スクリプトのメソッドに関する詳細については，313ページ「Groovy スクリプト・メソッド」を参照してください。

転送タイプが通知の統合を実装するために必要な設定手順の概要を次に示します。

- 355ページ「WSDL から Java コードを生成する」
- 357ページ「JAR ファイルのデプロイ」
- 357ページ「サービスにアクセスための Groovy スクリプトの変更 」
- 357ページ「外部イベント処理アプリケーションを接続サーバとして設定する」
- 359ページ「外部イベント作成のテスト」

その他の転送タイプをサポートするには，Groovy スクリプトの適切なメソッドを `forwardEvent()` メソッドの場合と同様の方法で実装する必要があります。

WSDL から Java コードを生成する

次の例では，Apache Axis を使用して WSDL ファイルから Java コードを生成します。

1. 次の前提条件をダウンロードし，インストールします。

   Java JDK 1.6

Apache Axis2

Apache Ant 1.7

2. たとえば，`integration` といった作業ディレクトリを作成します。
3. WSDL ファイルをそのディレクトリにコピーします(たとえば `integration/service.wsdl` のように) 。
4. 生成されたコード用のディレクトリを作成します(たとえば，`integration/gen` のように) 。
5. `gen` ディレクトリに変更します。
6. Apache Asix2 を実行し，WSDL ファイルからコードを生成します。

   ```
   wsdl2java -uri file:../service.wsdl
   ```

7. 生成された `build.xml` ファイルを編集して，JAR ファイルのマニフェストの `classpath` を正しく設定します。

   a. テキスト・エディタで `build.xml` ファイルを開きます。

   b. Ant ターゲットの `jar.client` を特定します。

   c. ファイルの最初にあるパス宣言の `axis2.class.path` の直後に次の Ant パスを追加します。

   ```
   <path id="axis2.client.class.path">
      <fileset dir="${axis2.home}">
         <include name="lib/*.jar"/>
      </fileset>
   </path> <pathconvert property="axis2.client.class.path.string"
                pathsep=" ">    <path
   refid="axis2.client.class.path" />    <flattenmapper />
   </pathconvert>
   ```

   d. jar タスクで，次のマニフェスト・ディレクティブを追加して，JAR ファイル・マニフェストで *classpath* を指定します。

   実行時には大量の Axis2 JAR ファイルが必要になるため，生成された JAR ファイルのマニフェストが実行時に依存関係を解決できるとよりシンプルになります。

   ```
   <jar destfile="${lib}/${name}-test-client.jar">
      <fileset dir="${classes}">
         <exclude name="**/META-INF/*.*"/>
         <exclude name="**/lib/*.*"/>
         <exclude name="**/*MessageReceiver.class"/>
         <exclude name="**/*Skeleton.class"/>
      </fileset>
      <manifest>
         <attribute name="Created-By"
                    value="Developer name goes here" />
         <attribute name="Class-Path"
                    value="${axis2.client.class.path.string}" />
      </manifest>
   </jar>
   ```

   e. 変更した `build.xml` ファイルを保存します。

f. コマンドラインで `ant` を実行します。サービスにアクセスするために必要な jar ファイルが作成されます。その出力は，`build/lib` ディレクトリで利用できます。

JAR ファイルのデプロイ

1. BSM ゲートウェイ・サーバのテスト・システムで新規統合ディレクトリを作成します。

   `<HPBSM のルート・ディレクトリ>/conf/opr/integration/test`

   `<HPBSM のルート・ディレクトリ>/conf/opr/integration/test/lib`

2. 生成された JAR ファイルを新しい `lib` ディレクトリにコピーします。
3. Axis2 JAR ファイルの `AXIS2_HOME/lib/*` を新しい `lib` ディレクトリにコピーします。

サービスにアクセスための Groovy スクリプトの変更

1. ログファイル・アダプタ Groovy スクリプトのコピーを作成します。

   `<HPBSM のルート・ディレクトリ>/conf/opr/integration/sample/LogfileAdapter.groovy`

   たとえば，

   `<HPBSM のルート・ディレクトリ>/conf/opr/integration/test/TestAdapter.groovy`

2. 新しい `TestAdapter.groovy` スクリプトを編集し，Axis2 で生成された JAR ファイルのクラスに対する呼び出しを追加します。
   - 最初に `forwardEvent()` スクリプト・メソッドを呼び出す必要があります。所定のオペレーション管理イベントのターゲット・オブジェクトを作成する必要があります。
   - Axis2 コード・ジェネレータによって，スタブ・クラスが生成されます。このクラスを構築し，`forwardEvent()` に渡された `args` のイベントを使用して，外部イベントを作成します。
   - それぞれのサービスは異なるため，ここでは特定の内容を提供しません。外部イベントの作成が成功すると，メソッドが終了する前に `externalID` が `args` に設定されます。

外部イベント処理アプリケーションを接続サーバとして設定する

オペレーション管理と外部イベント処理アプリケーションとの間のイベントおよびイベント変更の同期は，イベントを外部イベント処理アプリケーションに転送するオペレーション管理をホスティングしているサーバに依存します。そのため，接続サーバ・マネージャで外部イベント処理アプリケーションをターゲット接続サーバとして設定する必要があります。

接続サーバの設定方法に関する詳細については，オペレーション管理のオンライン・ヘルプの接続サーバのセクションを参照してください。

外部イベント処理アプリケーション・サーバをターゲット接続サーバとして設定するには，次の手順を実行します。

1. オペレーション管理のユーザ・インターフェイスの接続サーバ・マネージャに移動します。

   [**管理**]>[**オペレーション管理**]>[**セットアップ**]>[**接続サーバ**]

2. [新規]( ) ボタンをクリックして，[Create New Server Connection]ダイアログ・ボックスを開きます。

3. [表示名]フィールドにターゲットの外部イベント処理アプリケーション・サーバの名前を入力します。[名前]フィールドには，標準設定の名前が自動的に入力されます。たとえば，ターゲットの外部イベント処理アプリケーション・サーバの表示名に「`ExtEvtProcAppServer`」を入力すると，`ExtEvtProcAppServer` が[名前]フィールドに自動的に挿入されます。もちろん，この標準設定の名前を変更する場合は，[名前]フィールドに別の名前を入力できます。

   オプション :新規ターゲット・サーバの説明を入力します。

   [アクティブ]チェックボックスを選択していることを確認します。

   [**次へ**]をクリックします。

4. [外部イベント処理]を選択して，外部イベント処理アプリケーションに適したサーバの種類を選択します。

   [**次へ**]をクリックします。

5. ターゲット外部イベント処理アプリケーション・サーバの完全修飾 DNS 名を入力します。

   [**次へ**]をクリックします。

6. 統合のタイプを指定します。[統合タイプ]ダイアログでは，Groovy スクリプト・アダプタまたはイベント同期 Web サービスのいずれかの使用を選択できます。

   a. [**Groovy スクリプト・アダプタの呼び出し**]を選択します。

   b. 外部イベント処理のタイプ([**test**]など) を選択します。(このドロップダウン・リストのエントリは，統合で使用するファイルを含むディレクトリに対応しています) 。

   c. [Groovy Script File Name]フィールドで，[**TestAdapter.groovy**]を選択します。

   d. [Groovy Classpath]フィールドを次のように設定します。`<HPBSM のルート・ディレクトリ>/conf/opr/integration/test/lib/<jar ファイル>`

   e. [**次へ**]をクリックします。

7. [送信接続]ダイアログで，資格情報(ユーザ名，パスワード，ポート番号) を入力し，外部イベント処理アプリケーション・ターゲット・サーバに接続し，イベントを転送します。

   初期テストに対して[**同期および転送コントロールを有効化**]を選択します。[同期および転送コントロールを有効化]フラグを設定すると，オペレーション管理のオペレータは，イベントの所有権をターゲット接続サーバに移転することができるようになります。このフラグが設定されていないと，ルールの転送の設定時に[同期してコントロールを移す]オプションが転送タイプのリストに表示されません。

   いずれのターゲット接続サーバに対しても[同期および転送コントロールを有効化]フラグが設定されていない場合，[コントロールを次に移す]オプションがイベント・ブラウザのショートカット・メニューに表示されなくなることに注意してください。

   特定のサーバで[同期および転送コントロールを有効化]フラグが設定されていない場合は，そのサーバはイベント・ブラウザのショートカット・メニューで所有権の移転先のサーバとして利用できません。

   [**次へ**]をクリックします。

8. 残りのダイアログを完了して，[**終了**]をクリックします。

   ターゲットの外部イベント処理アプリケーション・サーバが接続サーバのリストに表示されます。

外部イベント作成のテスト

1. オペレーション管理を実行しているシステムでイベント・ブラウザを開きます。
2. イベントを 1 つ選択します。
3. イベントを右クリックして，[**転送コントロール**] > **<外部イベント処理アプリケーション・ターゲット・サーバ>** を選択します。
4. イベントが外部イベント処理アプリケーション・ターゲット・サーバに表示されることを確認します。

Service Manager の統合

本項では，HP Service Manager に接続する方法，HP Service Manager にイベントを転送する方法，転送済みイベントおよび後続のイベント変更が HP Service Manager からオペレーション管理に逆同期される仕組みについて説明します。

必要な設定手順の概要を次に示します。

- 360ページ「接続サーバとしての HP Service Manager サーバの設定」
- 362ページ「イベント転送ルールの設定」
- 365ページ「HP Service Manager サーバの設定」

