HP Operations Manager i

ソフトウェア・バージョン : 10.00

OMi 拡張性ガイド

ドキュメント・リリース日 : 2015 年 1 月（英語版）
ソフトウェア・リリース日 : 2015 年 1 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。

Citrix ® および XenDesktop ® は Citrix Systems, Inc. およびその 1 つ以上の子会社の登録商標で，米国およびその他の国の特許商標庁に登録されている可能性があります。

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

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

Linux ® は Linus Torvalds の米国およびその他の国における登録商標です。

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。

Red Hat ® は，Red Hat, Inc. の米国およびその他の国における登録商標です。

UNIX ® は The Open Group の登録商標です。

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。
https://softwaresupport.hp.com/group/softwaresupport/search-result?keyword=.

このサイトでは，HP Passport アカウントが必要です。アカウントがない場合は，HP Passport の［サインイン］ページで［**Create an account**］ボタンをクリックしてください。

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。https://softwaresupport.hp.com

連絡先情報と，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・サポートではセルフ・ソルプ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・カスタマーとの意見交換
- 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，https://softwaresupport.hp.com にアクセスし，［**登録**］をクリックしてください。

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。https://softwaresupport.hp.com/web/softwaresupport/access-levels

HP Software Solutions& 統合およびベスト・プラクティス

HP ソフトウェア・カタログの製品の連携方法，情報交換の方法，ビジネス・ニーズの解決方法を調べるには，HP Software Solutions Now（https://h20230.www2.hp.com/sc/solutions/index.jsp）にアクセスしてください。

さまざまなベスト・プラクティスのドキュメントや資料を入手するには，Cross Portfolio Best Practices Library（https://hpln.hp.com/group/best-practices-hpsw）にアクセスしてください。

コンテンツ

第I部: OMi の拡張

この『Operations Manager i 拡張性ガイド』では，HP Operations Manager i（OMi）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。

- 「コンテンツの展開」(19ページ)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 OMi 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。

- 第 2 部：「実行時サービス・モデルへのデータの追加」(62ページ)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・ルールを拡充して実行時サービス・モデル（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「コンテンツの展開」(19ページ)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。

- 「イベント処理インタフェース」(148ページ)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。

- 「OMi UI とほかのアプリケーションとの統合」(173ページ)では，ドリルダウン URL 起動を使用して OMi 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。

- 「オペレータ機能およびイベント変更検出の自動化」(186ページ)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。

  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。

- 「外部イベント・プロセスの統合」(244ページ)では，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。イベント同期 Web サービス・インタフェースは，イベント処理に外部プロセスを統合するために，外部アプリケーションとの統合を可能にします。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。

- 「Web サービス・インタフェース」(324ページ)では，インテグレータが OMi の機能を自動化でき

る Web サービスについて説明します。

- 「Groovy スクリプト」(376ページ)では，カスタマイズを実装するための Groovy スクリプトの開発およびデプロイの方法について説明します。
- 「サービス状況」(418ページ)では，Service Health ルール API および Service Health の外部 API について説明します。
- 「ダウンタイム REST サービス」(456ページ)では，ダウンタイムの取得，更新，作成，削除ができる RESTful Web サービスについて説明します。

第1章: 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 Operations Manager i モデリング・ガイド』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 ディスカバリおよび統合コンテンツ・ガイド』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ます。

    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。

<*OMi_HOME*>/conf/opr/topology-sync/sync-packages

この情報は『OMi 拡張性ガイド』に含まれています。

- **HP Operations Manager i** (OMi).

- 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II部: 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

第2章: 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。

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 Operations Manager i 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，「トポロジ」(23ページ)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

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 を入力します。イベントを RTSM 内の適切な CI にマップする必要があります。
- 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，「イベント・タイプ・インジケータと状況インジケータ」(36ページ)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，「相関処理ルールとマッピング」(50ページ)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，「追加のイベント処理」(52ページ)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，「ツール」(53ページ)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，「ビュー・マッピング」(54ページ)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，「グラフ」(55ページ)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

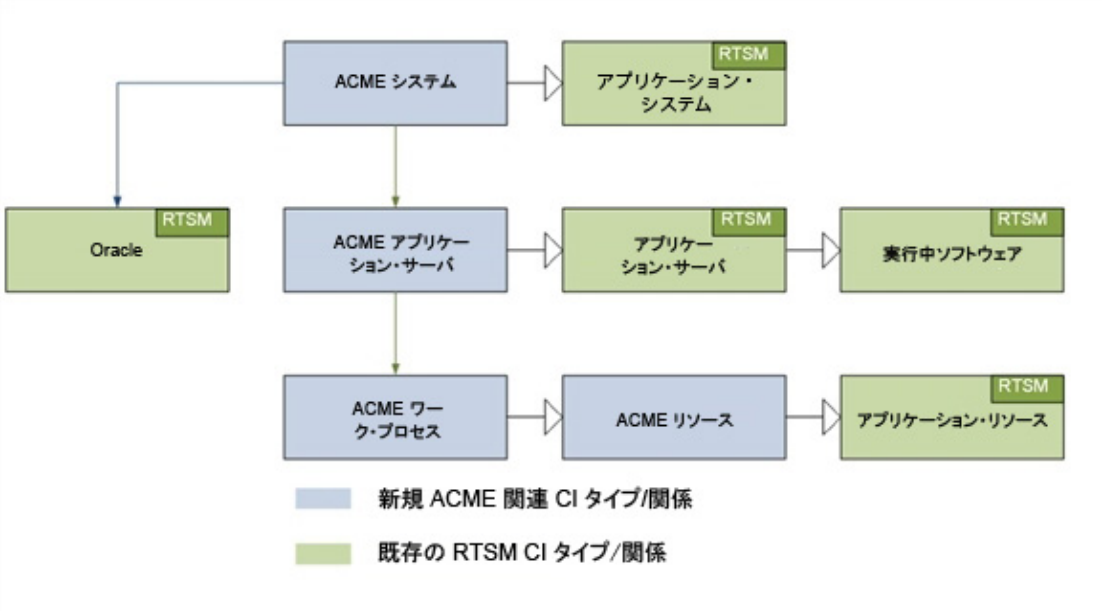
OMi 監視ソリューションに新規アプリケーションを統合し，新規アプリケーションのトポロジ・ビューを作成するには，次の手順を実行します。

- 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 概念の取り扱いの詳細については，『OMiモデリング・ガイド』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。

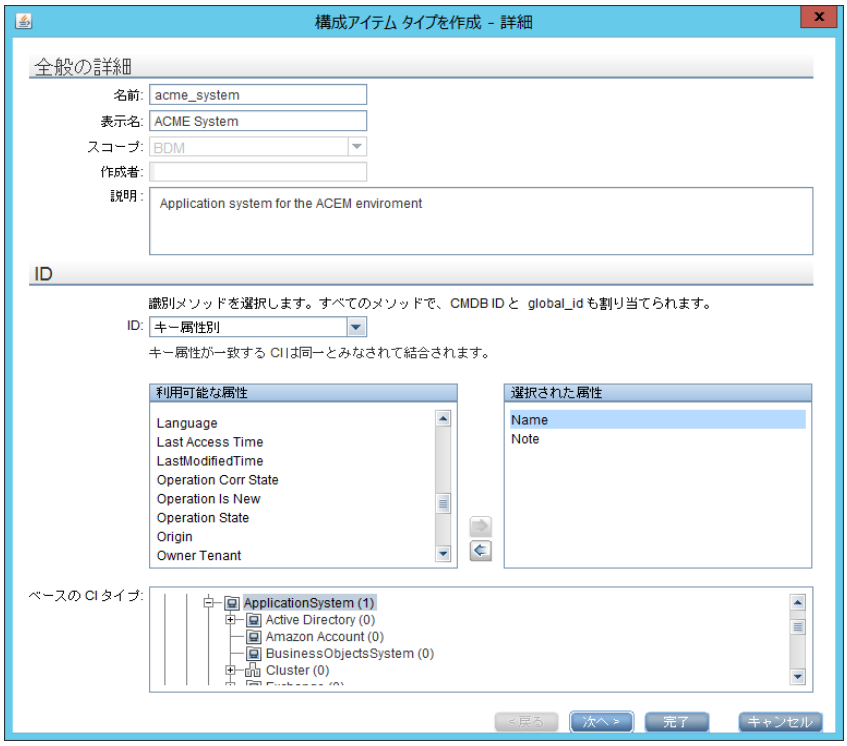
   ［RTSM 管理］ > **［モデリング］** > **［CI タイプ マネージャ］**

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「新しい CI タイプのキー属性値の設定」(26ページ)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

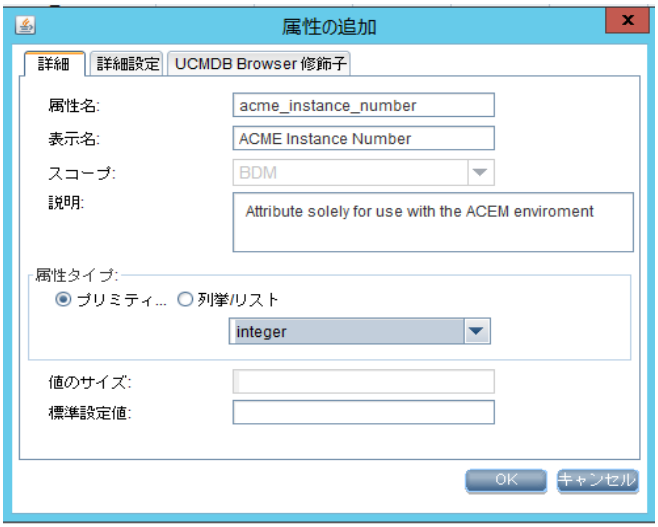
1. ［Create Configuration Item Type-Attributes］ダイアログで，CI タイプのキー属性（ACME System など）を設定します。

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）。

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. ［**完了**］をクリックします。

新規アプリケーションの関係の確立

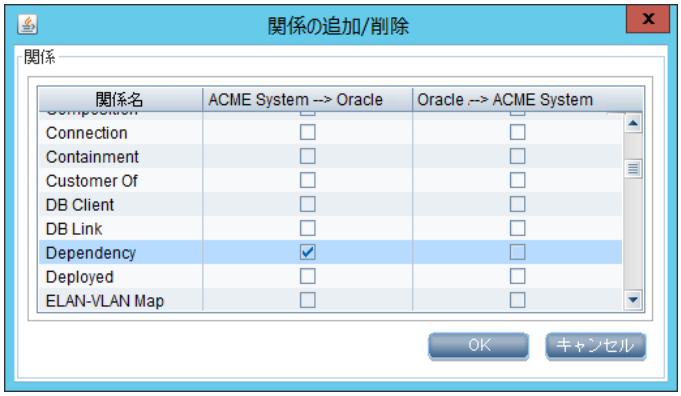
新規アプリケーションを統合するには，次にアプリケーションの関係を確立します。

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. ［関係の追加 / 削除］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。「RTSM/HP データ・フロー管理 ディスカバリによる CI の作成」(30ページ)を参照してください。
- 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。「トポロジ同期による CI の作成」(31ページ)を参照してください。

トポロジ同期の詳細については，「実行時サービス・モデルへのデータの追加」(62ページ)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。

「ディスカバリ方法を選択する際の考慮事項」(32ページ)も参照してください。

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 Operations Manager i 文書ライブラリの『データ・フロー管理ガイド』および『RTSM 開発者向け参考情報ガイド』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 Service Navigator 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，OMi と連携できる次の 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，「実行時サービス・モデルへのデータの追加」(62ページ)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。

- データ・フロー管理ガイドの使用が適している場合

  次の場合にデータ・フロー管理ガイドを使用します。

  - RTSM にデータを追加するために，すでにデータ・フロー管理ガイドを使用しているか，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。『データ・フロー管理ガイド』を使用することをお勧めします。これが 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

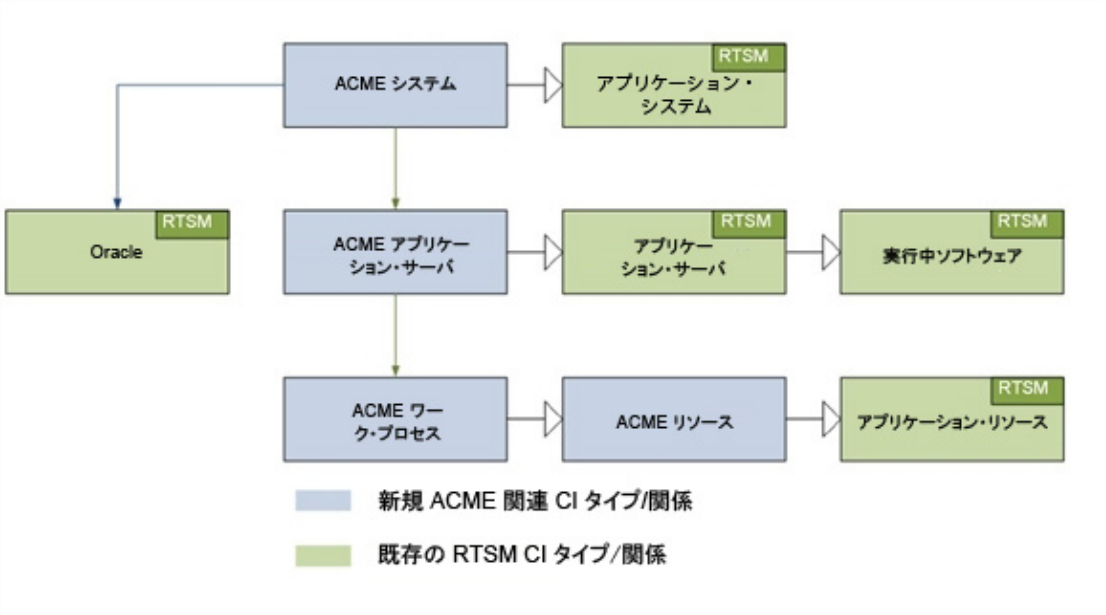
  次の場合は，同期パッケージに用意されている標準設定のトポロジ同期ルールを使用します。

  - RTSM へのデータの追加にデータ・フロー管理ガイドを使用しておらず（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