接続サーバとしての HP Service Manager サーバの設定

オペレーション管理と HP Service Manager 間のイベントとイベント変更の同期化は，HP Service Manager にイベントを転送するオペレーション管理をホストしているサーバに依存します。このサーバには，これらのイベントとイベント変更が Service Manager から逆同期化されます。これを実現するための最初の手順は，接続サーバ・マネージャで HP Service Manager をターゲット接続サーバとして設定することです。

接続サーバの設定方法に関する詳細については，オペレーション管理のオンライン・ヘルプの接続サーバのセクションを参照してください。

HP Service Manager サーバをターゲット接続サーバとして設定するには，次の手順を実行します。

1. オペレーション管理のユーザ・インターフェイスの接続サーバ・マネージャに移動します。

   [**管理**] > [**オペレーション管理**] > [**セットアップ**] > [**接続サーバ**]

2. [新規] ( ) ボタンをクリックして，[サーバ接続の新規作成] ダイアログ・ボックスを開きます。
3. [**表示名**] フィールドに，ターゲット HP Service Manager サーバの名前を入力します。[名前] フィールドには，標準設定の名前が自動的に入力されます。たとえば，ターゲット HP Service Manager サーバの表示名として「`Service Manager 1`」と入力すると，[名前] フィールドに `Service_Manager_1` が自動的に表示されます。もちろん，この標準設定の名前を変更する場合は，[名前] フィールドに別の名前を入力できます。

   注: 新しいターゲット・サーバの名前 (この例では，`Service_Manager_1`) をメモします。この名前は後で HP Service Manager サーバを設定してオペレーション管理をホストしているサーバとの通信を行えるようにするときに，`username` として提供する必要があります。

   オプション: 新規ターゲット・サーバの説明を入力します。

   [**アクティブ**] チェックボックスが選択されていることを確認します。

   [**次へ**] をクリックします。

4. [外部イベント処理] を選択して，HP Service Manager などの外部インシデント・マネージャに適したサーバ・タイプを選択します。

   [**次へ**] をクリックします。

5. HP Service Manager ターゲット・サーバの完全修飾 DNS 名を入力します。

[**次へ**]をクリックします。

6. 次に，統合のタイプを確立する必要があります。[統合タイプ]ダイアログでは，Groovy スクリプト・アダプタまたはイベント同期 Web サービスのいずれかの使用を選択できます。

   a. HP Service Manager との統合用に Service Manager Groovy スクリプト・アダプタが提供されているため，[**Groovy スクリプト アダプタの呼び出し**]を選択します。

   b. [外部イベント処理タイプ]フィールドで，[**sm**]を選択します。

   c. [Groovy スクリプトのファイル名]フィールドで，[**ServiceManagerAdapter.groovy**]を選択します。

      (この場合，外部リソースが必要でないため，[Groovy クラスパス(Groovy Classpath)]フィールドは空白のまま残します。)

   d. [**次へ**]をクリックします。

7. HP Service Manager で，統合ユーザのユーザ名とパスワードを設定します。これは，HP Service Manager ターゲット・サーバにアクセスするために必要なユーザ名とパスワードです。

8. オペレーション管理ユーザ・インタフェースで，次の手順として，HP Service Manager ターゲット・サーバに接続し，そのサーバにイベントを転送するための資格情報(ユーザ名，パスワード，ポート番号)を入力します。[送信接続]ダイアログ・ボックスで，次の値を入力します。

   a. [**ユーザ名**]フィールドに，HP Service Manager で設定した統合ユーザのユーザ名を入力します。

   b. [**パスワード**]フィールドに，指定したユーザのパスワードを入力します。[**パスワード (確認入力)**]フィールドで，パスワードを繰り返し入力します。

   c. [**ポート**]フィールドで，オペレーション管理との統合用として HP Service Manager 側で設定されたポートを指定します。入力するポート番号を検索するには，次の手順を実行します。

      - 次のファイルに移動します。

        `<HP Service Manager のルート・ディレクトリ>/HP/Service Manager <バージョン>/Server/RUN/sm.ini`

      - `sm.ini` ファイルで，セキュア HTTP 接続を使用するかどうかに応じて，2 つのポート・エントリを探します。この 2 つのポート・エントリとは，標準設定のポート番号が `13080` の `httpPort` と標準設定のポート番号が `13443` の `httpsPort` です。設定に応じて，ポートの実際の値はこれらの標準設定値と異なる可能性があります。[ポート]フィールドに適切な値を入力します。

   d. セキュア HTTP を使用しない場合は，[セキュア HTTP を使用]チェックボックスが選択されていないことを確認してください。

   e. [**Supports Synchronize and Transfer Control**]チェックボックスが選択されていることを確認します。[Supports Synchronize and Transfer Control]フラグがオンに設定されている場合は，オペレーション管理のオペレータがイベントの所有権をターゲット接続サーバに移転できます。このフラグが設定されていないと，ルールの転送の設定時に[同期してコントロールを移す]オプションが転送タイプのリストに表示されません。

      また，いずれのターゲット接続サーバに対しても[Supports Synchronize and Transfer Control]フラグが設定されていない場合は，[コントロールを次に移す]オプションがイベント・ブラウザのショットカット・メニューに表示されません。

特定のサーバで[Supports Synchronize and Transfer Control]フラグが設定されていない場合は，そのサーバがイベント・ブラウザのショットカット・メニューで所有権の移転先のサーバとして利用できません。

f. 接続をテストします。

g. [**次へ**]をクリックします。

9. 単にイベントを HP Service Manager に転送するだけでなく，HP Service Manager にドリルダウンできるようにする場合は，イベント・ドリルダウンを実行する HP Service Manager システムの完全修飾 DNS 名とポートを指定する必要があります。

> **注:** HP Service Manager に対するイベント・ドリルダウンを有効にするには，HP Service Manager サーバのインストール/設定手順に従って，HP Service Manager サーバ用の Web 層クライアントをインストールする必要があります。
>
> 接続サーバ・マネージャの[イベント ドリルダウン]ダイアログで，使用する設定済みのポートと一緒に Web 層クライアントをインストールしたサーバを設定します。
>
> 接続サーバ・マネージャの[イベント ドリルダウン]ダイアログでサーバを指定しなかった場合は，HP Service Manager にイベントとイベント変更を転送し，HP Service Manager からイベント変更を受け取るために使用されるサーバ上に Web 層クライアントがインストールされていると見なされます。
>
> [イベント ドリルダウン]ダイアログで何も設定されず，Web 層クライアントが HP Service Manager サーバ・マシン上にインストールされていない場合は，Web ブラウザで要求された URL を見つけることができません。

[**次へ**]をクリックします。

10. 次の手順は，HP Service Manager からオペレーション管理にイベント変更を逆同期化できるようにすることです。そのためには，HP Service Manager サーバからオペレーション管理をホストしているサーバにアクセスするための資格情報を入力する必要があります。

a. [受信接続]ダイアログで，[**Support Event BackSync**]チェックボックスを選択してから，HP Service Manager サーバからオペレーション管理をホストしているサーバに接続するために必要なパスワード(この例では，`Myqwer1_`)を入力します。

> **注:** このパスワード(この例では，`Myqwer1_`)をメモします。後で，オペレーション管理をホストしているサーバと通信するように HP Service Manager サーバを設定するときに入力する必要があります。このパスワードは，手順 3 で設定したサーバ名(`Service_Manager_1`)と一緒に使用されます。

b. [**終了**]をクリックします。ターゲット HP Service Manager サーバが，接続サーバの一覧に表示されます。

イベント転送ルールの設定

次の手順では，どのイベントが自動的に HP Service Manager に転送されるかを決めるイベント転送ルールを設定します。

フィルタの設定に関する完全な詳細情報については，オペレーション管理のオンラインヘルプを参照してください。

転送ルールを設定するには，次の手順を実行します。

1. オペレーション管理のユーザ・インタフェースの転送ルール・マネージャに移動します。

   [**管理**]>[**オペレーション管理**]>[**イベントの自動化**]>[**イベント転送**]

2. [新規]( )ボタンをクリックして，[転送ルールの新規作成]ダイアログを開きます。
3. [**表示名**]フィールドに，転送ルールの名前を入力します。この例では，`Forward Critical (Sync and Transfer Control)` になっています。

   任意。作成している転送ルールの説明を入力します。

   [**アクティブ**]チェックボックスが選択されていることを確認します。HP Service Manager でそのステータスを確認するためには，ルールがアクティブになっている必要があります。
4. イベント・フィルタ・フィールドの隣の参照ボタンをクリックします。[イベント フィルタを選択]ダイアログが開きます。

   [イベント フィルタを選択]ダイアログ・ボックスで，次の手順のいずれかを実行します。

   - 既存のフィルタを選択します。
   - 次のように新しいフィルタを作成します。

     i. [新規]( )ボタンをクリックして，[フィルタ構成]ダイアログを開きます。

     ii. [**フィルタ表示名**]フィールドに，新しいフィルタの名前を入力します。この例では，**FilterCritical** になっています。

     危険域以外のすべての重大度レベルに関するチェックボックスの選択を解除します。

     [**OK**]をクリックします。

     iii. 新規フィルタが[イベント フィルタを選択]ダイアログに表示されていることが確認できます(ハイライト表示されていない場合は選択します)。

     [**OK**]をクリックします。
5. [**ターゲット サーバ**]の下で，360ページ「接続サーバとしての HP Service Manager サーバの設定」の項で設定したターゲット接続サーバを選択します。この例では，`Service Manager 1` です。

   ターゲット・サーバ選択フィールドの横にある[追加]( )ボタンをクリックします。これにより，接続サーバの詳細を確認できるようになります。[**転送タイプ**]フィールドで，転送タイプを選択します。

   転送タイプの詳細については，304ページ「イベントおよびイベントの変更を外部イベント・プロセスに転送」というタイトルの項を参照してください。

   [**OK**]をクリックします。

HP Service Manager からイベント・ブラウザの URL 起動の設定

オペレータがイベント・ブラウザの URL 起動を使用して，HP Service Manager からオペレーション管理ユーザ・インタフェースへのイベント・ドリルダウンを実行できるためには，Operations Manager i の適切な権限を持つ BSM 内の有効なユーザとしてオペレータを設定する必要があります。

- **ユーザ・アカウント要件**

  シングル・サインオン(SSO)認証が設定されている場合は，HP Service Manager にログオンして URL 呼び出しを実行するために HP Service Manager のオペレータが使用するユーザ名と同じユーザ名で BSM 内の各ユーザを設定します(各 BSM ユーザのパスワードは，空白または任意の文字列に設定できます)。HP Service Manager へのログインが成功したら，BSM ユーザは追加の認証なしでオペレーション管理イベント・ブラウザを起動できます。

  HP Service Manager が SSO 認証を使用するように設定されていない場合は，HP Service Manager のオペレータが使用するユーザ名と同じユーザ名で各ユーザを設定し，有効なパスワードを指定します。ユーザは，オペレーション管理イベント・ブラウザを起動するときにユーザ名とパスワードの入力が要求されます。

- **必要なユーザ権限**

  必要なアクションを含む「ユーザに割り当てられたイベント」権限を各 BSM ユーザに付与する必要があります。オプションで，ユーザに割り当てられていないイベントを表示するための権限を付与できます。

注: 有効なユーザ名を使用しなかった場合または必要な表示権限を持っていない場合は，HP Service Manager からオペレーション管理イベント・ブラウザの URL 起動を実行しようとすると，空白のブラウザ・ウィンドウが表示されます。

308ページ「外部アプリケーションからイベント・ブラウザの URL 起動を実行する」も参照してください。

イベント・ブラウザからの HP Service Manager の URL 起動の設定

Web 層クライアントを使用して，オペレーション管理イベント・ブラウザから HP Service Manager の URL 起動を実行できるようにするには，次の手順を実行します。

1. 次の場所にある Groovy スクリプト `ServiceManagerAdapter.groovy` に移動します。

   `<HPBSM のルート・ディレクトリ>/conf/opr/integration/sm/ServiceManagerAdapter.groovy`

2. `ServiceManagerAdapter.groovy` Groovy スクリプトを開きます。

3. Groovy スクリプト内で次のテキストを探します。

   ```
   private static final String DRILLDOWN_ROOT_PATH = '/webtier-9.20/index.do?ctx=docEngine&file=probsummary&query=number%3D';
   ```

4. `webtier-9.20` の値を HP Service Manager Web 層クライアントにアクセスするために必要な値に変更します。

   完全なドリルダウン URL が次のようになります。

   `http://<HP Service Manager Web 層サーバの FQDNS>/<HP Service Manager への Web パス>/<URL クエリ・パラメータ>`

   ここで，`<HP Service Manager Web 層サーバの FQDNS>` は，Web 層クライアントがインストールされている HP Service Manager サーバの完全修飾 DNS 名です。この URL の部分は，接続サーバ・マネージャで HP Service Manager をターゲット接続サーバとして設定したときに

(360ページ「接続サーバとしての HP Service Manager サーバの設定」を参照) 入力した値に従って(`http://` と一緒に) 自動的に追加されます。

ここで, ドリルダウン URL の例を示します。

```
http://smserver.example.com/SM920/index.do?ctx=docEngine&file=probsummary&query=number%3D
```

この例では, `webtier-9.20` を `SM920` に置き換える必要があります。その他の URL の部分は自動的に設定されます。

5. HP Service Manager Web 層設定ファイル `web.xml` で, `querySecurity` パラメータの値を標準設定値(`true`) から **`false`** に変更します。

   詳細については, HP Service Manager オンライン・ヘルプの「Web パラメータ : querySecurity」の項を参照してください。

HP Service Manager サーバの設定

次の手順では, オペレーション管理と統合するように HP Service Manager サーバを設定します。

HP Service Manager サーバを設定するには, HP Service Manager で次の手順を実行します。

1. HP Service Manager ユーザ・インタフェースの左側のペインで, 次の場所に移動します。

   [**Tailoring**] > [**Integration Manager**]

2. [**追加**]をクリックして, 新しい設定を追加します。
3. [Integration Template]フィールドで, [**SMOMi**]統合テンプレートを選択します。[**次へ**]をクリックします。
4. 任意。ログ・レベルを必要な値に変更します。

   任意。説明を「`This is for SMOMi integration`」などに変更します。

   [**次へ**]をクリックします。

5. [General Parameters]タブで, 既存のエントリを次の値に置き換えます。

名前	値	カテゴリ
`omi.server.url`	`http://<BSM ゲートウェイの FQDN>/opr-gateway/rest/9.10/synchronization/event/`	一般
`username`	`Service_Manager_1` (これは, 前に360ページ「接続サーバとしての HP Service Manager サーバの設定」の項で設定した HP Service Manager ターゲット・サーバの名前です)。	ヘッダ

名前	値	カテゴリ
omi.eventdetail.baseurl	http://<BSM ゲートウェイの FQDN>/opr-console/opr-evt-details.jsp?eventId=	一般

6. [Secure Parameters]タブで，360ページ「接続サーバとしてのHP Service Manager サーバの設定」の項でターゲット接続サーバを設定するときに[受信接続]ダイアログで指定したパスワードを設定します。この例では，HPqwer1_ です。

   [**次へ**]をクリックします。

7. [Integration Instance Fields]ダイアログで，[**次へ**]をクリックします。

8. [Integration Instance Mapping]ダイアログで，[**終了**]をクリックします。

> **注:** ルールがアクティブになっていることを確認してください。ルールをアクティブにするには，ルールを選択して，[**有効にする**]をクリックします。

マッピングおよびカスタマイズ

独自のカスタム属性を Groovy スクリプトに追加してから，それらのカスタム属性を HP Service Manager 内の該当するフィールドにマップできます。オペレーション管理と HP Service Manager 間の属性のマッピング方法を変更することもできます。マッピングは HP Service Manager 内の BDM マッピング・マネージャで行います。

[**System Administration**]＞[**Ongoing Maintenance**]＞[**BDM Mapping Management**]

属性のマッピングに関する詳細については，HP Service Manager オンライン・ヘルプを参照してください。

接続のテスト

接続をテストするには，定義したフィルタ(このフィルタの例では，重大度が危険域になっている)に一致するオペレーション管理をホストしているサーバにイベントを送信してから，そのイベントが期待どおりに HP Service Manager に転送されることを確認します。

接続をテストするには，次の手順を実行します。

1. オペレーション管理を実行しているシステム上で，イベント・ブラウザを開きます。
2. オペレーション管理を実行しているシステム上で，コマンド・プロンプトを開いて，次のディレクトリに変更します。

   <HPBSM のルート・ディレクトリ>\opr\support

3. 次のコマンドを使用してイベントを送信します。

   ```
   sendevent -s critical -t test111-1
   ```

4. イベントがオペレーション管理のイベント・ブラウザに表示されることを確認します。

5. [**外部情報**]タブを選択します。
6. [外部 ID]フィールドに，有効な HP Service Manager インシデント ID が表示されます。
7. 次に，インシデントが HP Service Manager の[Incident Details]に表示されることを確認します。

   イベント・ドリルダウン接続が正しく設定されている場合は，[**編集**]ボタンをクリックします。ブラウザ・ウィンドウが開いて，HP Service Manager の[Incident Details]内のインシデントに直接移動できます。

   イベント・ドリルダウン接続が設定されていない場合は，次の手順を実行します。

   a. オペレーション管理イベント・ブラウザの[外部情報]タブで，[外部 ID]フィールドからインシデント ID をコピーまたはメモします。

   b. HP Service Manager ユーザ・インタフェースで，次に移動します。

      [**Incident Management**] > [**Search Incidents**]

   c. [Incident Id]フィールドにインシデント ID を貼り付けるか，入力します。

   d. [**検索**]ボタンをクリックします。これにより，[Incident Details]内のインシデントに移動します。
8. HP Service Manager でインシデントをクローズします。
9. インシデントの状態の変化 (今はクローズ済み) がオペレーション管理に反映されることを確認します。HP Service Manager でクローズされたイベントは，アクティブなイベント・ブラウザには表示されず，履歴ブラウザに表示されます。

属性の同期

標準設定では，HP Service Manager とオペレーション管理間ですべての属性が同期化されるわけではありません。オペレーション管理から HP Service Manager への一方向に一度だけ更新される属性もあれば，双方向で同期化される属性もあります。

一方向の同期化 : オペレーション管理から HP Service Manager へ

次の属性は，一度だけ，つまり，イベントが初めて作成され，イベントのコントロールの移転が接続サーバ・マネージャで設定されている場合にだけ，オペレーション管理から HP Service Manager に転送されます。

- タイトル
- 重大度
- 優先度
- オペレータ : イベントを転送した，イベントに割り当てられたオペレータ
- カテゴリ
- サブカテゴリ
- 関連 CI

これらの属性は，HP Service Manager からオペレーション管理への逆同期化は行われません。

双方向の同期化:

オペレーション管理とHP Service Manager 間の双方向同期化に対応している属性は次のとおりです。

- 詳細
- ライフサイクル状態(状態は終了に変化したときにだけ更新される)
- ソリューション
- オペレーション管理イベントの注釈は HP Service Manager のアクティビティ・ログに同期化される
- [イベント詳細]の[外部情報]タブの内容

Groovy スクリプトを使用した属性の同期化

どの属性が更新されるかに関する標準動作を変更する場合は, それを Groovy スクリプト内で指定できます。Groovy スクリプトで, HP Service Manager 内で更新されるフィールドとオペレーション管理内で更新されるフィールドを指定します。Groovy スクリプトでカスタム属性を指定することもできます。

注: 複数サーバ環境では, Groovy スクリプト・ファイルがすべてのゲートウェイ・サーバ上で更新されていることを確認してください。

Groovy スクリプトのカスタマイズに関するヒント

本項では, Groovy スクリプトのカスタマイズに関するヒントを提示し, カスタマイズ可能な内容についていくつかの例を紹介します。その他の変更可能な項目については, Groovy スクリプトの設定セクションを参照してください。

Groovy スクリプトの設定セクションでは, オペレーション管理とHP Service Manager 間で同期される属性を定義または変更できます。また, Groovy スクリプトの設定セクションには, ライフサイクル状態, 重大度, 優先度に関する標準設定値のマッピングが含まれています。さらに, これらのマッピングを変更したり, 受信要求と送信要求用のマッピングを別々に定義したりできます。

必要に応じて, Groovy スクリプトの別の部分でさらに詳細な設定を実行できます。

Groovy スクリプトの設定セクションの始まりと終わりは次のようにマークされます。

```
//

// configuration section to customize the Groovy script

// BEGIN

...

...

//

// configuration section to customize the Groovy script

// END
```

Groovy スクリプトを変更する前に, オリジナル(標準設定)のスクリプトのコピーを作成してください。これは, 新しいバージョンの Groovy スクリプトがパッチ, サービス・パック, またはホットフィックスとともに配布された場合に, オリジナルのスクリプトが上書きされる可能性があるためです。カスタマイズした

Groovy スクリプトが安全な場所にコピーされていることを確認してください。変更分と，パッチ，サービス・パック，またはホットフィックスで配布する新しい Groovy スクリプトを結合しなければならない場合があります。

オペレーション管理から HP Service Manager へのマッピングは，BDM 1.1 インシデント Web サービス仕様に準拠しています。BDM 1.1 インシデント Web サービスから HP Service Manager へのマッピングは，HP Service Manager 内の BDM マッピング・マネージャで指定されます。BDM マッピング・マネージャの詳細については，HP Service Manager オンライン・ヘルプの BDM マッピング・マネージャの項を参照してください。

属性同期化の制御

いくつかのブール値変数を true または false に設定することにより，特定の属性に対する更新をオペレーション管理と HP Service Manager 間でどのように同期化するかを制御できます。

2 つの例を示します。

- `private static final SyncTitleToSMOnUpdate = false;`

  この Groovy スクリプト行は，オペレーション管理で加えられたタイトルに対する変更の HP Service Manager への同期化を無効にします。

- `private static final Boolean SyncTitleToOPROnUpdate = false;`

  この Groovy スクリプト行は，HP Service Manager で加えられたタイトルに対する変更のオペレーション管理への同期化を無効にします。

タイトルは HP Service Manager 内の必須属性であり，上記フラグに関係なく，インシデントの作成中にオペレーション管理に入力されたタイトルを使用して設定されます。

OPR ライフサイクル状態の BDM ライフサイクル状態へのマッピング

Groovy スクリプトを変更することにより，オペレーション管理 (OPR) ライフサイクル状態を HP Service Manager 内の(BDM) ライフサイクル状態にマップできます。

2 つの例を示します。

- `private static final Map OPR2BDMLifecycleState = ["open":null, "in_progress":null, "resolved":null, "closed":"closed"];`

  この例では，OPR ライフサイクル状態の "closed" だけが BDM ライフサイクル状態の "closed" にマップされます。`null` は，HP Service Manager 内の状態を変更しない命令です。

- `private static final Map OPR2BDMLifecycleState = ["open":null, "in_progress":null, "resolved":"resolved", "closed":"closed"];`

  この例では，OPR 状態の `resolved` が BDM 状態を `resolved` に設定します。

  インシデントの初期ライフサイクルはインシデントの作成中に設定されるため，OPR ライフサイクル状態の `open` を SM ライフサイクル状態の `open` にマップする必要はありません。

BDM ライフサイクル状態の OPR ライフサイクル状態へのマッピング

次の設定行を使用すれば，BDM ライフサイクル状態から既知の OPR 状態へのマッピングを指定できます。

```
private static final Map BDM2OPRLifecycleState = ["open":null, "work-in-progress":null, "resolved":null, "closed":"closed"];
```

この例では，(BDM) インシデントがクローズされたときに，(OPR) イベントがクローズされます。null は，HP Service Manager 内のインシデント状態が変化してもイベント状態を変更しない命令です。

BDM ライフサイクル状態の open を OPR ライフサイクル状態の open にマップした場合は，次のようなことが起きます。HP Service Manager でインシデントをクローズしてから，再開すると，オペレーション管理で対応するイベントが再開されます。

構文エラー

Groovy スクリプトのカスタマイズ中に構文エラーが発生した場合は，ログ・ファイル opr-event-sync-adapter.log でエラーの解決方法に関する情報を確認してください。ログ・ファイルは次の場所にあります。

<HPBSM のルート・ディレクトリ>/log/opr-event-sync-adapter.log

Service Manager 9.2 Integration のカスタマイズ

ServiceManagerAdapter の groovy スクリプトは，Service Manager へのイベント転送用として提供されます。このスクリプトは，ご使用のインストール内容向けにカスタマイズすることが可能です。

ServiceManagerAdapter の groovy スクリプトをカスタマイズするには，Scripts Manager( ) を開き，**sm:ServiceManagerAdapter** スクリプトを選択して編集用として開きます( )。[スクリプトの編集 (Edit Script)] ウィンドウが開きます。スクリプトの内容は [**スクリプト**] タブ内に表示されます。

> **ヒント:** スクリプトのテキストを任意のテキスト・エディタにコピーします。編集が完了したら，編集済みのテキストを [スクリプトの編集 (Edit Script)] ウィンドウに戻してスクリプトを保存します。

スクリプトの開始部分の付近に，Service Manager との BSM イベント同期の標準設定動作を変更するために使用される 2 つのセクションがあります。

アクセス方法

[**管理**] > [**オペレーション管理 >**] > [**セットアップ>**] > [**接続サーバ>**] ボタンを選択します。

ServiceManagerAdapter のスクリプトの設定

このセクションは，どのイベントおよびインシデントのプロパティが Service Manager との同期を確立するか，および次のコメント内に含まれるかを制御します。

- BEGIN Configuration: Customization of properties for synchronization
- END Configuration:Customization of properties for synchronization

ServiceManagerAdapter スクリプトは，定数を含むセクションを設定可能で，6 "マップ" および "8" セットによりプロパティの同期の設定が可能になります。それぞれについて，次以降で説明します。

Service Manager ドリルダウン定数

調整可能な 1 番目の変数は SM_WEB_TIER_NAME です。この値を Service Manager システムの Tomcat コンテナにデプロイ済みの Web アプリケーションのベース名に設定します。この Web アプリケーションは Service Manager へのドリルダウン用として使用されます。名前はドリルダウン用の URL パスとして使用されます。Web アプリケーションのベース名 (".war" は削除されます) と一致する必要があります。標準設定を以下に示します。

```
private static final String SM_WEB_TIER_NAME = 'webtier-9.30'
```

BSM 管理者ユーザ

BSM_ADMINISTRATOR_LOGIN_NAME 変数が BSM 管理者ユーザの名前を含めるために使用されます。標準設定では，admin に設定されています。

転送ルールによって自動的に転送されるオペレーション管理イベントの場合，_is_recorded_by 属性は BSM_ADMINISTRATOR_LOGIN_NAME 変数で指定されたユーザに設定されます。

手動で転送されるオペレーション管理イベントの場合，_recorded_by 属性は転送要求を開始するユーザに設定されます。

```
private static final String BSM_ADMINISTRATOR_LOGIN_NAME = 'admin'
```

列挙値マップ

マップは，オペレーション管理イベントのプロパティの列挙された値を Service Manager のインシデント・プロパティ上の値にマップするよう定義されます。これらのマップは一般的にカスタマイズするべきものではありませんが，次に記載の設定で指定可能な有効な値のリストが提供されます。各マップの詳細については，スクリプト内で定義された実際の値を表示してください。

- **MapOPR2SMStatus**:オペレーション管理イベント state を Service Manager のインシデント status にマップします
- **MapSM2OPRState**:Service Manager のインシデント status をオペレーション管理イベント state にマップします
- **MapOPR2SMUrgency**:オペレーション管理イベント severity を Service Manager のインシデント urgency にマップします
- **MapSM2OPRSeverity**:Service Manager のインシデント urgency をオペレーション管理イベント severity にマップします
- **MapOPR2SMPriority**:オペレーション管理イベント priority を Service Manager のインシデント priority にマップします
- **MapSM2OPRPriority**:Service Manager のインシデント priority をオペレーション管理イベント priority にマップします

ユーザ定義プロパティ・マップ

次のマップにより，ユーザは任意の最上位レベルのオペレーション管理イベント・プロパティを任意の任意の最上位レベルのService Manager のインシデント・プロパティにマップすることが可能になります。