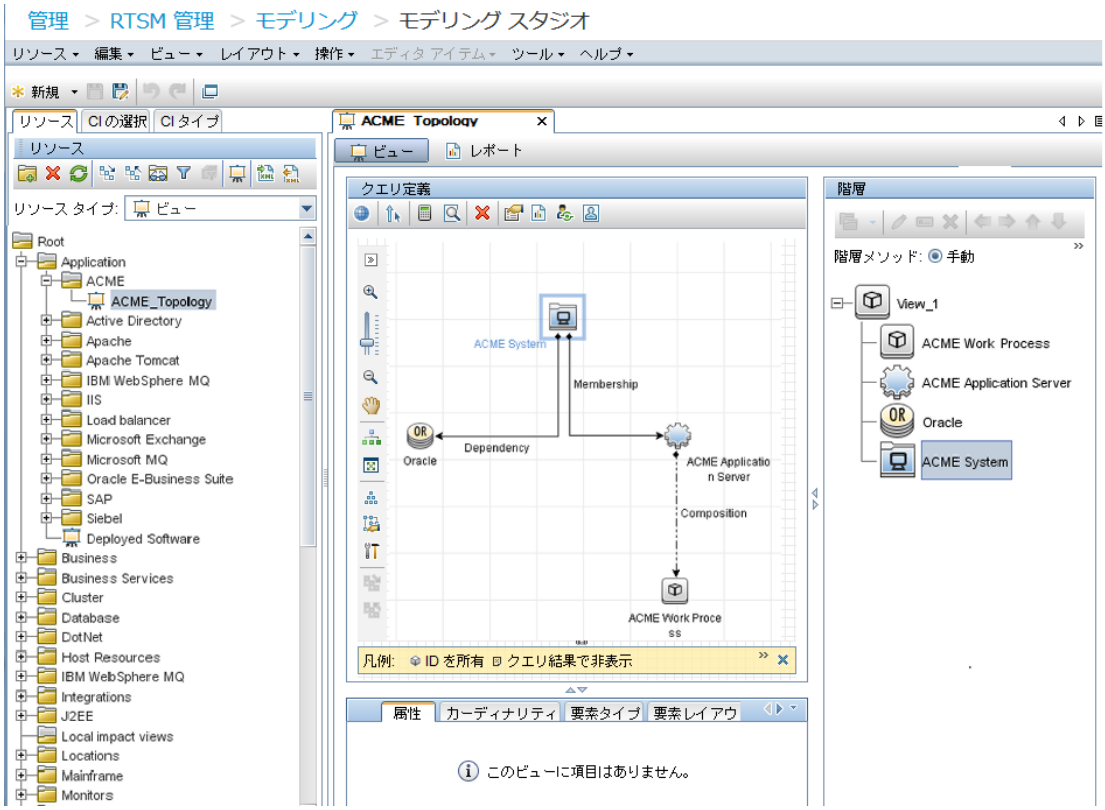


この図は，ACME システムと Oracle データベースの間にクロスドメイン関係がある ACME モデルを示しています。ACME ディスカバリが Oracle データベース CI を作成するために十分なキー属性がない場合，この依存関係を強化ルールによって生成できます。

強化ルールの作成および管理の詳細については，『モデリング・ガイド』の「Enrichment Manager」の項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。



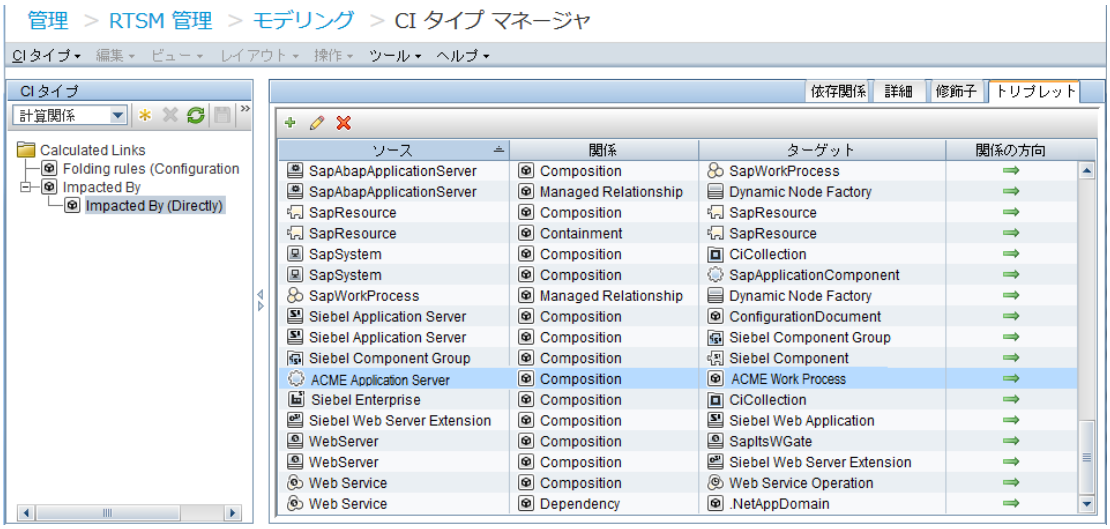
RTSM Modeling Studio の ACME トポロジ・ビュー

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，『モデリング・ガイド』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。

第4章: 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，OMi 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，OMi 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。

CI へのイベントのマッピング

OMi の高度なイベント監視機能および状況監視機能にとって，RTSM でイベントが適切な CI にマップされていることは，必要不可欠な要件です。

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）

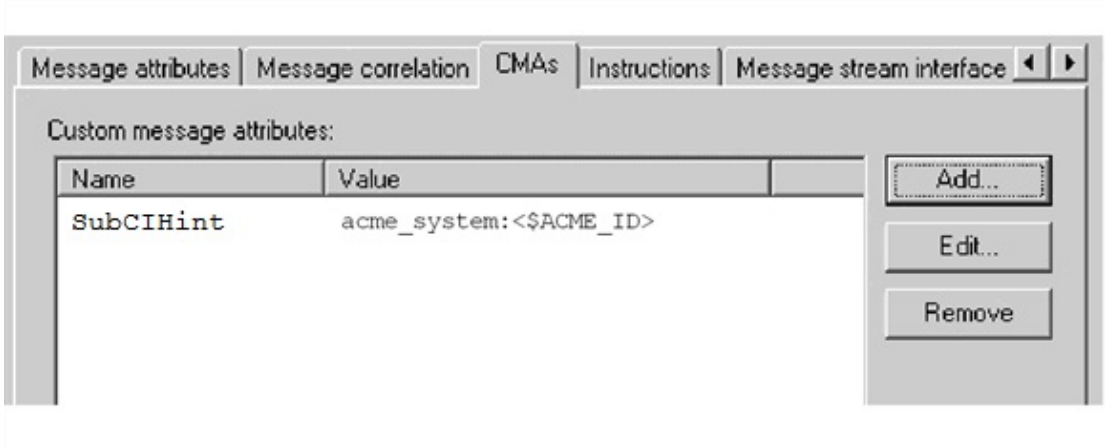
区切り文字（:）は，OMi インフラストラクチャ設定で設定できます。詳細については，OMi オンライン・ヘルプを参照してください。

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。



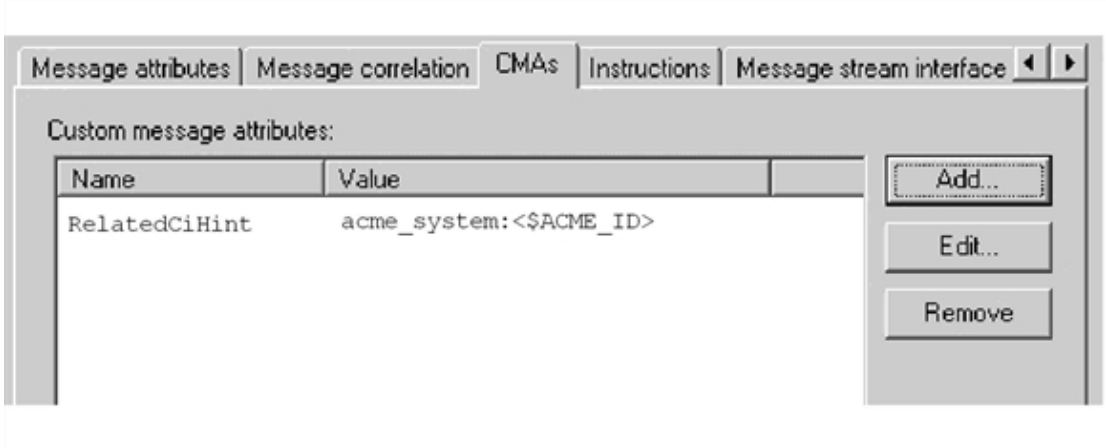
HPOM メッセージでのカスタム・メッセージ属性 RelatedCiHint の設定

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・インジケータと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）を作成できるようにするには，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）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で特定のタイプのイベントが生成される場合，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。「KPI への HI の割り当て」(48ページ) 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，「新規アプリケーションの統合」(23ページ)で説明しました。また，「」(24ページ)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。「ETI および HI の概要」(41ページ)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	重要度	ポリシー
ACME System	HI	ACME System Status	Available	Normal	ACME_ SystemStatus001
ACME System	HI	ACME System Status	Unavailable	Critical	ACME_ SystemStatus001
ACME Application Server	HI	ACME Application Server Status	Available	Normal	ACME_ AppServerStatus_001
ACME Application Server	HI	ACME Application Server Status	Unavailable	Major	ACME_ AppServerStatus_001
ACME Work Process	ETI	Job Aborted	Occurred		ACME_opcmsg_001
ACME Work Process	ETI	Job Start Passed	Occurred		ACME_opcmsg_002

ETI および HI の概要 (続き)

CI タイプ表示名	カテゴリ	名前	値	重要度	ポリシー
ACME Work Process	HI	Job Queue Length	Normal	Normal	ACME_JobQueue001
ACME Work Process	HI	Job Queue Length	Too Long	Major	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，OMi ユーザ・インタフェースの次の領域で作成します。

［**管理**］ > ［**サービス状況**］ > ［**状況インジケータとイベント タイプ インジケータ**］

HI と ETI の作成方法の詳細については，OMi 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**］ > ［**サービス状況**］ > ［**状況インジケータとイベント タイプ 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。

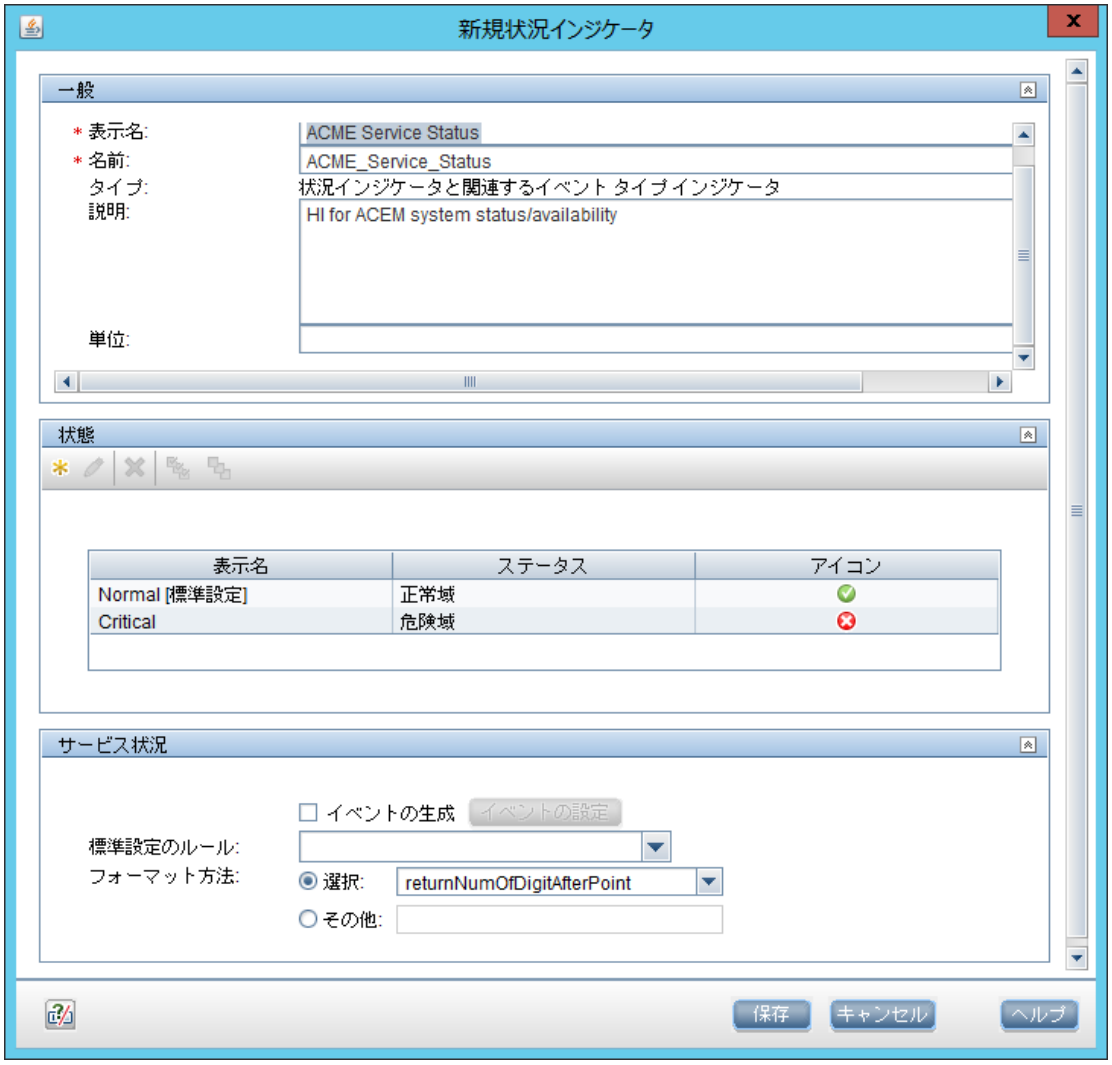
4. ［新規状況インジケータ］ダイアログの［一般］領域に次の情報を入力します。

   ［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］フィールドで，［サービス状況］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・ボックスをクリアします。

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。

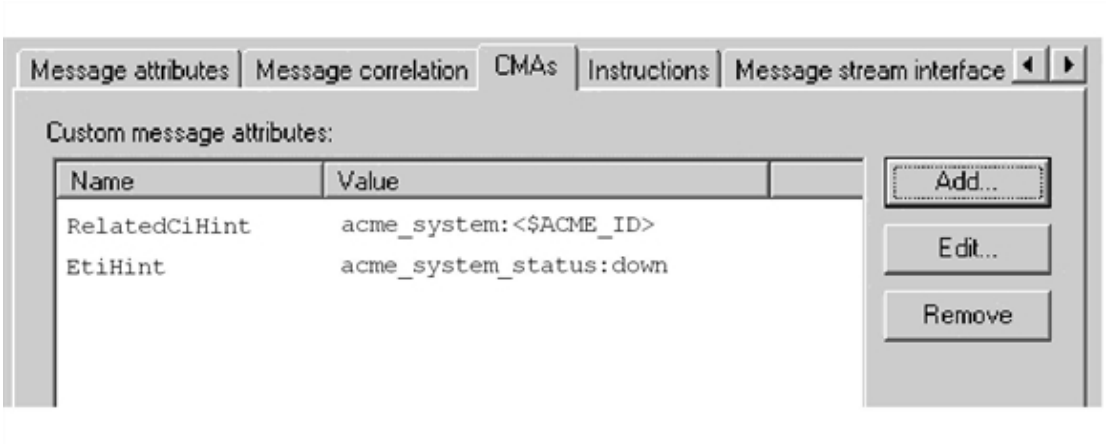
イベントへのイベント・タイプ・インジケータの割り当て

イベントに ETI を割り当てる方法は 2 つあります。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/OMi 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，インジケータ・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。

- HPOM ポリシー内にカスタム・メッセージ属性 EtiHint を設定する

  「ETI および HI の概要」(41ページ)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。

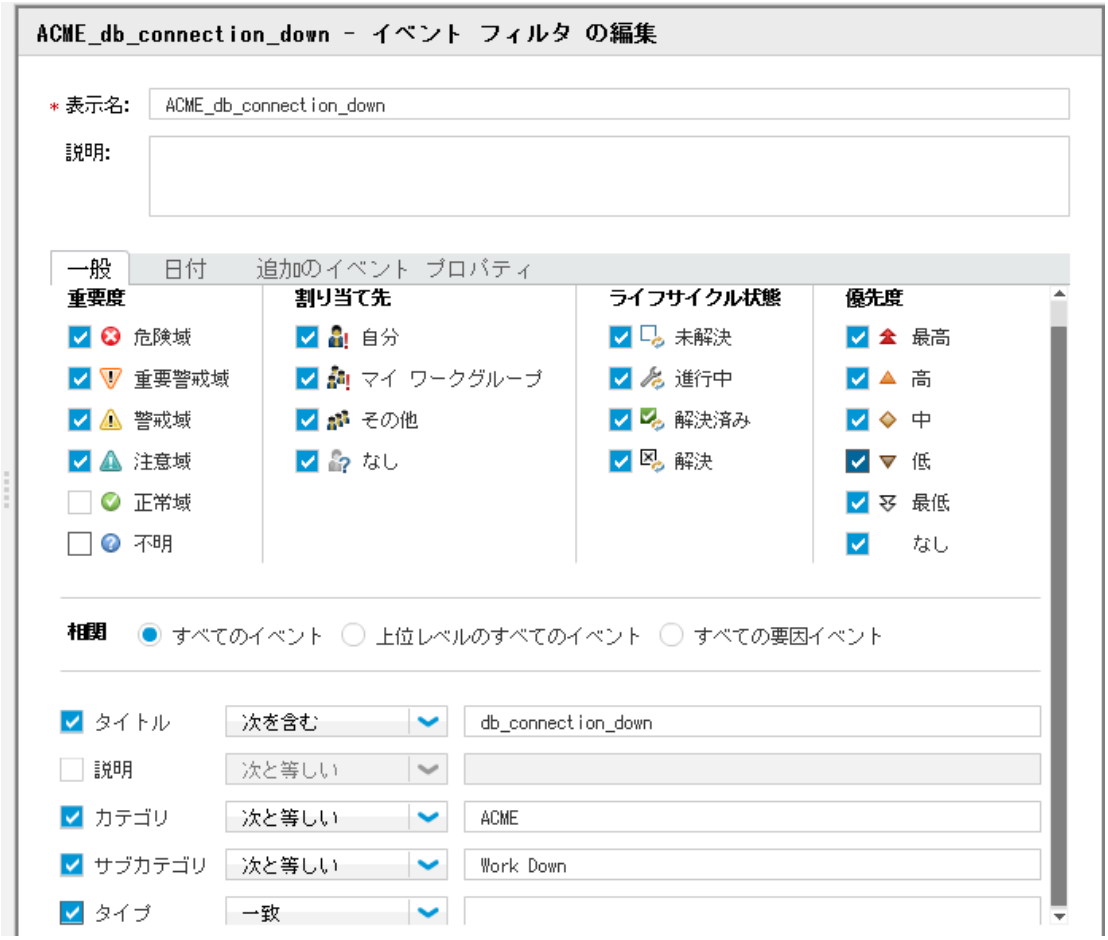
  次の図に CMA EtiHint の設定例を示します。



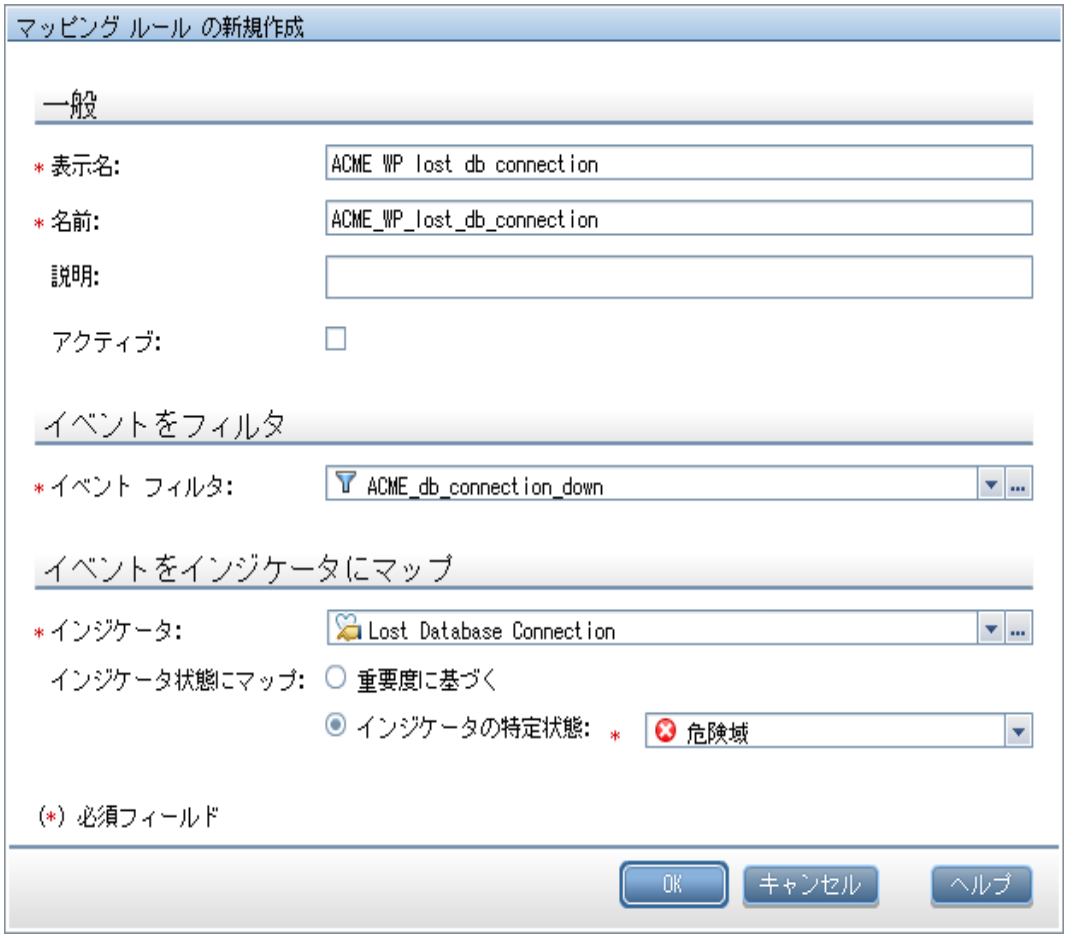
- インジケータ・マッピング・ルールの使用

  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，上に示すフィルタ構成を使用して，Lost Database Connection ETI のマッピング・ルール自体の構成を示します。



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。

KPI への HI 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。OMi の次の領域で HI を KPI に割り当てます。

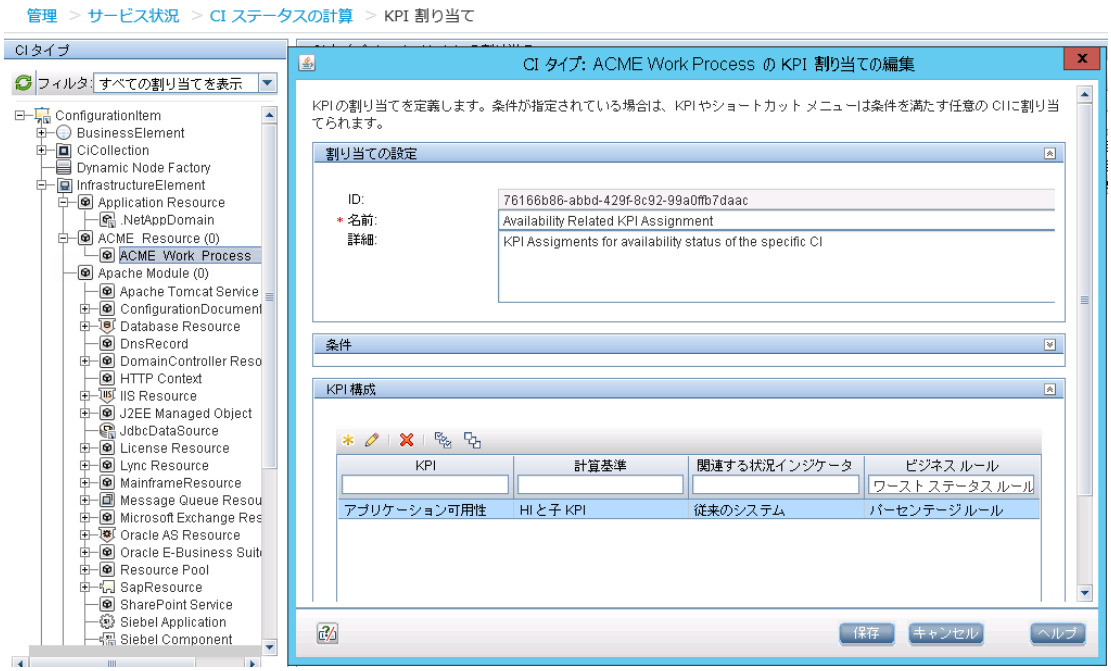
［管理］ > ［サービス状況］ > ［割り当て］ > ［KPI の割り当て］

KPI 割り当ての詳細については，OMi オンライン・ヘルプを参照してください。

「ETI および HI の概要」(41ページ) 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，OMi 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」の項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 メッセージのカスタム・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

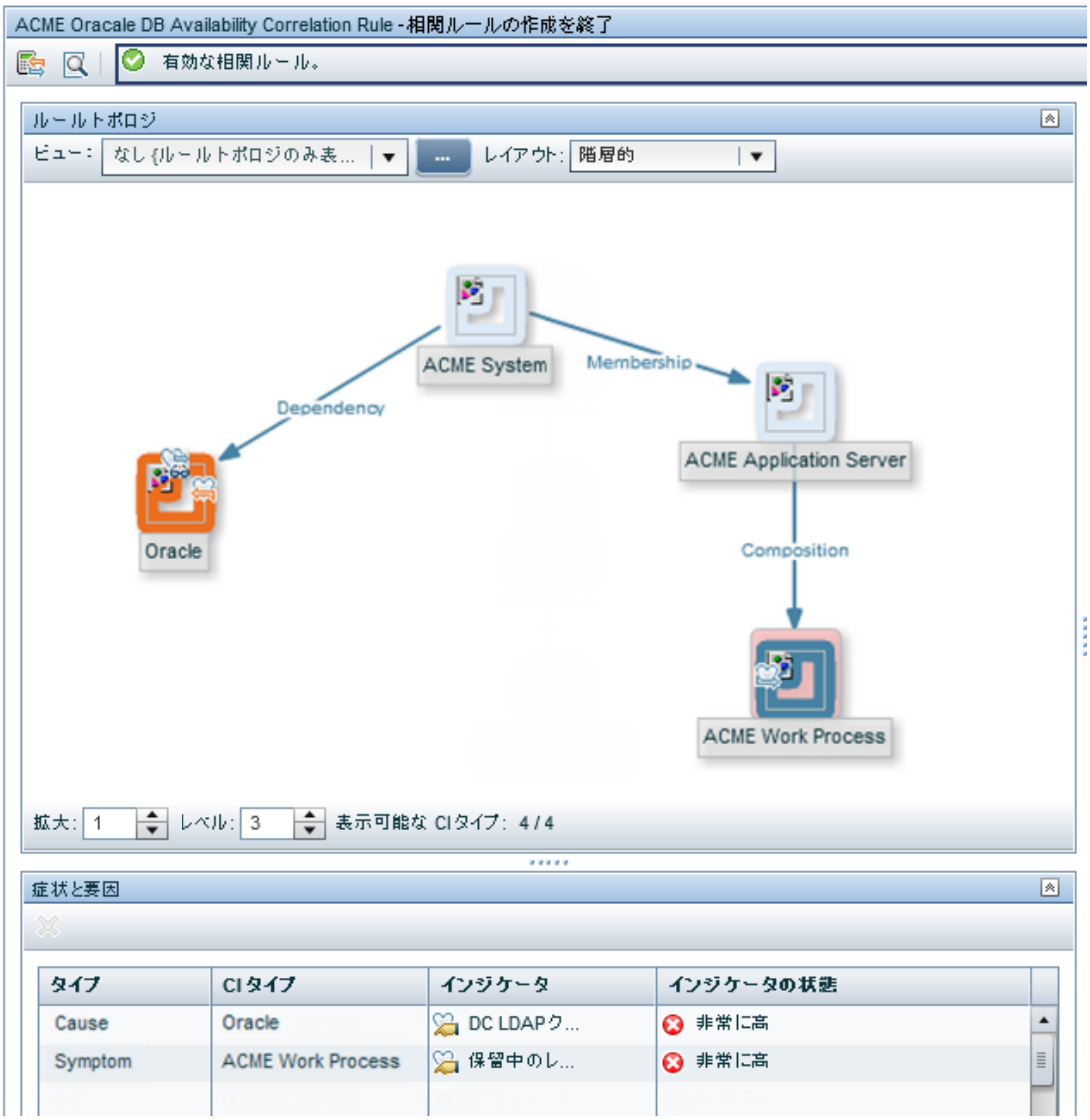
CMA	説明
Description	イベントの原因を説明するテキスト。
NoDuplicateSuppression	既存のイベントと重複するイベントが抑制されない。
Solution	イベントが示す問題の解決に役立つソリューションが記述されたテキスト・フィールド。
SubCategory	イベントが属している論理サブグループ（カテゴリ）の名前（Oracle（データベース），アカウント（セキュリティ），ルータ（ネットワーク）など）。
EtiHint	各 CI の ETI を識別するためのイベント情報。
NodeHint	イベントに関連するノード CI を識別するためのイベント情報。
RelatedCiHint	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。

第5章: 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，OMi オンライン・ヘルプの「イベントの相関処理」の項を参照してください。

第6章: 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。たとえば，変数のプレースホルダ（${node.label} および ${relatedCi.label}）を使用することにより，ノードと CI ラベルのどちらか一方またはその両方を使用してイベントの［タイトル］フィールドと［説明］フィールドを強化することができます。

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。 Groovy スクリプトの開発およびデプロイの詳細については，「Groovy スクリプト」(376ページ)を参照してください。

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，「イベント処理インタフェース」(149ページ)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，「カスタム・アクションのスクリプト」(160ページ)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，「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」(162ページ)を参照してください。