- **MapOPR2SMCustomAttribute**:同期のために，指定したオペレーション管理のカスタム属性を Service Manager のインシデント・プロパティにマップします。

  Service Manager のインシデント・プロパティの名前 (XML タグ名) とともに CA 名をマップに追加します。

  "activity_log" のターゲットの Service Manager インシデント・プロパティ名は，CA の変更を Service Manager のインシデント・アクティビティ・ログに追加します。

  > **注:** 最上位レベルの Service Manager インシデント・プロパティのみがこのマップでサポートされます。

- **MapSM2OPRCustomAttribute**:同期のために，指定した Service Manager のインシデント・プロ

パティを，オペレーション管理イベントのカスタム属性にマップします。

Service Manager のインシデント・プロパティの名前を，オペレーション管理イベントのカスタム属性名とともに，マップに追加します。

注 : 最上位レベルの Service Manager インシデント・プロパティのみがこのマップでサポートされます。

例 :

次によってオペレーション管理イベントのカスタム属性 `MyCustomCA` が Service Manager のインシデント `activity_log` に，カスタム属性 `MyCustomCA1` が Service Manager のインシデント・プロパティ `SMCustomAttribute` にそれぞれ同期します。

```
private static final Map<String, String> MapOPR2SMCustomAttribute =
["MyCustomCA" :"activity_log", "MyCustomCA1" :"SMCustomAttribute"]
```

次によって Service Manager のインシデント・プロパティ `incident_status` がオペレーション管理のカスタム属性 `SMIncidentStatus` に同期します。

```
private static final Map<String, String> MapSM2OPRCustomAttribute =
["incident_status" :"SMIncidentStatus"]
```

同期変更セット

次のセットは，オペレーション管理イベントおよび Service Manager インシデントに変更が発生するごとに，どのプロパティおよび列挙された値が同期されるかを定義します。標準で変更発生時に同期されるプロパティは**太字**で表記されます。各リストで，"*" の値を指定可能です。この場合，すべての可能なプロパティまたは列挙された値は指定したリストで同期されます。

注 : Service Manager のインシデントが作成されると，すべての可能なオペレーション管理イベント・プロパティおよび列挙された値が Service Manager インシデント内で設定されます。変更内容の同期で主に使用されるセットは次のとおりです。

SyncOPRPropertiesToSM

変更発生時に対応する Service Manager のインシデント・プロパティに同期するオペレーション管理イベント・プロパティ

- title
- **description**
- **state**
- severity
- priority
- **solution**
- assigned_user
- assigned_group

SyncOPRPropertiesToSMActivityLog

変更発生時に対応する Service Manager のインシデント・アクティビティ・ログに同期するオペレーション管理イベント・プロパティ

- **title**
- description
- **state**
- **severity**
- **priority**
- solution
- **annotation**
- **duplicate_count**
- **custom_attribute**
- **cause**
- **symptom**
- control_transferred_to
- **assigned_user**
- **assigned_group**

SyncSMPropertiesToOPR

変更発生時に対応するオペレーション管理イベント・プロパティに同期する Service Manager のインシデント・プロパティ

- name
- **description**
- **incident_status**
- urgency
- priority
- **solution**

SyncOPRStatesToSM

変更発生時に対応する Service Manager のインシデント・ステータスに同期するオペレーション管理イベントの状態

> **注 :** `state` は `SyncOPRPropertiesToSM` に含まれている必要があります。さもないとこのリストは無視されます。

- open
- in_progress
- in_progress

- resolved
- **closed**

SyncOPRSeveritiesToSM

変更発生時に対応する Service Manager のインシデントの緊急度に同期するオペレーション管理イベントの重大度

> **注 :** `severity` は `SyncOPRPropertiesToSM` に含まれている必要があります。さもないとこのリストは無視されます。

- **critical**
- **major**
- **minor**
- **warning**
- **normal**
- **unknown**

SyncSMStatusToOPR

変更発生時にオペレーション管理イベント状態に同期する Service Manager のインシデント状態

> **注 :** `status` は `SyncSMPropertiesToOPR` に含まれている必要があります。さもないとこのリストは無視されます。

- accepted
- assigned
- open
- reopened
- pending-change
- pending-customer
- pending-other
- pending-vendor
- referred
- suspended
- work-in-progress
- rejected
- replaced-problem
- resolved

- cancelled
- **closed**

SyncSMUrgenciesToOPR

変更発生時にオペレーション管理イベントの重大度に同期する Service Manager のインシデントの緊急度

> **注:** `urgency` は `SyncSMPropertiesToOPR` に含まれている必要があります。さもないとこのリストは無視されます。

許容される値は **1-4** です。

SyncSMPrioritiesToOPR

変更発生時にオペレーション管理イベントの優先度に同期する Service Manager のインシデント・プロパティ

> **注:** `priority` は `SyncSMPropertiesToOPR` に含まれている必要があります。さもないとこのリストは無視されます。

許容される値は **1-4** です。

例:

次の例では，オペレーション管理のイベントで対応するプロパティが変更されるたびにオペレーション管理のタイトル，状態，説明が Service Manager のインシデントに同期しています。

```
private static final Set SyncOPRPropertiesToSM = ["title", "state",
"description"]
```

次の例では，オペレーション管理のイベントで対応するプロパティが変更されるたびにオペレーション管理の解決または終了した状態が Service Manager のインシデントに同期することになります。

```
private static final Set SyncOPRStatesToSM = ["resolved", "closed"]
```

> **注:** Service Manager のアクティビティ・ログに同期しているプロパティは各変更で一緒に連結され，Service Manager のインシデント・アクティビティ・ログに追加されます。

ローカリゼーション

このセクションは，次で表示されるテキストのいくつかをローカライズ可能にするために提供されます。

- Operations Manager のイベント・ブラウザの[転送]タブ(イベント・チャネルのデプロイメントでは使用不可)
- HP Service Manager のインシデント・アクティビティ・ログ

このセクションは，次のコメント内に含まれます。

- `BEGIN Localization:Customization of text values for language localization`
- `END Localization:Customization of text values for language localization`

次のセクションはローカライズ可能なテキストを記載しています。

[転送]タブ

Service Manager インシデント・プロパティの緊急度と優先度は整数型です。[転送]タブで表示される値により意味があるようにするために，文字列を表示できるようマップが提供されます。これらの文字列はブラウザでの表示用にローカライズ可能です。

- **Service Manager の緊急度の値**

  テキストの値が[転送]タブに表示されます。

  注: このテキストは任意のロケールでローカライズ可能です。

```
privatestaticfinal Map SMUrgency = ["1":"1 - Critical", "2":"2 -
High", "3":"3 - Average", "4":"4 - Low"]
```

- **Service Manager の優先度の値**

  テキストの値が[転送]タブに表示されます。

  注: このテキストは任意のロケールでローカライズ可能です。

```
privatestaticfinal Map SMPriority = ["1":"1 - Critical", "2":"2 -
High", "3":"3 - Average", "4":"4 - Low"]
```

Service Manager のインシデント・アクティビティ・ログ

BSM から Service Manager への同期によって，さまざまなテキストが Service Manager インシデント・アクティビティ・ログに追加されます。.このテキストは次のようにローカライズされます。

- 一般的なロケール設定 :主に日付のフォーマットで使われます。Locale.JAPAN などに変更することが可能です。すべての有効な値については Java Locale クラスのドキュメントを参照してください。

```
private static final Locale LOCALE = Locale.getDefault()
```

- アノテーションの日付フォーマット :構文の詳細については Java SimpleDateFormat クラスのドキュメントを参照してください。スクリプトの標準設定は次のとおりです。

```
private static final String ANNOTATION_DATE_FORMAT = "yyyy.MM.dd
HH:mm:ss z"
```

- 説明 :Service Manager ではインシデントの説明は必須の属性です。BSM で設定されていない場合，この値が取得されます。空白の文字列は許可されていません。

```
private static final String EMPTY_DESCRIPTION_OVERRIDE = "<none>"
```

- テキストのログ :オペレーション管理イベント・プロパティが Service Manager のインシデント・アクティビティ・ログに同期するときに，次のテキストは適切なオペレーション管理イベント・プロパティに対するプレフィックスになります。

  注 :このテキストは任意のロケールでローカライズ可能です。標準設定は次のとおりです。

```
private static final String ACTIVITY_LOG_TITLE = "[Title]\n"

private static final String ACTIVITY_LOG_TITLE_CHANGE = "Event title
changed to: "
```

```
private static final String ACTIVITY_LOG_STATE = "[State]\n"

private static final String ACTIVITY_LOG_STATE_CHANGE = "Event state
changed to: "

private static final String ACTIVITY_LOG_DESCRIPTION = "
[Description]\n"

private static final String ACTIVITY_LOG_DESCRIPTION_CHANGE = "Event
description changed to: "

private static final String ACTIVITY_LOG_SOLUTION = "[Solution]\n"

private static final String ACTIVITY_LOG_SOLUTION_CHANGE = "Event
solution changed to: "

private static final String ACTIVITY_LOG_ASSIGNED_USER = "[Assigned
User]\n"

private static final String ACTIVITY_LOG_ASSIGNED_USER_CHANGE =
"Event assigned user changed to: "

private static final String ACTIVITY_LOG_ASSIGNED_GROUP = "[Assigned
Group]\n"

private static final String ACTIVITY_LOG_ASSIGNED_GROUP_CHANGE =
"Event assigned group changed to: "

private static final String ACTIVITY_LOG_SEVERITY = "[Severity]\n"

private static final String ACTIVITY_LOG_SEVERITY_CHANGE = "Event
severity changed to: "

private static final String ACTIVITY_LOG_PRIORITY = "[Priority]\n"

private static final String ACTIVITY_LOG_PRIORITY_CHANGE = "Event
priority changed to: "

private static final String ACTIVITY_LOG_CONTROL_TRANSFERRED_TO = "
[Control Transferred To]\n"

private static final String ACTIVITY_LOG_CONTROL_TRANSFERRED_TO_
CHANGED = "Event control transfer state changed to: "

private static final String ACTIVITY_LOG_ANNOTATION = "[Annotation]
\n"

private static final String ACTIVITY_LOG_CA = "[Custom Attribute]\n"

private static final String ACTIVITY_LOG_CAUSE = "[Cause] "

private static final String ACTIVITY_LOG_OMI_CAUSE = "[OMi Cause] "

private static final String ACTIVITY_LOG_OMI_SYMPTOM = "[OMi
Symptom] "

private static final String ACTIVITY_LOG_DUPLICATE_COUNT = "
[Duplicate Count] "

private static final String ACTIVITY_LOG_PREVIOUS = "previous "
```

```
private static final String ACTIVITY_LOG_CURRENT = "current "
```