第7章: ツール

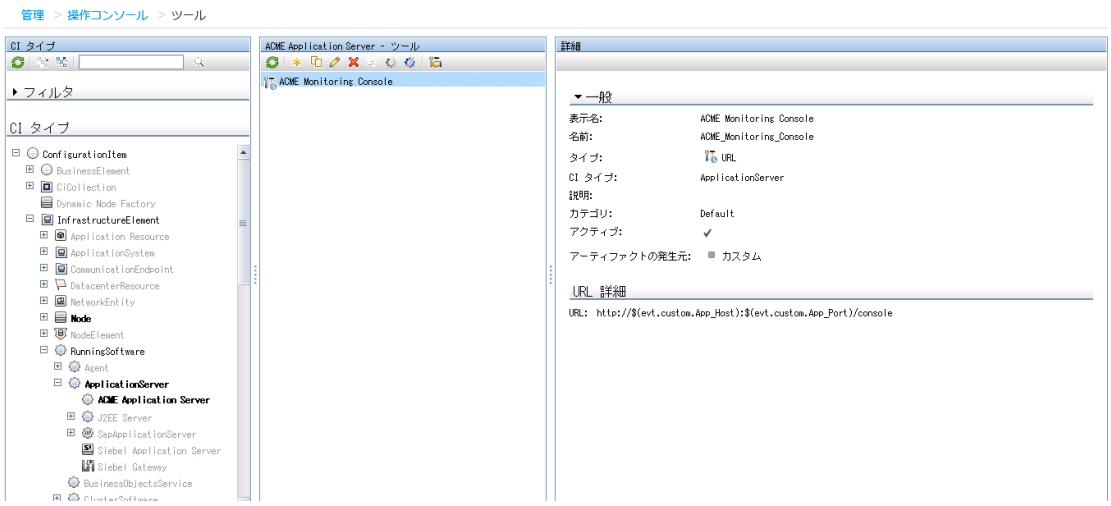
ドメイン・オペレータによる特定のイベントおよびCIの管理と監視や新しいアプリケーション領域で発生した問題の解決に役立つツールを設定できます。

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，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，OMi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。

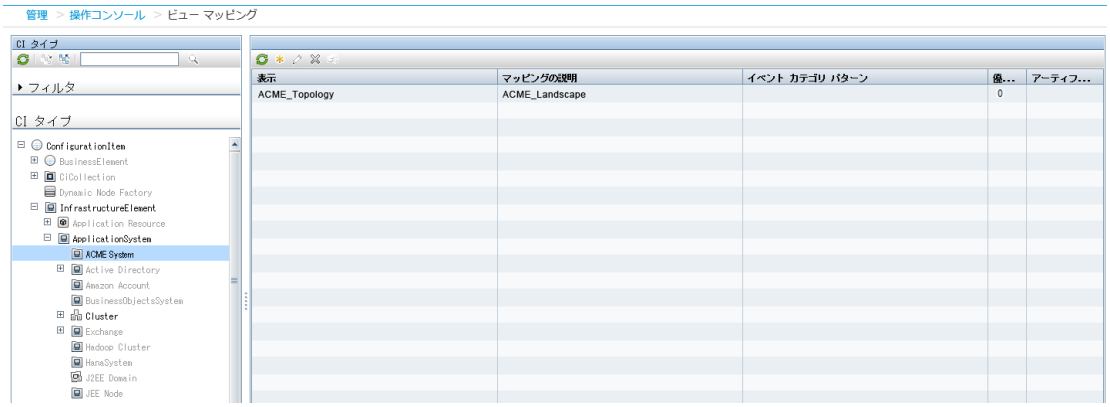


第8章: ビュー・マッピング

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。複数のビューが優先度別に一覧表示され，最も優先度の高いビューが標準設定ビューとして表示されます。

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。

次の図が［ビュー マッピング］表示枠に表示されます。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。



第9章: 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，OMi 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，HP Operations Agent 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 OMi 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。

第10章: コンテンツのパッケージ化

本項では，ある OMi 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。

別の OMi インスタンスで作成したコンテンツを使用する場合は，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 OMi 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 OMi 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。

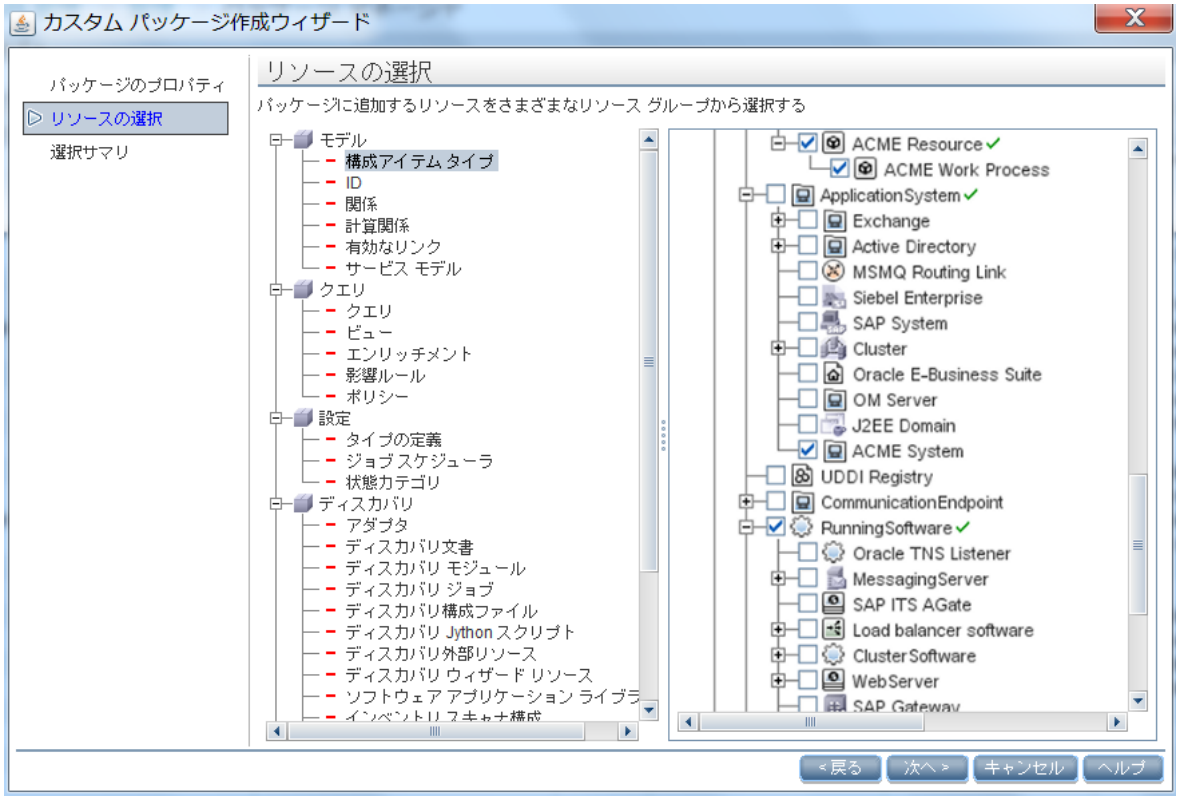
次に，ACME RTSM パッケージを作成するためのワークフローを示します。

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，OMi 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（「トポロジ」(23ページ)を参照）とビュー（「ビュー・マッピング」(54ページ)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。

<*OMi_HOME*>/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，「実行時サービス・モデルへのデータの追加」(62ページ)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，OMi 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，作成することにより，OMi 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 OMi 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 OMi 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。

<*OMi_HOME*>/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III部: 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（実行時サービス・モデル））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

第11章: 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，OMi 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 ディスカバリおよび統合コンテンツ・ガイド』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（https://softwaresupport.hp.com/group/softwaresupport/search-result?keyword=）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，ノード・グループ，ノード・レイアウト・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，Groovy を使用します。 Groovy スクリプトの開発およびデプロイの詳細については，「Groovy スクリプト」(376ページ)を参照してください。

トポロジ同期には次の 2 つの方法があります。

- 「動的トポロジ同期」(64ページ)

  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 バージョンの詳細については，OMi サポート・マトリックス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 データベース内の CI および CI 関係を更新します。

- 「基本的なトポロジ同期」(73ページ)

  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。サポートされている HPOM バージョンの詳細については，OMi サポート・マトリックス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，ノード・グループ，ノード・レイアウト・グループからのトポロジ・データを *1* つの HPOM サーバから OMi 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 OMi 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。

OMi のインスタンスは，HPOM およびほかの OMi 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 OMi のほかのインスタンスに転送するように OMi 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，OMi 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。

サポートされている HPOM バージョンの詳細については，『OMi サポート・マトリックス』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 OMi）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。OMi 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，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，『モデリング・ガイド』を参照してください。

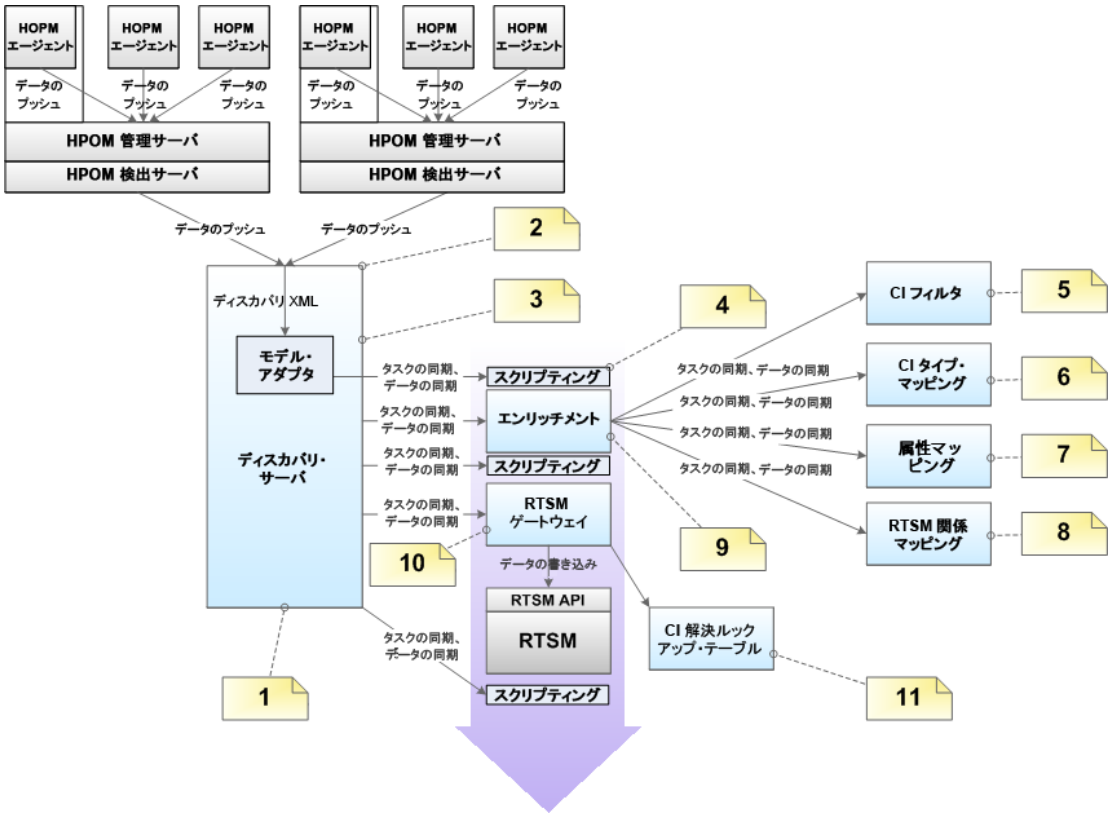
マッピング・ルールに基づき，HPOM トポロジ・データをソースとして使用して RTSM に CI を作成するために，動的トポロジ同期では次の操作が実行されます。

- 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 OMi 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，OMi 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	メモ
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。
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。

参照情報	参照	メモ
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 OMi 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，次の設定手順を実行してください。

1. 接続サーバとして HPOM 管理サーバを追加します。詳細については，OMi 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 管理サーバ間の信頼関係を確立します。詳細については，OMi オンライン・ヘルプを参照してください。
3. *オプション*: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，「同期パッケージの管理」(83ページ)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。

OMi 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。［サーバの構成］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**］をクリックして変更を保存し，［サーバの構成］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. ［**トポロジの同期**］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**］ > ［**インベントリの同期**］ > ［**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。

      ［サーバの構成］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**］をクリックして変更を保存し，［サーバの構成］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. ［**トポロジの同期**］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。

OMi 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 UNIX または Linux 管理サーバで次の手順を実行します。

1. *推奨：* トポロジ同期は，標準設定では，HPOM サービス，ノード，ノード・グループのみを RTSM 内の CI に変換します。［**トポロジ同期のパッケージ**］インフラストラクチャ設定で

nodegroups 同期パッケージを layoutgroups パッケージに置き換えることで，HPOM for UNIX または Linux ノード・レイアウト・グループをマッピングに含めることができます。

詳細については，「標準設定の同期パッケージ」(87ページ)を参照してください。

2. *前提条件 :* HPOM 9.10 for UNIX / 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。
3. *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 11.04 以上であることを確認します
4. *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。
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 -target <*サーバのコンマ区切りリスト*>

   <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。
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

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

基本的なトポロジ同期

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。サポートされている HPOM バージョンの詳細については，OMi サポート・マトリックスを参照してください。

基本トポロジ同期は，OMi 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。

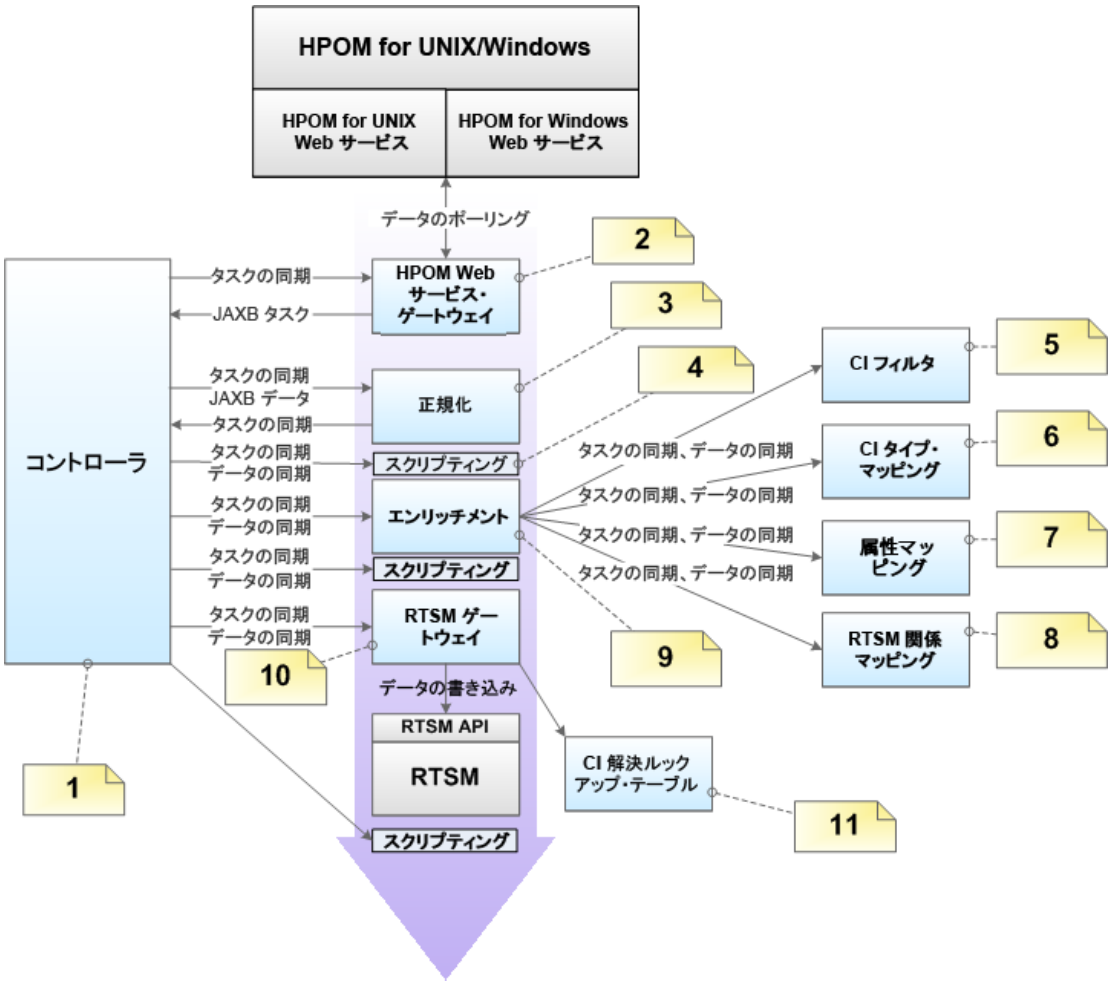
マッピング・ルールに基づき，HPOM ノード，ノード・グループ，および HPOM サービス・モデルをソースとして使用して RTSM に CI を作成する場合，基本トポロジ同期は次の操作を実行します。

- 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 OMi に転送します。
- サービス管理データを OMi 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 にアップロードします。
- データをオンデマンドで RTSM に更新します。
- 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，OMi 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	メモ
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。

参照情報	参照	メモ
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 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，OMi 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。
<*OMi_HOME*>/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 : <*OMi_HOME*>/bin/opr-startTopologySync.bat

Linux : <*OMi_HOME*>/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 : <*OMi_HOME*>/bin/opr-startTopologySync.bat – touch

Linux : <*OMi_HOME*>/bin/opr-startTopologySync.sh – touch

RTSM のエージングの詳細については，『HP Operations Manager i モデリング・ガイド』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 : <*OMi_HOME*>/bin/opr-startTopologySync.bat – skipservices

Linux : <*OMi_HOME*>/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 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，ノード・グループと 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，XML 設定ファイルで構成されます（「マッピング・ファイル」(89ページ)を参照）。これらのファイルによって，HPOM サービス，ノード，ノード・グループが 実行時サービス・モデル 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，ノード・グループ，ノード・レイアウト・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，「標準設定の同期パッケージ」(87ページ)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，「追加の標準設定の同期パッケージ」(88ページ)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，「トポロジ同期の設定 :ACME の例」(91ページ)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 実行時サービス・モデル 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 実行時サービス・モデル にアップロードする必要があります。詳細については，「同期パッケージの管理」(83ページ)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，「同期パッケージ」(85ページ)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。トポロジ同期スクリプトの詳細については，「スクリプティング」(98ページ)を参照してください。 Groovy スクリプトの開発およびデプロイの詳細については，「Groovy スクリプト」(376ページ)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，OMi 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。

［**管理**］>［**セットアップと保守**］>［**インフラストラクチャ設定**］

［**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**］で編集します。

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 OMi 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。

基本的なトポロジ同期

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:

- <*OMi_HOME*>/opr/lib/opr-ts-*.jar
- **Linux :**
  - <*OMi_HOME*>/bin/opr-startTopologySync.sh
  - <*OMi_HOME*>/bin/opr-sdtool.sh

    <*OMi_HOME*>/bin/opr-startTopologySync.sh
- **Windows :**
  - <*OMi_HOME*>\bin\opr-sdtool.bat
  - <*OMi_HOME*>\bin\opr-startTopologySync.bat

ログ・ファイル:

<*OMi_HOME*>/log/opr-topologysync

ログ・ファイル設定ファイル:

<*OMi_HOME*>/conf/core/Tools/log4j/opr-topologysync/opr-topologysync.properties

トポロジ同期パッケージ:

<*OMi_HOME*>/conf/opr/topology-sync/sync-packages

スキーマ・ファイル:

<*OMi_HOME*>/conf/opr/topology-sync/schemas

動的なトポロジ同期

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 :<*OMi_HOME*>/bin/opr-sdtool.bat

Linux :<*OMi_HOME*>/bin/opr-sdtool.sh

<*OMi_HOME*>/opr/lib/opr-ts-*.jar

<*OMi_HOME*>/opr/lib/OvSvcDiscServer.jar

ログ・ファイル:

<*OMi_HOME*>/log/wde/opr-svcdiscserver.log

<OvDataDir>/log/OvSvcDiscServer.log

ログ・ファイル設定ファイル:

<*OMi_HOME*>/conf/core/Tools/log4j/wde/opr-svcdiscserver.properties

トポロジ同期パッケージ:

<*OMi_HOME*>/conf/opr/topology-sync/sync-packages

スキーマ・ファイル:

<*OMi_HOME*>/conf/opr/topology-sync/schemas

トポロジ同期設定

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。

  接続設定には次のようにアクセスできます。

  ［**管理**］>［**セットアップと保守**］>［**インフラストラクチャ設定**］

  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，OMi 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，正しい設定のため，また OMi および HPOM の監視対象となる環境でオブジェクト・トポロジの同期を成功させるために不可欠です。

  HPOM トポロジ同期設定には次のようにアクセスできます。

  ［**管理**］>［**セットアップと保守**］>［**インフラストラクチャ設定**］

  オペレーション管理 - HPOM トポロジ同期設定

同期パッケージの管理

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て有効にしたり，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。

注: コンテンツ・マネージャを使用して，既存の同期パッケージをコンテンツ・マネージャ形式でインポート，エクスポートすることもできます。

- **同期パッケージのアップロードおよび有効化 :** 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。

  OMi は，有効になっている同期パッケージのみを処理します。任意の -enablepackage オプションを使用すると，同期パッケージが［**トポロジ同期のパッケージ**］インフラストラクチャ設定に追加されます。

  ［**管理**］>［**セットアップと保守**］>［**インフラストラクチャ設定**］

  ［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］

  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して有効にするには，次のコマンドを実行します。

  Windows : opr-sdtool.bat – uploadpackage <*path of synchronization package*> [-enablepackage]

  Linux : opr-sdtool.sh – uploadpackage <*path of synchronization package*> [-enablepackage]

- **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，OMi を分散環境にインストールしたときには，各 OMi 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，標準設定でパッケージは <*OMi_HOME*>/conf/opr/topology-sync/sync-packages にダウンロードされます。

- **opr-sdtool ユーザ権限 :** opr-sdtool コマンドライン・インタフェースを実行しているユーザは，ローカル・ユーザ（Windows）または OMi プロセスを実行している環境のユーザ（Linux）である必要があります。SQL Server インスタンスが Windows 認証モードを使用している場合，opr-sdtool を実行しているユーザはデータベース・イベントへのアクセスを許可されている必要があります。

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>

opr-sdtool コマンドライン・インタフェースを実行しているユーザは，ローカル・ユーザ（Windows）または OMi プロセスを実行している環境のユーザ（Linux）である必要があります。SQL Server インスタンスが Windows 認証モードを使用している場合，opr-sdtool を実行しているユーザはデータベース・イベントへのアクセスを許可されている必要があります。

第12章: 同期パッケージ

本項では，HPOM トポロジ・データを RTSM 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 のサービス，ノード，ノード・グループ，ノード・レイアウト・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（「パッケージ記述子ファイル : package.xml」(89ページ)を参照）。

同期パッケージに含めることができるマッピング・ファイルは次のとおりです。

- contextmapping.xml
- typemapping.xml
- attributemapping.xml
- relationmapping.xml

XML 設定ファイルの詳細については，「マッピング・ファイル」(89ページ)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，「マッピング・エンジンと構文」(119ページ)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，「Groovy スクリプト」(98ページ)を参照してください。

標準設定の同期パッケージ

OMi 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 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 自体を作成し，さらに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 自体を作成し，さらに HPOM ノード・グループを RTSM 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。

- layoutgroups

  ホスト CI 自体を作成し，さらに HPOM レイアウト・グループを RTSM の CI タイプ CI Collection（ci_group）CI タイプ CI コレクション（ci_group）にマップし，CI タイプ CI コレクション（ci_group）のインスタンスを作成し，グループ内のノードの関係を作成します。

  > **注:** ci_group は **CI コレクション**として表示されます。

［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。

［**管理**］>［**セットアップと保守**］>［**インフラストラクチャ設定**］

［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。

<*OMi_HOME*>/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。

   ［管理］ > **［セットアップと保守］** > **［インフラストラクチャ設定］**

   ［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。

<*OMi_HOME*>/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「標準設定の同期パッケージ」(87ページ)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 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。

  <*OMi_HOME*>/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 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 では，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

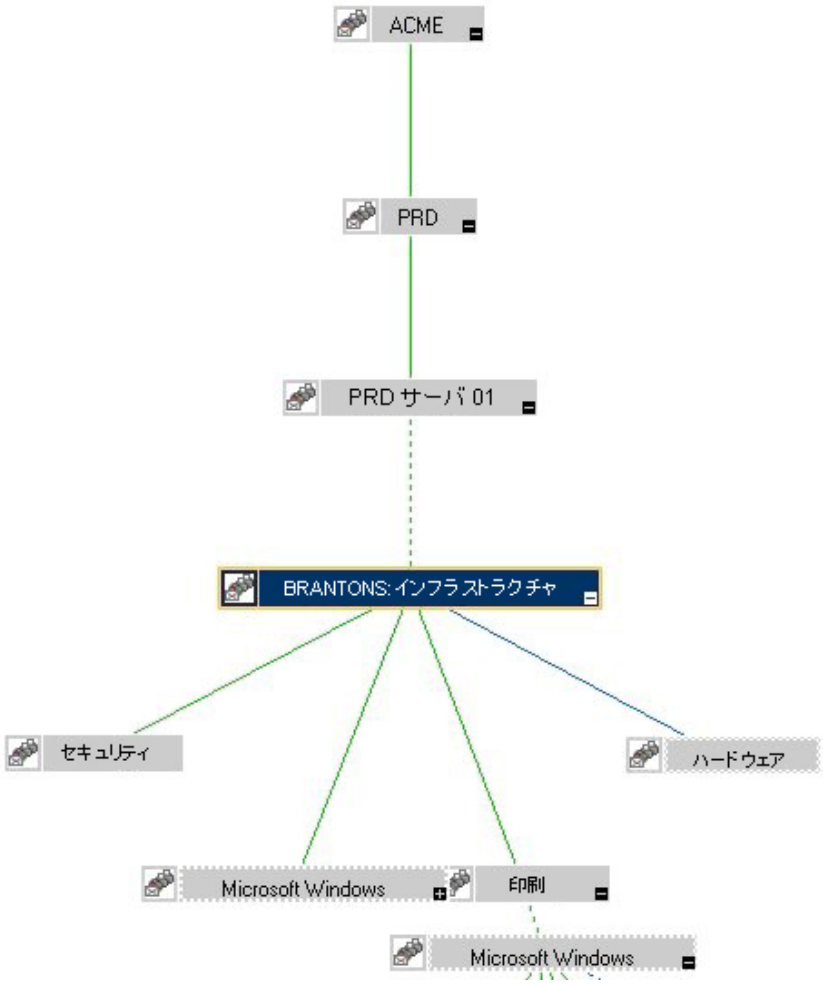
関係マッピング・ファイル relationmapping.xml を使用して，RTSM 内に作成済みの指定された HPOM サービス間の CI 関係を定義できます。

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 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

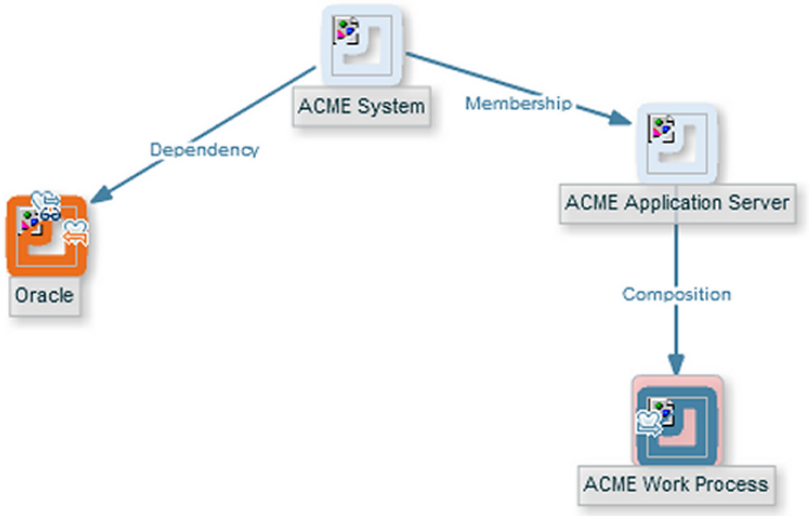
package.xml ファイルでは，トポロジ同期パッケージの名前を定義し，パッケージの説明と優先度を指定します。ACME トポロジ同期パッケージの package.xml ファイルは，次のように記述されています。

```
<パッケージ>
  <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）ファイルの設定 :contextmapping.xml

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 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，RTSM 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 -->
 <ルール>
  <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・ファイルの設定 :typemapping.xml

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 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・ファイルの設定 :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 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・ファイルの設定 :relationmapping.xml

関係マッピング・ファイル relationmapping.xml では，RTSM 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>
          <開始>
           <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 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，「スクリプティング」(98ページ)を参照してください。Groovy スクリプトの開発およびデプロイの詳細については，「Groovy スクリプト」(376ページ)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。

<*OMi_HOME*>/conf/opr/topology-sync/sync-packages/<*SyncPackageName*>

第13章: 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，「同期パッケージの場所」(97ページ)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 スクリプトの開発およびデプロイの詳細については，「Groovy スクリプト」(376ページ)を参照してください。

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［Groovy スクリプト使用の有効化］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。

［管理］>**［セットアップと保守］**>**［インフラストラクチャ設定］**

［オペレーション管理 - HPOM トポロジ同期］ > ［Groovy スクリプト使用の有効化］

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。

<*OMi_HOME*>/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。

<*OMi_HOME*>/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。

エラー処理

スクリプト内でエラーが発生すると，例外が生成されます。エラーの処理は各スクリプトを呼び出す前後に行います。標準設定では，スクリプト内で例外が発生すると，同期が中止されます。この動作は次のコマンドを呼び出すことで変更できます。

scriptInterface.setAbortSyncOnError(boolean)

false に設定すると，メソッド abortSync("...") を使用してスクリプトを強制的に失敗にできます。たとえば，スクリプトで条件を確認します。強制的に失敗になったため，同期を完了できません。

次の表に，同期ステータス（同期の成功または失敗）およびスクリプトの動作の関係を示します。

ステータス	スクリプトの動作
同期成功	エラーが発生せず，スクリプト内で同期が強制的に中断されることなく，スクリプトが完了した。 例外がスローされ，AbortSyncOnError が false に設定されていたとしても，エラーが発生することなくスクリプトが完了した。
同期失敗	abortSync(String) コマンドを使用したスクリプトの条件が原因で，スクリプトの実行によって例外やスクリプトの強制的な失敗が発生した。

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 ();

// すべての HPOM サービス，ホスト，ノード・グループを取得する
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 およびクラスタ・パッケージとの関係を作成する
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 アドレス属性について 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");

    }
}
```