オペレーション管理イベントから BDM インシデント・プロパティのマッピング・テーブル

標準の統合により，次のオペレーション管理イベントのプロパティが Service Manager インシデント・プロパティに同期します。

オペレーション管理の UI(イベント)	オペレーション管理の UI(転送)	オペレーション管理の Web サービス(イベント)	Service Manager の Web サービス(インシデント)	Service Manager オブジェクト	Service Manager の UI	コメント
-	-	-	global_id	id	-	オペレーション管理イベントと Service Manager のインシデントとの間に `global_id` に対するマッピングおよび同期はありません。
ID	-	id	external_process_reference	external.process.reference	-	Service Manager のインシデント作成時のみに設定されます。
-	External ID	control_transferred_to external_id	reference_number	number	Incident ID	Service Manager のインシデント作成時のみに設定されます。

オペレーション管理の UI(イベント)	オペレーション管理の UI (転送)	オペレーション管理の Web サービス(イベント)	Service Manager の Web サービス (インシデント)	Service Manager オブジェクト	Service Manager の UI	コメント
Title	-	title	name	brief.description	Title	インデント作成時に設定されます。標準的な構成 :オペレーション管理のタイトルの変更のみが Service Manager のインシデント・アクティビティ・ログに同期されます。オペレーション管理のイベント・タイトルが 256 文字を超えるか, 改行文字を含んでいる場合 , その部分は切り捨てられ, タイトル全体が説明に付加されます。

オペレーション管理の UI(イベント)	オペレーション管理の UI (転送)	オペレーション管理の Web サービス(イベント)	Service Manager の Web サービス (インシデント)	Service Manager オブジェクト	Service Manager の UI	コメント
Description	-	description	descriptio-n	action	Descriptio-n	オペレーション管理イベントのタイトルと説明を組み合わせになることが可能です。詳細についてはタイトルのコメントを参照してください。標準的な構成 :双方向の同期です。
Solution	-	solution	solution	Resolution	Solution	標準的な構成 :双方向の同期です。
Lifecycle State	-	state	incident_ status	problem.status	Status	インデント作成時に設定されます。標準的な構成 : 「完了」状態のみが同期されます。
Severity	Severity	severity	urgency	severity	Urgency	インデント作成時に設定されます。標準的な構成 :値の変更はすべて同期されます。

オペレーション管理の UI(イベント)	オペレーション管理の UI(転送)	オペレーション管理の Web サービス(イベント)	Service Manager の Web サービス(インシデント)	Service Manager オブジェクト	Service Manager の UI	コメント
Priority	Priority	priority	priority	priority.code	Priority	インデント作成時に設定されます。標準的な構成:値の変更はすべて同期されます。
Assigned User	-	assigned_user login_name	is_requested_by party display_label	contact.name	Contact	標準設定では同期しません。
-	Assigned User	-	has_assigned party display_label	assignee.name	Assignee	オペレーション管理が Service Manager Incident の Web サービスに対して,[転送]タブのみに表示するためにリアルタイムでクエリを行います。オペレーション管理イベントとService Manager のインシデントとの間に同期は発生しません。

オペレーション管理の UI(イベント)	オペレーション管理の UI (転送)	オペレーション管理の Web サービス(イベント)	Service Manager の Web サービス (インシデント)	Service Manager オブジェクト	Service Manager の UI	コメント
-	-	control_ transferre-d_to initiated_ by	is_ recorded_ by party display_ label	opened.by	Opened By	標準設定では同期しません。
Assigned Group	Assigne-d Group	assigned_ group name	has_ assigned_ group functiona-l_group display_ label	assignment	Assignme-nt Group	オペレーション管理が Service Manager Incident の Web サービスに対して, [転送] タブのみに表示するためにリアルタイムでクエリを行います。オペレーション管理イベントとService Manager のインシデントとの間に同期は発生しません。

オペレーション管理の UI(イベント)	オペレーション管理の UI (転送)	オペレーション管理の Web サービス(イベント)	Service Manager の Web サービス (インシデント)	Service Manager オブジェクト	Service Manager の UI	コメント
Category	-	category	category	sub_category	Area	カテゴリは Service Manager インシデントの作成時にオペレーション管理によってのみ設定されますが, 標準設定では Service Manager で無視されます。
Subcategor-y	-	sub_ category	sub_ category	product_type	Subarea	サブカテゴリは Service Manager インシデントの作成時にオペレーション管理によってのみ設定されますが, 標準設定では Service Manager で無視されます。
Related CI	-	related_ci	is_ registere-d_for	logical.name	Affected CI	影響を受ける CI は, Service Manager のインシデント作成時にオペレーション管理によってのみ設定されます。

オペレーション管理の UI(イベント)	オペレーション管理の UI (転送)	オペレーション管理の Web サービス(イベント)	Service Manager の Web サービス (インシデント)	Service Manager オブジェクト	Service Manager の UI	コメント
Annotation-s	-	annotatio-n_list	activity_ log descriptio-n	update.action	Activity Log	オペレーション管理から Service Manager へのみ同期します。OOTB 同期が有効化されます。
Custom Attributes	-	custom_ attribute_ list	activity_ log descriptio-n	update.action	Activity Log	特定の Service Manager インシデント・プロパティ, またはアクティビティ・ログへの同期のための設定が可能ですService Manager のアクティビティ・ログに対しては, 同期は Operations Manageme-nt から Service Manager に対してのみ可能です。標準的な構成 :同期は無効化されています。

オペレーション管理の UI(イベント)	オペレーション管理の UI (転送)	オペレーション管理の Web サービス(イベント)	Service Manager の Web サービス (インシデント)	Service Manager オブジェクト	Service Manager の UI	コメント
Cause	-	cause-> control_ transferre-d_to external_id	is_ caused_ by -master_ referenc-e_number	Links the two SM Incidents	Related Records	要因イベントが Service Manager に同期されている場合 , 2 つの Service Manager インシデントが関連しています。そうでない場合 , オペレーション管理の要因に関する情報が Service Manager のインシデント・アクティビティ・ログに付加されます。標準的な構成 :同期が有効化されています。
Symptoms	-	symptom_ list	activity_ log descriptio-n	update.action	Activity Log	同期はオペレーション管理から Service Manager へのみになります。標準的な構成 :同期が有効化されています。

オペレーション管理の UI(イベント)	オペレーション管理の UI (転送)	オペレーション管理の Web サービス(イベント)	Service Manager の Web サービス (インシデント)	Service Manager オブジェクト	Service Manager の UI	コメント
Duplicate Count	-	duplicate_ count	activity_ log descriptio- n	update.action	Activity Log	同期はオペレーション管理から Service Manager へのみになります。標準的な構成 :同期が有効化されています。

エラー処理

イベントおよびそれ以降の変更を転送するオペレーション管理イベントの同期インターフェイスは，本項に記載されているエラー処理アルゴリズムを使用して，転送要求がリトライされるか廃棄されるかを決定します。

エラー処理は，統合の種類によって異なります。

- 387ページ「Groovy スクリプトの統合」
- 388ページ「Web サービス統合」
- 389ページ「HP Service Manager の統合」

Groovy スクリプトの統合

Groovy スクリプトを実装してご使用のアプリケーションを統合する場合のエラー処理は次のようになります。

`forwardEvent()` および `forwardChange()` の場合：

- true の返し :要求が FORWARDED とマークされ，転送要求がキューから削除されます。
- false の返し :要求はキューに戻されます。リトライは，要求が成功するかキューに留まっている時間が転送有効期限の時間(標準設定では 12 時間で，インフラストラクチャ設定で変更可能)を超えるまで 1 分間に 1 回行われます。新しいイベントおよび更新が，転送要求が受信された順に転送されます。要求が Non-RuntimeException エラーにより失敗すると，このサーバの他のリクエストすべてが遮断されます。未処理の更新を配信する前にキュー内の新しいイベントがすべて配信されます。

`forwardEvents()` の場合：

- true の返し :すべての転送イベント要求が FORWARDED とマークされ，それらの転送要求がキューから削除されます。すべてのイベントが FORWARDED に設定された転送ステータスを持っている場合，例外がログに記録されます。
- false の返し :FORWARDED とマークされていないイベントが存在する場合，標準的な例外処理で結果がない最初のイベントが処理されます。残存するイベントはキューに格納されたままで，後でリトライされます。