第14章: 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。

XSD XML 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 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。

<*OMi_HOME*>/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

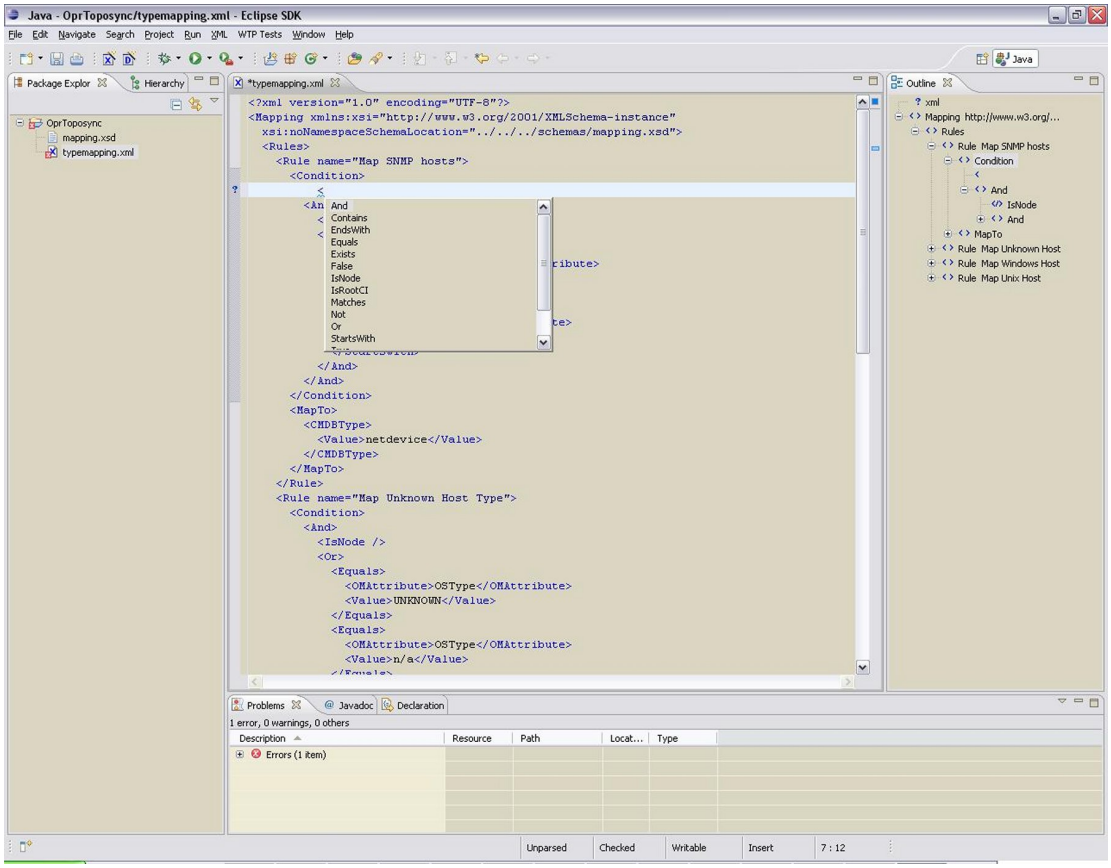
最新の XML エディタでは，ファイルをスキーマに基づいて検証できます。たとえば，Eclipse では，XML ドキュメントの最上位の要素が XSD ファイルの参照である場合，スキーマに基づいて XML ファイルを検証できます。検証を有効にするには，XML ファイルの最上位の要素に次の属性を追加します。

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="<OMi_HOME>/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。

   ［**管理**］>［**セットアップと保守**］>［**インフラストラクチャ設定**］

   ［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 : <OMi_HOME>/bin/opr-startTopologySync.bat

   Linux : <OMi_HOME>/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。

<*OMi_HOME*>/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 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 式を検証できます。

   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，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，「XML 設定ファイルの検証」(104ページ)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。

   <*OMi_HOME*>/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。

<*OMi_HOME*>/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「トポロジ同期ファイルの場所」(80ページ)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 OMi 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，『OMi 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 for Windows 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。

症状	要因	ソリューション
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（「同期パッケージの管理」(83ページ)を参照）。
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 OMi 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 OMi 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が，ログ・ファイルで不可解な RTSM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，*<OMi_HOME>*/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。

制限事項

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 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 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 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 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 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 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 OMi に到着しないことがあります。この場合，RTSM に反映されません。その結果，HPOM 内のステータスと RTSM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。

第15章: 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 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。

- 「共通マッピング・ファイル形式」(119ページ)
- 「マッピング・ファイルの構文」(120ページ)
- 「XPath ナビゲーション」(142ページ)

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，「マッピング・ファイルの構文」(120ページ)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。

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，「フィルタリング」(134ページ)を参照してください。

ルール条件

<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。

例 :

```
<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。

例 :

```
<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://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html

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

HPOM サービス ID

<OMId/>

戻り値の型 :文字列

OMi に保存されている CI の HPOM ID 文字列を返します。HPOM ID は，次のように異なる値を返します。

サービス :HPOM ID はサービス ID

ノード :HPOM ID は一意の ID

ノード・グループ :HPOM ID はノード・グループ ID

HPOM タイプ

<OMType/>

戻り値の型 :文字列

OMi に保存されている HPOM タイプを返します。HPOM サービスの場合，HPOM タイプはサービス・タイプ定義です。ノードの場合，HPOM 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 :文字列

RTSM 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 :文字列

RTSM または OMi の CI のキャプション文字列を返します。

HPOM 属性

<OMAttribute>[Name]</OMAttribute>

戻り値の型 :文字列

指定名の HPOM 属性の値を返します。

CMDB 属性

<CMDBAttribute>[Name]</CMDBAttribute>

戻り値の型 :文字列

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"]>
  <含む>
    <!-- 1st.Operand -->
  </In>
  <For>
    <!-- 2nd.Operand -->
  </For>
  <By>
    <!-- 3rd.Operand -->
  </By>
</Replace>
```

戻り値の型 :文字列

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。

```
<置換>
  <含む>
    <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://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html

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">
  <含む>
    <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 :文字列

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，「XPath ナビゲーション」(142ページ)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，「XPath ナビゲーション」(142ページ)を参照してください。

値

<Value>[String]</Value>

戻り値の型 :文字列

定数値を返します。

リスト

```
<リスト>
  <!-- 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

```
<開始>
  <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 :

```
<開始>
  <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 :文字列

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。

マッピングの例

HPOM サービスの属性 Caption を RTSM 内の CI 属性 display_label にマップします。

例 :

```
<MapTo>
  <CMDBAttribute>
    <Name>display_label</Name>
    <SetValue>
      <Caption/>
    </SetValue>
  </CMDBAttribute>
</MapTo>
```

HPOM サービスの属性 OMID を RTSM 内の CI 属性 data_name にマップします。

例 :

```
<MapTo>
  <CMDBAttribute>
    <Name>data_name</Name>
    <SetValue>
      <OMId/>
    </SetValue>
  </CMDBAttribute>
</MapTo>
```

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>

次の例において，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。

例：

```
<Mapping xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="../mapping.xsd">
 <ルール>
  <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>
```

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

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。

```
<CMDBType>
  [Operand]
  ...
</CMDBType>
```

次の例では，OM タイプが Exch2k7_ByServer で，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。

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>
```

次の例では，属性 OSType が文字列 Windows で始まるすべてのノードが CMDB タイプ nt にマップされます。

例：

```
<Mapping>
  <ルール>
   <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>
```

属性マッピング

属性マッピング・ファイル attributemapping.xml では，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。

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。

文字列値へのマッピング

```
<CMDBAttribute>
  <Name>[Attribute Name]</Name>
  <SetValue>
    [Operands]
  </SetValue>
</CMDBAttribute>
```

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>
```

整数値へのマッピング

```
<CMDBAttribute>
  <Name>[Attribute Name]</Name>
  <SetIntValue>
    [Operands]
  </SetIntValue>
</CMDBAttribute>
```

ブール値へのマッピング

```
<CMDBAttribute>
  <Name>[Attribute Name]</Name>
  <SetBoolValue>
    [Operands]
  </SetBoolValue>
</CMDBAttribute>
```

ロング値へのマッピング

```
<CMDBAttribute>
  <Name>[Attribute Name]</Name>
  <SetLongValue>
    [Operands]
  </SetLongValue>
</CMDBAttribute>
```

日付値へのマッピング

```
<CMDBAttribute>
  <Name>[Attribute Name]</Name>
  <SetdateValue>
    [Operands]
  </SetdateValue>
</CMDBAttribute>
```

浮動小数値へのマッピング

```
<CMDBAttribute>
  <Name>[Attribute Name]</Name>
  <SetFloatValue>
    [Operands]
  </SetFloatValue>
</CMDBAttribute>
```

バイト値へのマッピング

```
<CMDBAttribute>
  <Name>[Attribute Name]</Name>
  <SetByteValue>
    [Operands]
  </SetByteValue>
</CMDBAttribute>
```

倍精度値へのマッピング

```
<CMDBAttribute>
  <Name>[Attribute Name]</Name>
  <SetDoubleValue>
    [Operands]
  </SetDoubleValue>
```

```
</CMDBAttribute>
```

StringList 値へのマッピング

```
<CMDBAttribute>
  <Name>[Attribute Name]</Name>
  <SetStringListValue>
    [Operands] (  カンマ区切り値)
  </SetStringListValue>
</CMDBAttribute>
```

IntList 値へのマッピング

```
<CMDBAttribute>
  <Name>[Attribute Name]</Name>
  <SetIntListValue>
    [Operands]   カンマ区切り値
  </SetIntListValue>
</CMDBAttribute>
```

属性マッピングの例

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。

```
<Mapping xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="../../../schemas/mapping.xsd">
 <ルール>
  <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>
```

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>
 <終了>
  [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>
 <開始>
   [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。

次の例では，STD Exch2k7_ByServer の CI により，その CI がホストされているノードとの is_impacted_from タイプの関係，および Exch2k7_Role_ で始まる OM タイプのすべての子孫 CI との deployed 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。

例 :

```
<Mapping>
 <Rules Context="exchange">
  <Rule name="Create relation server to node">
   <Condition>
    <Equals>
     <OMType/>
     <Value>Exch2k7_ByServer</Value>
    </Equals>
```