  `com.hp.opr.api.ws.adapter.ExternalProcessAdapter` インターフェイスで定義された `forwardEvent()` の標準的なエラー処理も参照してください。

`forwardChanges()` の場合：

- true の返し :すべての転送変更要求が FORWARDED とマークされ，それらの転送要求がキューから削除されます。すべての変更要求が FORWARDED に設定された転送ステータスを持っている場合，例外がログに記録されます。
- false の返し :FORWARDED とマークされていない変更が存在する場合，標準的な例外処理で結果がない最初の変更が処理されます。残存する変更はキューに格納されたままで，後でリトライされます。

  `com.hp.opr.api.ws.adapter.ExternalProcessAdapter` インターフェイスで定義された `forwardEvent()` の標準的なエラー処理も参照してください。

`receiveChanges()` の場合：

- エラー処理は `com.hp.opr.api.ws.adapter.ExternalProcessAdapter` インターフェイスで定義された `receiveChange()` の場合と同じです。

Groovy スクリプトが例外をスローする場合，次のように処理されます。

- `org.apache.wink.client.ClientWebException`:例外は，サーバから返された HTTP ステータス・コードに対してクエリされます。HTTP ステータス・コードが見つかると，そのステータス・コードは 388ページ「Web サービス統合」に記載されている Web サービス統合のケースと同様に処理されます。ステータス・コードが存在しない場合，その例外は他の例外と同様に処理されます。詳細については以下を参照してください。
- `ClientWebException` 以外のすべての例外:例外は再帰的に根本原因の例外について検索され，次のように解釈されます。
  - RuntimeException :エラーは `opr-event-sync-adapter.log` ログ・ファイルにログ記録され，その要求は FAILED としてマークされます。この要求に対するリトライはありません。
  - Non-RuntimeException :この場合の例は IOException，SocketException などになります。エラーのログが `opr-event-sync-adapter.log` ログ・ファイルに記録されます。この場合，サーバへの接続が未来のある時点で回復し，要求を送信可能になることが期待されます。要求はキューに戻されます。リトライは，要求が成功するかキューに留まっている時間が転送有効期限の時間(標準設定では 12 時間で，インフラストラクチャ設定で変更可能)を超えるまで 1 分間に 1 回行われます。新しいイベントおよび更新が，転送要求が受信された順に転送されます。要求が Non-RuntimeException エラーにより失敗すると，このサーバの他のリクエストすべてが遮断されます。未処理の更新を配信する前にキュー内の新しいイベントがすべて配信されます。

Web サービス統合

オペレーション管理によって直接呼出し可能なイベント同期 Web サービスのエンドポイントを実装すると，エラー処理は次のようになります。

- **2xx の HTTP ステータスが返されます。** 特定の呼び出しによって，たとえば，新規イベントの POST では 200(OK) または201(作成済み) がともに受け入れられ，イベント更新の PUT では 200(OK) が期待されます。HTTP ステータス 202(許容済み) も，PUT や POST で受け入れられます。
  - 返されるオブジェクトは，ID が POST の外部イベント ID に設定された状態の OPR イベントである必要があります。
  - PUT の返されるオブジェクトは無視されます。
- **3xx の HTTP ステータスが返されます。** これらはリダイレクトのステータスです。リダイレクトはサービスによってサポートされません。これらは 4xx と同様に処理されます。このため，要求が FAILED としてマークされ，それ以上のリトライは行われません。
- **4xx の HTTP ステータスが返されます。** 4xx のどのステータスもクライアントの要求でエラーとして見なされます。このため要求は FAILED としてマークされ，それ以上のリトライは行われません。
- **5xx の HTTP ステータスが返されます。** 5xx のどのステータスもサーバ上でエラーとして見なされます。この場合，サーバが未来のある時点で回復し，要求を受信可能になることが期待されます。要求はキューに戻されます。リトライは，要求が成功するかキューに留まっている時間が転送有効期限の時間(標準設定では 12 時間で，インフラストラクチャ設定で変更可能)を超えるまで 1 分間に 1 回行われます。新しいイベントおよび更新が，転送要求が受信された順に転送されま

す。要求が 5xx エラーにより失敗すると，このサーバの他のリクエストすべてが遮断されます。未処理の更新を配信する前にキュー内の新しいイベントがすべて配信されます。

- サーバとの通信を試行している際に遭遇したあらゆる種類の IOException は，再度キューに格納される要求を生じさせ，5xx の HTTP ステータスについて記載されているとおりにリトライされるという結果になります。

HP Service Manager の統合

`ServiceManagerAdapter` Groovy スクリプトは Apache Wink クライアントを使用して HP Service Manager と通信します。このため，HP Service Manager によって返された HTTP エラー・ステータスがある場合，`ClientWebException` をスローします。このエラーの種類が HP Service Manager の統合でどのように処理されるかについては，387ページ「Groovy スクリプトの統合」を参照してください。

第5部分

BSM でのレポート

第14章

汎用レポート・エンジン API

プロファイル・データベースへのAPIレベル・クエリを作成するための推奨される方法は，カスタム・クエリ・ビルダを使用してクエリを作成することです。カスタム・クエリ・ビルダによりグラフィカルなユーザ・インタフェースを使用したクエリの作成が可能になり，レポートの生成，異なる形式のデータの抽出，およびサードパーティまたはカスタム・ツールと使用可能なクエリURLの生成が簡単になります。詳細については，『BSMユーザ・ガイド』の「Building a Custom Query Using Custom Query Builder」を参照してください。

汎用レポート・エンジンAPIにより，次の方法を使用したクエリの手動作成も可能になります。

- **Webブラウザ**：要求はHTMLクエリとして送信され，データがHTMLか，Microsoft Excelで開ける，またはカスタム・ツールで処理できるCSV (カンマ区切りの値) ファイルとして返されます。
- **Webサービス**：リターン・オブジェクトには，CSV形式のデータが含まれています。

本章の残りの部分では，クエリの手動作成方法について説明します。

前提条件

APIのユーザは，SQLの構文，BSM管理およびアプリケーションに精通している必要があります。Webサービスを介したAPIのユーザは，SOAP仕様，およびC++，Javaなどのオブジェクト指向プログラミング言語にも精通している必要があります。

権限

以下のAPIクエリ構文を使用してデータにアクセスするためのクエリの場合，クエリで渡されるユーザおよびパスワード・パラメータは，システム閲覧者またはスーパーユーザ権限のあるユーザのものである必要があります(権限マネージャでの権限設定の詳細については，『BSMプラットフォーム管理ガイド』の「権限の概要」を参照してください)。

設定

API オプションを設定するには，[**管理**]＞[**プラットフォーム**]＞[**セットアップと保守**]＞[**インフラストラクチャ設定**]を選択します。

- [**ファウンデーション**]を選択します。
- [**汎用データ エンジン Open API**]を選択します。
- [汎用データ エンジン Open API - 汎用データ エンジン Open API 設定]テーブルで，次を検索します。
  - **最大行数**：返される最大データ行数を変更します。
  - **Open API を有効化**：必要に応じて，汎用レポート・エンジンAPIの使用を有効化または無効化します。

サンプル上のメタデータの取得

クエリの作成時には，サンプルのデータ表記を理解している必要があります。一般的にクエリされるサンプルの詳細とそのフィールドの説明については，『BSMアプリケーション管理ガイド』の「データ・サンプル」を参照してください。

詳細サンプルの取得

特別レポートのニーズがあるユーザは，MBean Inspector を使用してすべてのサンプルとそのフィールドのリストを取得できます。ブラウザに次の URL を入力して，[MBean Inspector] ページにアクセスします。

http://<サーバ>[:port]/jmx-console/HtmlAdaptor?action=inspectMBean&name=Topaz%3Aservice%3DMeta-Data+Manager

標準設定のポート番号は 8080 です。このポートが正しくない場合，正しいポート番号をシステム管理者に問い合わせてください。

JMX コンソールの認証資格情報を入力します。認証資格情報がわからない場合，システム管理者にお問い合わせください。

[MBean Inspector] ページで，操作 **showMetaDataDBMapping** の隣の [**呼び出し**] ボタンをクリックします。ビーンによって各サンプルのフィールドのリストが返されます。

返されるデータ

要求がブラウザから行われても，Web サービスによって行われても，同じデータが返されます。ブラウザを使用した場合，データは応答本体にあり，Web サービスの場合，データはリターン・オブジェクトにあります。

Web ブラウザ応答本体

クエリがブラウザから送信されると，応答 CSV または HTML にはデータ，エラー・コード，メッセージのいずれかが含まれます。返される行数が最大を超えると，データの最後の行は `Returned X of Y rows` です。*X* は返される行数で，*Y* はクエリの条件を満たす実際の行数です。エンジン・レベルでエラーがある場合，HTTP 成功コードが返されますが，応答の本体は <error code>，<error message> です。

Web サービス・リターン・オブジェクト

Web サービス・リターン・オブジェクトには次が含まれています。

- **retval :** データまたはエラー・メッセージ
- **errorCode :** エラー・コード (タイプ int) 。可能なエラー・コードは次のとおりです。
  - 0 - 成功
  - 100 - 権限エラー
  - 101 - 処理エラー
  - 102 - Open API が無効化されている
- **origRowCount :** クエリによって返される必要があった実際の行数 (タイプ int) 。返される行数が最大を超えると，**origRowCount** フィールドが最大を超えなかった場合にクエリによって返される実際の行数に設定されます。