```
    </Condition>
    <MapTo>
     <RelationTo>
      <終了>
       <DependencyCI relationType="hosted_on">
         <True/>
       </DependencyCI>
      </To>
      <Type>is_impacted_from</Type>
     </RelationTo>
     <RelationTo>
      <終了>
       <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>
```

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 クエリ言語を理解するには，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，「XPath ナビゲート・データ構造の例」(145ページ)に記載されています。

CI データ構造

- OMAttributes

  元の RTSM 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。

- タイプ

  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

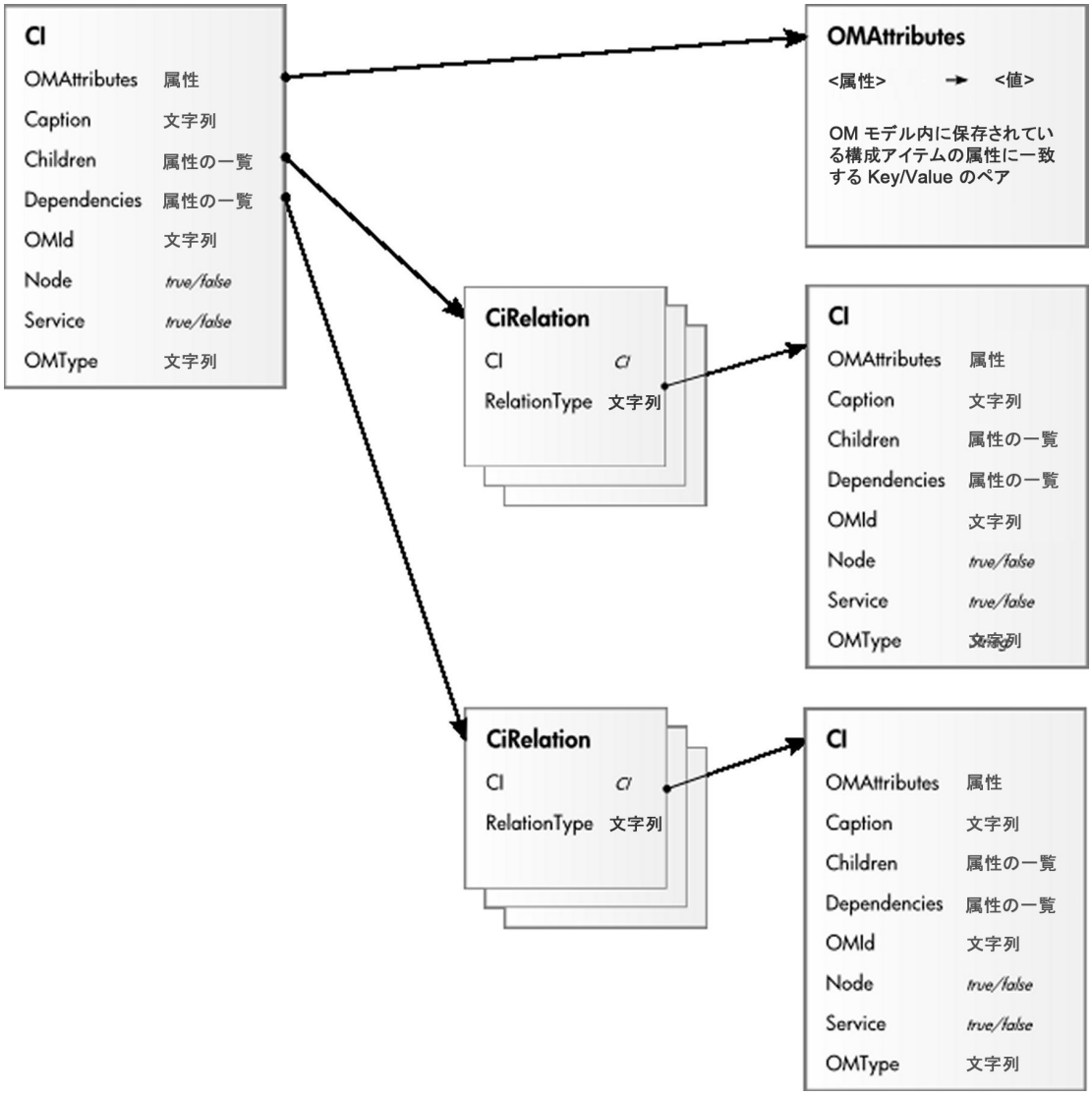
  RTSM に保存されている関係タイプ。

  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。

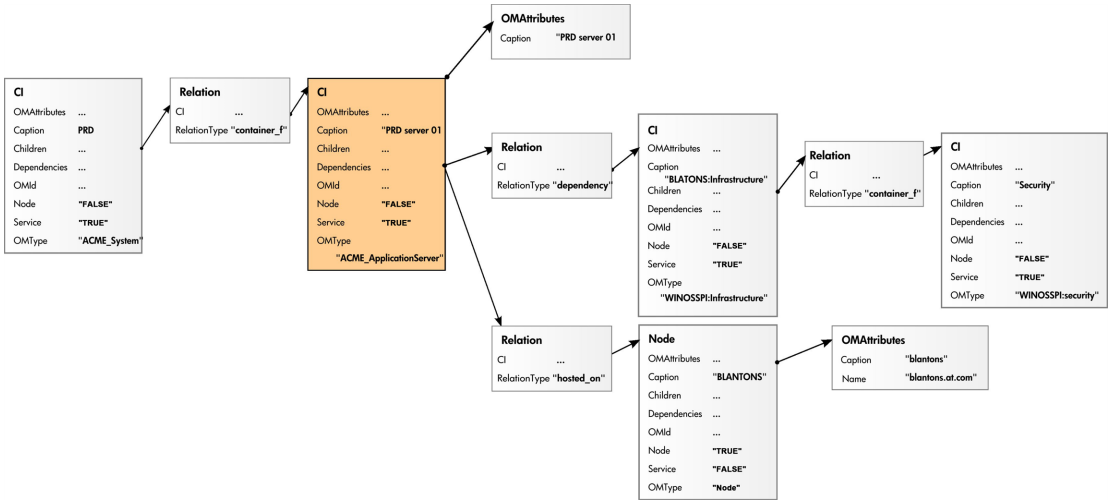
次の図に，ナビゲーションに表示されるデータ構造を示します。



XPath ナビゲート・データ構造の例

XPath ナビゲート・データ構造の例については，次の図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）。

次の図は，いくつかの XPath の例を示しています。



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
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IV部: 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

第16章: イベント処理インタフェース

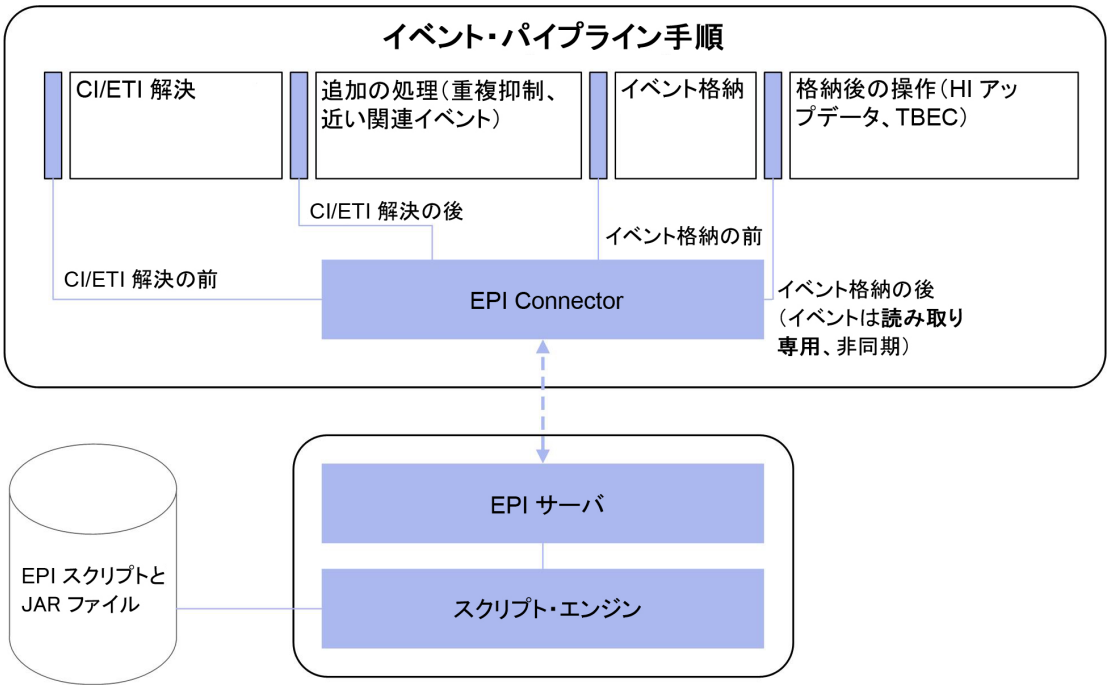
本項では，イベント処理インタフェース（EPI）の概要を示し，イベント・パイプラインのさまざまな段階と，イベントを変更および強化するスクリプトを実行するための適切なエントリ・ポイントについて説明します。

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。 Groovy スクリプトの開発およびデプロイの詳細については，「Groovy スクリプト」(376ページ)を参照してください。

「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」(149ページ)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 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。

- **イベントがデータベースに保存される前 :** 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。

- **イベントがデータベースに保存された後 :** 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］ > ［イベント処理］ > ［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，「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」(162ページ)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，OMi 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。

イベント・エンリッチメントの EPI スクリプト

解決済み CI の RTSM 内に存在する CI 関連データにアクセスし，この情報を使用してイベントを強化することができます。このアプローチの基盤は，EPI Groovy スクリプトと RTSM JAVA API の組み合わせです。実際の例については，「サンプル・スクリプト」(157ページ)を参照してください。

イベント・エンリッチメントで使用可能なデータは，次のパスで表示されるリストに含まれている任意の属性です。

Administration > RTSM Administration > Modeling > CI Type Manager

CI タイプを選択して，［属性］タブを開きます。

詳細

基本的なスクリプト設計

JAVA API を介しての RTSM へのアクセスにおいては，最初に接続を確立することが（有効なユーザの資格情報を入力を含む），要件の一つとして挙げられます。接続の確立にはオーバーヘッドの処理が関係しており，処理中のイベントごとに新規接続を開いて閉じることは実際的ではありません。EPI スクリプトの場合は，処理中の各イベントの処理時間を最小限にする必要があります。

処理時間を最小限にするために，イベント CA の作成に使用される CI 属性の対応する値を含む文字列にリンクされている CI ObjectID が含まれる，単純なスクリプト内部の HashMap を事前読み込みすることができます。事前読み込みされる HashMap はメモリ内に存在し，RTSM 情報の小さなサブセットを高速で取得できるようになります。

init() メソッドは，RTSM への接続を確立するために使用します。別個のスレッドが開始され，そこで HashMap が初期化されます。このスレッドは，（同期オブジェクトを介して）メイン・スレッドと

も同期されるため，HashMap を一時的にブロックし，指定した待ち時間（10 分など）が経過してから再読み込みすることができます。

process() メソッドは，次の処理で使用します。

- イベントの関連 CI の ObjectID を取得する
- HashMap にアクセスして「所有者」情報を取得する
- event.AddCustomAttribute を呼び出して CA を作成する

destroy() メソッド（スクリプトがアンロードまたは非アクティブ化されたときに呼び出されます）は，HashMap によって使用されているメモリを解放するために使用します。

HashMap の作成

RTSM から CI のセットを取得するため，HP UCMDB API を使用して TQL クエリが実行されます。使用可能な API の詳細については，「HP UCMDB API Reference」を参照してください。これらのファイルは次のフォルダにあります。

<OMi GW Server install directory>/AppServer/webapps/site.war/amdocs/eng/doc_lib/API_docs/UCMDB_JavaAPI

スクリプトは，「名前付きクエリ」（RTSM ですでに定義されているクエリ）の実行方法を例示しています。

クエリを介して CI を取得したら，HashMap を初期化する必要があります。クエリ結果に含まれる CI ごとに，ObjectID と（ci.getPropertyValue メソッドによって取得される）その属性の対応する文字列値が読み込まれます。任意の文字列属性を使用できます。

> **注:** クエリにより，RTSM で定義されている（および理想的には値が入力されている）必要な属性が実際に指定されている CI が返されます。

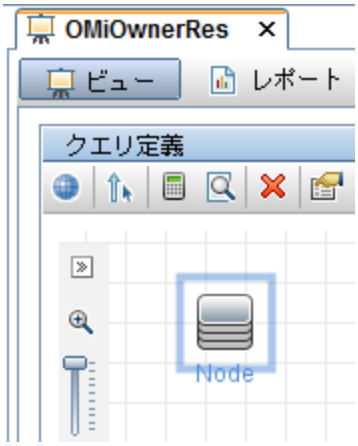
バックグラウンド・スレッドは，メイン・スレッドと同期されていないコードの一部に含まれるパラレル・マップ（newMap）を読み込みます。スレッドは，ロックされる（同期される）と，一時的 HashMap から「実行環境」HashMap（ownerMap）を初期化します。これは，RTSM トポロジにおける継続的な変更を反映するように HashMap を最新の状態で維持するための安全で有効なアプローチです。

OwnerResolver スクリプトの使用方法

次の手順は，CI が RTSM からの CI 属性によって解決された後，イベントの処理中にイベントを許可するスクリプトの使用方法を示しています。

1. 対象の CI を返すモデリング・スタジオで TQL クエリを定義します。

   ［管理］ > **［RTSM 管理］** > **［モデリング］** > **［モデリング スタジオ］**

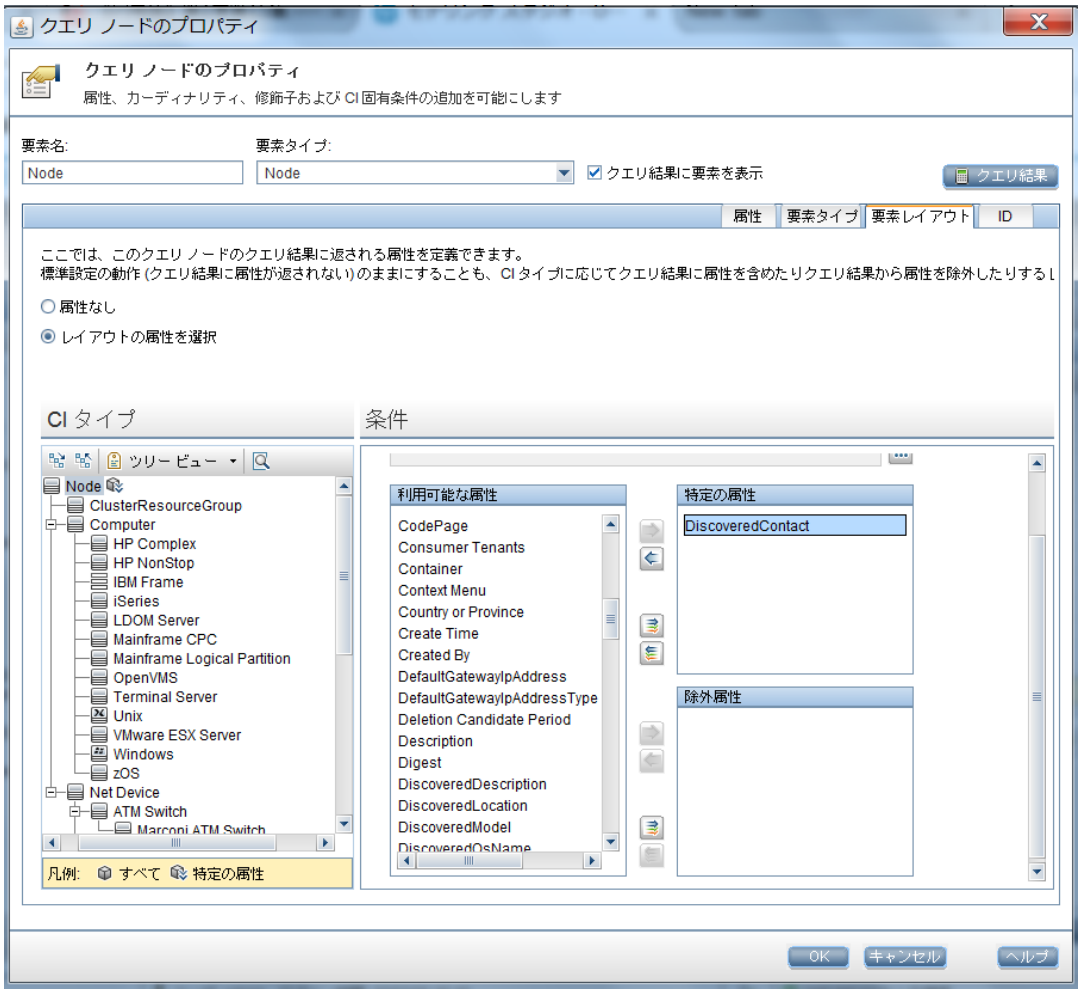


この例において，OMiOwnerRes クエリは，DiscoveredContact 属性の値が NOT null になっている Node タイプの CI を返します。

> **注:** DiscoveredContact は，Node CI タイプの標準属性です。ノード CI ごとに DiscoveredContact 属性に手動で値を追加し，サンプル・スクリプトがイベント・ブラウザの［連絡先］列に表示される連絡先名を返すようにする必要があります。

Node CI タイプ条件は，次のようにして［クエリ ノードのプロパティ］ダイアログ・ボックスで設定します。

2. RTSM JAVA API で DiscoveredContact 属性を使用できるようにするには，［要素のレイアウト］で，計算に使用するようマークする必要があります。



3. **CI/ETI 解決後**パイプライン手順で，スクリプトを（この場合は，適切なホスト名，資格情報，クエリ名，CI 属性名を指定して）インストールします。

> **注:** 特別なクラスパスは不要です。

4. ［利用可能なカスタム属性］のオペレーション管理の［インフラストラクチャ設定］で CA を設定することにより，ブラウザで使用する CONTACT CA を公開します。

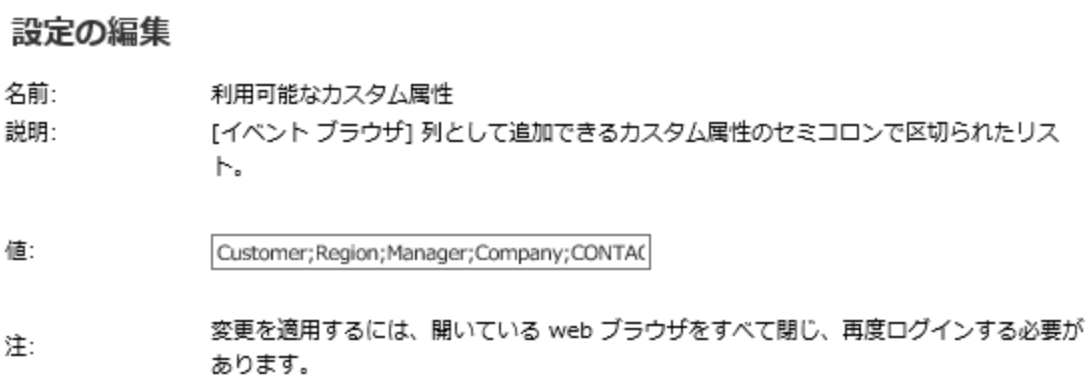
   a. インフラストラクチャ設定を開きます。

      ［**管理**］>［**セットアップと保守**］>［**インフラストラクチャ設定**］

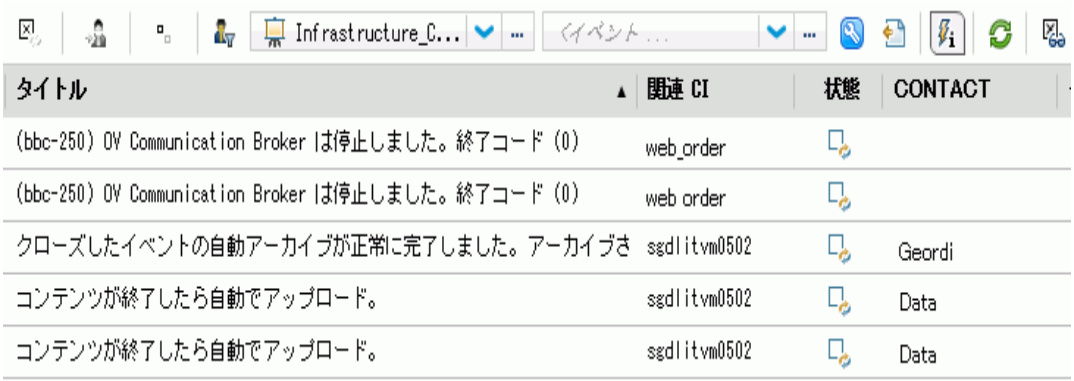
   b. ［**アプリケーション**］を選択し，リストを使用して管理コンテキストを［**オペレーション管理**］に設定します。

c. ［カスタム属性設定］表示枠で，［利用可能なカスタム属性］項目を開きます。

d. CONTACT を新規値として追加します。



5. ブラウザの列を変更して新規 CA（CONTACT）を表示します。



RTSM からの DiscoveredContact 属性は，［連絡先］列に値を入力するために使用されます。

［連絡先］列の一部のフィールドは，関連する CI タイプが Node（または Node の子 CI タイプ）ではないため，空になっています。その結果，この CI タイプは HashMap の作成に使用される TQL に含められず，HashMap 内には値がなくなります。

サンプル・スクリプト

RTSM からの CI 属性によって CI が解決された後のイベント処理中にイベントを強化するには，EPI Groovy スクリプトで次の定義を使用できます。