ブラウザによるクエリ

ブラウザによるクエリ時に，getData サービスが URL によって呼び出されます。
`http://<`**サーバ**`>[:port]/topaz/gdeopenapi/GdeOpenApi?method=getData&user=<`**ユーザ名**`>&password=<`**パスワード**`>&query=<`**クエリ**`>`

URL には，オプションの resultType パラメータを含めることができます。
`http://<`**サーバ**`>[:port]/topaz/gdeopenapi/GdeOpenApi?method=getData&user=<`**ユーザ名**`>&password=<`**パスワード**`>&query=<`**クエリ**`>&resultType=csv`

ポートの指定は，標準以外のインストールの場合のみ必要です。正しいポート番号については，システム管理者にお問い合わせください。

標準設定の戻り値の型は HTML です。`resultType=csv` が指定されると，カンマ区切り値のファイルが返されます。

Web サービスの使用

API Web サービスにより，ユーザ名，パスワード，SQL に似た選択ステートメントから構成されるクエリの送信が可能になります。エンジンは，ステートメントを解析できない場合，またはクエリ実行に問題がある場合はエラー詳細を返します。エラーがない場合，クエリの結果が返されます。

SOAP WSDL は次の URL にあります。

`http://<サーバ>[:port]/topaz/gdeopenapi/services/GdeWsOpenAPI?wsdl`

ポートの指定は，標準以外のインストールの場合のみ必要です。正しいポート番号については，システム管理者にお問い合わせください。

サポートされている SQL 構文

サポートされている言語は SQL のサブセットで，次のキーワード，修飾子，演算子をサポートしています。

- SELECT
- WHERE
- FROM
- TOP
- HAVING
- AS キーワードによるエイリアシング
- 論理演算子 OR，AND，NOT
- DISTINCT 修飾子 (選択リスト・アイテムに対してのみサポート)
- IN 演算子 :内部選択が IN 演算子の値を返すために使用可能です。
- BETWEEN 演算子
- IS NULL(IS NOT NULL はサポートされていません)
- LIKE :ワイルドカード文字はアスタリスク(*) です。パーセント記号 (%) は使用しないでください。アスタリスクはそれ自体によって使用できません(LIKE *) 。ほかの文字と使用する必要があります。
- 算術演算子 : +, -, *, /, (,)
- コンパレータ :=，IS，!=，<>，>，>=，<，<=
- ORDER BY 修飾子，ASC 修飾子，DESC 修飾子

サポートされている関数

サポートされている関数は次のとおりです。

- MAX
- MIN
- SUM
- COUNT
- AVG
- STDDEV
- SUMOFSQR
- LOG
- CEIL
- FLOOR
- MOD
- SQRT
- REPLACENULL(Oracle の NVL，Microsoft SQL Server の ISNULL と同等)
- IF
- 400ページ「byTime 関数」

クエリの制約

次の制約がサービスに送信されるクエリに適用されます。

- 1 つのモニタ・タイプのみが 1 つのクエリで選択可能です。
- アスタリスク(*) は，LIKE 演算子との組み合わせ以外ではワイルドカード文字としてサポートされていません。乗算演算子としてサポートされています。
- 内部選択と結合はサポートされていませんが，例外が 1 つあります。内部選択は IN 句に対する値を返すために使用できます。
- ORDER BY 句には，ORDER BY 1 などのカラム番号が必要です。ORDER BY というカラム名はサポートされていません。
- エンジンでは，WHERE 句でクエリに時間制限 (time_stamp フィールドの条件) が含まれていることが要求されます。
- GROUP BY 句はサポートされていません。エンジンでは集計関数を持たないすべてのフィールドが GROUP BY フィールドとして処理されるため，これは不要です。
- 空白または特殊文字を含む文字列から構成されているフィルタ(たとえば，`where bb_guid IN (a b, c)`)を手動で定義するとき，その空白と特殊文字列を引用符で囲む必要があります(たとえば，`where bb_guid IN (`a b', c)`)。[フィルタビルダ] ページでフィルタを作成する場合，BSM では自動的に引用符が追加されます。特殊文字は，数字，英字，および次の文字以外の文字として定義されます。"_", "$", "#".
- 1 つ以上の単一引用符文字を含む文字列から構成されているフィルタを定義する場合，各インスタンスの横に 2 つ目の単一引用符文字を追加する必要があります。たとえば，`szTransactionName = ('Login_to_O'Brien')` を `szTransactionName = ('Login_to_O''Brien')` に変更します。
- 返されたデータのカラムは `Column 0, Column 1` のようにラベル付けされます。意味のあるカラム・ラベルを返すには，SQL AS 演算子を使用します。たとえば，`Select time_stamp as TimeStamp` です。AS 演算子の使用により，カラム・ラベルは `TimeStamp` となります。
- "COUNT (DISTINCT <フィールド>)" 構文はサポートされていません。代わりに "COUNT DISTINCT (<フィールド>)" 構文を使用します。

日時値

クエリと戻りデータの時間が 1970 年 1 月 1 日以来の秒単位で指定されています。Microsoft Excel を使用して，時間値を秒，日時に変換できます。

時間はタイム・スタンプ・フィールドに最も一般的に使用されます。

クエリで使用するための GMT 時間を取得するには :

日付形式セルに日付と時間を入力し，別の一般形式セルに次の式を入力します。
=(<日付セル> - 25569) * 86400

ローカル・タイムゾーン用に修正するには :

タイムゾーンのオフセット × 3600 秒を結果に追加します。たとえば，中央ヨーロッパ(GMT + 2) の場合は次のようになります。
=(<日付セル> - 25569) * 86400 + (2 * 3600)

クエリからの時間値を GMT 日付として Excel で表示するには，次の手順を実行します。

日付形式をセルに使用して，次の式を入力します。
=<タイム・スタンプ> / 86400 + 25569

ローカル・タイムゾーン用に修正するには :

タイムゾーンのオフセット × 3600 秒をタイム・スタンプから減算します。たとえば，アメリカ東部標準時間 (GMT - 3) の場合，次のようになります。
=(<タイム・スタンプ> - (-3 * 3600))/ 86400 + 25569

byTime 関数

汎用レポート・エンジン SQL では，期間別にグループ化されたデータを返す関数 **byTime** がサポートされています。たとえば，byTime 関数を使用しない場合，過去 1 日に対するトランザクションの平均応答時間のクエリで 1 つの値が返されます。byTime 関数を使用して，過去 1 日の時間ごとのトランザクションの平均応答時間を表示できます。この場合，値は過去 24 時間の時間ごとに返されます。

関数の構文は次のとおりです。
byTime(<*timefield* >, <*step value*>, <*number of step*>, <*offset*>)

引数	説明
timefield	通常は timestamp フィールド
step value	次のうちいずれか : 10 - 秒 20 - 分 30 - 時間 40 - 日 50 - 週 60 - 月 70 - 四半期 80 - 年
number of step	グループ化対象の step value で指定された単位数
offset	GMT からのタイムゾーン・オフセット (時間単位) 。正の数は GMT の東にあるタイムゾーンを示します。負の数は GMT の西にあるタイムゾーンを示します。

たとえば，3 日ごとに 1 つの値 (GMT から東 1 時間に修正された) を返すには，次のように入力します。

byTime(time_stamp, 40, 3, 1)

クエリの例

次に，データベースから異なるタイプのデータを取得するクエリ URL の例をいくつか示します。

ss_t サンプルの例

この例では，指定の測定値およびモニタの SiteScope サンプルの平均値の取得を示します。

http://myServer/topaz/gdeopenapi/GdeOpenApi?method=getData&user=admin&password=admin-&query=select szMeasurementName, szMonitorName, avg(dValue) from ss_t where u_iStatus=1 and time_stamp > 123456 and szMeasurementName = `myMeasurmentName' and szMonitorName = `myMonitorName'

trans_t サンプルの例

この例では，平均応答時間の取得を示します。これは，BPM データからの指定期間の Springfield_Location アプリケーションにおける Springfield_infra_ems_login トランザクションに対し，分によってグループ化され，GMT + 3 にオフセットされた時間です。

http://myServer/topaz/gdeopenapi/GdeOpenApi?method=getData&user=admin&password=admin-&query=select byTime(time_stamp, 20, 1, 3.0), application_name as ApplicationName, szTransactionName as TransactionName, AVG(dResponseTime) from trans_t where time_stamp>=1126594800.64 and time_stamp<1126596000.64 and application_name='Springfield_Location' and szTransactionName='Springfield_infra_ems_login'

rum_action_t サンプルの例

この例では，RUM によって測定される URL ごとの合計サーバ時間の取得を示します。

http://myServer/topaz/gdeopenapi/GdeOpenApi?method=getData&user=admin&password=admin-&query=select application_name as ApplicationName,action_name as ActionName,action_descriptor,AVG(tot_server_time) from rum_action_t where time_stamp>=1304197200.64 and time_stamp<1306702800.64 and application_name='EC2%20jpetstore' and action_name='Sign In'

rum_application_stats_t サンプルの例

この例では，RUM によって測定される，アプリケーションを提供する各サーバ上のアプリケーションのアクションの平均サーバ時間の取得を示します。

```
http://MyServer/topaz/gdeopenapi/GdeOpenApi?method=getData&user=admin&-
password=admin&query=select application_name as ApplicationName,
server_host_name as hostName,Avg(tot_server_time) as serverTime from
rum_application_stats_t where time_stamp>=1304197200.64 and time_
stamp<1306702800.64 and application_name='EC2 jpetstore' group by
application_name, server_host_name
```