```
def ucmdbServiceProvider
UcmdbService ucmdbServiceAccess = null
```

次の OwnerResolver スクリプト・テンプレートは，定義の使用方法を示しています。

```
import com.hp.ucmdb.api.UcmdbService
import com.hp.ucmdb.api.UcmdbServiceProvider
import com.hp.ucmdb.api.topology.Topology
import com.hp.ucmdb.api.topology.TopologyQueryService
import com.hp.ucmdb.api.types.TopologyCI
import org.apache.commons.logging.Log
import org.apache.commons.logging.LogFactory

class OwnerResolver {
  private static Log s_log = LogFactory.getLog(OwnerResolver.class.canonicalName)

  def ucmdbService
  final String QUERY_NAME = "OMiOwnerRes"

  HashMap<String, String> ownerMap = new HashMap<String, String>()
  boolean running = true
  UcmdbServiceProvider provider = null
  UcmdbService service = null
  final Object syncObject = new Object()

  void init() {《》

    service = ucmdbService.getCmdbService()《》

    Thread.start {
      while (running) {
        TopologyQueryService tqs = service.getTopologyQueryService()
        Topology topology = tqs.executeNamedQuery(QUERY_NAME)
        HashMap<String, String> newMap = new HashMap<String, String>()
        topology.getAllCIs().each { TopologyCI ci ->
          final String ciId = ci.id.getAsString()
          final String owner = ci.getPropertyValue("discovered_contact")
          if (owner)
            newMap.put(ciId, owner)
          s_log.debug("CI with ID=${ciId} owned by ${owner}")
        }
        synchronized (syncObject) {
          ownerMap = newMap
          s_log.info("Owner map initialized with " + ownerMap.size() + " entries.")
          try {
            syncObject.wait(600000)
          }
          catch (InterruptedException ignore) {
            // 無視する
          }
        }
      }
    }
  }

  void destroy() {
    ownerMap = null
    running = false
    synchronized (syncObject) {
      syncObject.notifyAll()
    }
  }

  void process(eventList) {
    synchronized (syncObject) {
      eventList.each { event ->
        def ciId = event.getRelatedCiId()
        if (!ciId) // Check for empty CI Id
```

```
        s_log.warn("Related CI ID is NULL or empty")
       else {
        s_log.info("Related CI ID = " + ciId)
        String owner = ownerMap.get(ciId)
        if (owner) {
         s_log.info("CI Owner = " + owner)
         event.addCustomAttribute('CONTACT', owner)
        } else {
         s_log.info("Owner not found for CI with id=" + ciId + ".所有者
         map has " + ownerMap.size() + " entries.")
        }
       }
      }
     }
    }

}
```

ヒントと制限事項

RTSM データの取得でスクリプトを使用する場合は，次のヒントを参考にしてください。

- クエリで条件を使用して，返される CI を，使用する属性に何らかの値が指定されている CI に限定します。これにより，対応する属性値が存在しない CI の読み込みを回避します。
- クエリによって返される CI の数に留意します。スクリプト内でクエリを使用する前に，RTSM 管理で**クエリ結果数の計算**を実行します。返される CI が多くなるほど，HashMap のために必要とされるメモリ領域も増えます。
- スクリプト名は，スクリプトで指定したクラス名と同じになっている必要があります。スクリプト名は，次の場所に保存されます。

  ［管理］ > ［イベント処理］ > ［自動化］ > ［イベント処理のカスタマイズ］

- 表示名（getPropertyValue メソッド）ではなく，スクリプトでの実際の CI 属性名を使用します。
- 各イベントでスクリプトの実行に要した時間は，次のログ・ファイルで確認できます。

  <*OMi-HOME*>/log/opr-epi-server/opr-epi-server.log

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，「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」(162ページ)を参照してください。

第17章: 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。 Groovy スクリプトの開発およびデプロイの詳細については，「Groovy スクリプト」(376ページ)を参照してください。

カスタム・アクション・スクリプトの指定

注: カスタム・アクションを指定，実行するには，OMi の［ユーザ管理］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，OMi 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・マネージャでは，イベントの発生時にカスタム・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］ > ［イベント処理］ > ［自動化］ > ［イベント処理のカスタマイズ］

［管理］ > ［操作コンソール］ > ［カスタム アクション］

OMi 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，「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」(162ページ)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，OMi 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，「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」(162ページ)を参照してください。

第18章: 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については，製品に含まれる Java API のマニュアルに記載されています。 Groovy スクリプトの開発およびデプロイの詳細については，「Groovy スクリプト」(376ページ)を参照してください。

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。

<*OMi_HOME*>/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。

<*OMi_HOME*>/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);

    def groupId = event.getAssignedGroupId();
    event.setAssignedGroupId(groupId);

    def assignedUserId = event.getAssignedUserId();
    event.setAssignedUserId(assignedUserId);

    Action autoAction = createSampleAction();
    event.setAutoAction(autoAction);

    String assignedGroupName = event.getAssignedGroupName();
    event.addCustomAttribute( “ASSIGNED_GROUP_NAME” , assignedGroupName);

    String assignedUserLogin = event.getAssignedUserLogin();
    event.addCustomAttribute("ASSIGNED_USER_LOGIN", assignedGroupName);

    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 etiDisplayName = event.getEtiDisplayName();
    event.addCustomAttribute("ETI_DISPLAY_NAME", etiDisplayName);

    String etiName = event.getEtiName ();
    event.addCustomAttribute ("ETI_NAME", etiName);

    String etiStateDisplayName = event.getEtiStateDisplayName ();
    event.addCustomAttribute("ETI_STATE_DISPLAY_NAME", etiStateDisplayName);

    String etiStateName = event.getEtiStateName();
    event.addCustomAttribute("ETI_STATE_NAME", etiStateName);
```

```
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);

    /*
    * isReceivedOnCiDowntime の呼び出しは，CI/ETI 解決の後に実行する。
    * CI はその時点で指定される。
    */

        Boolean receivedDuringDowntime = isReceivedOnCiDowntime();
        event.addCustomAttribute(“RECEIVED_ON_DOWNTIME”, receivedDuringDowntime);

    }

    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。

   <*OMi_HOME*>/lib/jtds-1.0.jar

```
import groovy.sql.Sql;

class ResolveLocationFromDB
{
 def connection;
 void init()
 {
  def properties = new Properties();

  // プロパティ・オブジェクトを作成する properties.put("user", "sa"),
  // 接続のユーザ ID を設定する properties.put("password", "installed"),
  // 接続のパスワードを設定する def d = new net.sourceforge.jtds.jdbc.Driver (),

  def conn = d.connect("jdbc:jtds:sqlserver://localhost/asset_db", properties);

  connection = Sql.newInstance(conn);
 }

 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();
 }
}
```

カスタム・アクションのサンプル Groovy スクリプト

本項では，カスタム・アクションのサンプル Groovy スクリプトを示します。

カスタム・アクションのサンプル Groovy スクリプトは，次のディレクトリにあります。

<*OMi_HOME*>/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;
    }
}
```

第19章: EPI の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。

<*OMi_HOME*>/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。

   <*OMi_HOME*>/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V部: OMi 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 OMi 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。

本項の内容

第20章: 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，OMi 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適しています。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を使用している場合は，統合によって，OMi UI にドリルダウンできるようになります。たとえば，オペレータは，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・アプリケーションを持つことが可能です。

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。

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 OMi 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://<hostname:port>/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 「イベント・ブラウザの URL 起動用パラメータ」(175ページ)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://<hostname:port>/opr-console/opr-evt-browser?<set_parameters_here>

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

「イベント・ブラウザの URL 起動用パラメータ」(175ページ)には，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

イベント・ブラウザの URL 起動用パラメータ (続き)

パラメータ	説明	可能な値	標準設定値
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
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>（「表示するカラムを定義する固定カラム・キー」(178ページ)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

イベント・ブラウザの URL 起動用パラメータ (続き)

パラメータ	説明	可能な値	標準設定値
readOnly	イベント・ブラウザを読み取り専用モードで起動します。 読み取り専用モードでは，イベントでユーザがアクションを実行できないよう，ほとんどのアイコン，ボタン，メニューが非表示になります。ユーザは，イベント詳細の表示 / 非表示，ブラウザのリフレッシュ，イベントの検索，OMi オンライン・ヘルプ の表示ができます。	次のうちいずれか: • true • false	false
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
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>（「表示するカラムを定義する固定カラム・キー」(178ページ)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，「表示するカラムを定義する固定カラム・キー」(178ページ)で定義されています。「カスタム属性カラム・キー」(179ページ)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，OMi 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
eventAge	作成日時に基づくイベントの有効期間
eti	イベント・タイプ・インジケータ値
externalId	外部イベント ID
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

表示するカラムを定義する固定カラム・キー (続き)

カラム・キー	説明
relatedCi	関連 CI
relatedCiHint	関連 CI のヒント
sendingServer	送信サーバ
severity	重要度
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，［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，OMi 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

「イベント・ブラウザの URL 起動用フィルタ・パラメータ」(180ページ)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

イベント・ブラウザの URL 起動用フィルタ・パラメータ (続き)

フィルタ・パラメータ	説明	値
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「文字列属性に可能なフィルタ・タイプと値」(181 ページ)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

文字列属性に可能なフィルタ・タイプと値 (続き)

	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 OMi）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://<hostname:port>/opr-console/opr-evt-details?eventId=<id_of_the_event>

説明 :

<id_of_the_event> は，［イベント詳細］に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VI部: 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，「イベントの転送およびイベント変更の同期」(245ページ)を参照してください。

本項の内容

第21章: 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 OMi 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，「イベント・ブラウザの URL 起動」(174ページ)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。

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://<*server.example.com*>/opr-console/rest/

- セキュア保護された環境:

  https://<*server.example.com*>/opr-console/rest/

説明:

<*server.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，「サービス・ドキュメントの OPR イベント・サービスのリスト」(188ページ)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・サービス:

  http://<*server.example.com*>/opr-console/rest/<*version*>/event_list

  <*version*> は，OMi 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。

- イベント変更サービス:

  http://<*server.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。「サービス・ドキュメントの OPR イベント・サービスのリスト」(188ページ)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://<*server.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://<*server.example.com*>/opr-console/rest/9.10/event_list/{event}/annotation_list/{annotation}
注釈サービス:	http://<*server.example.com*>/opr-console/rest/9.10/event_list/{event}/annotation_list
自動アクション・サービス	http://<*server.example.com*>/opr-console/rest/9.10/event_list/{event}/auto_action
カスタム属性サービス:	http://<*server.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://<*server.example.com*>/opr-console/rest/9.10/event_list/{event}/custom_attribute_list
イベント変更サービス	http://<*server.example.com*>/opr-console/rest/event_change_list/{event_change}
イベント変更（複数）サービス	http://<*server.example.com*>/opr-console/rest/event_change_list
イベント・サービス	http://<*server.example.com*>/opr-console/rest/9.10/event_list/{event}
イベント（複数）サービス	http://<*server.example.com*>/opr-console/rest/9.10/event_list
履歴行サービス	http://<*server.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://<*server.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://<*server.example.com*>/opr-console/rest/9.10/event_list/{event}/user_action
症状サービス	http://<*server.example.com*>/opr-console/rest/9.10/event_list/{event}/symptom_list/{symptom}
症状サービス	http://<*server.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，「履歴行」(241ページ)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://<server.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，「メディア・タイプ」(230ページ)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://<*server.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，「イベント・ブラウザの URL 起動」(174ページ)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，イベント詳細の URL 起動を開始し，OMi ユーザ・インタフェースにドリルダウンします。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，「イベント・ブラウザの URL 起動」(174ページ)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。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，「HTTP クエリ・パラメータ」(222ページ)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://<*server.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://<*server.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: OMi 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，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。詳細については，「イベント Web サービスのセキュリティ」(202ページ)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/

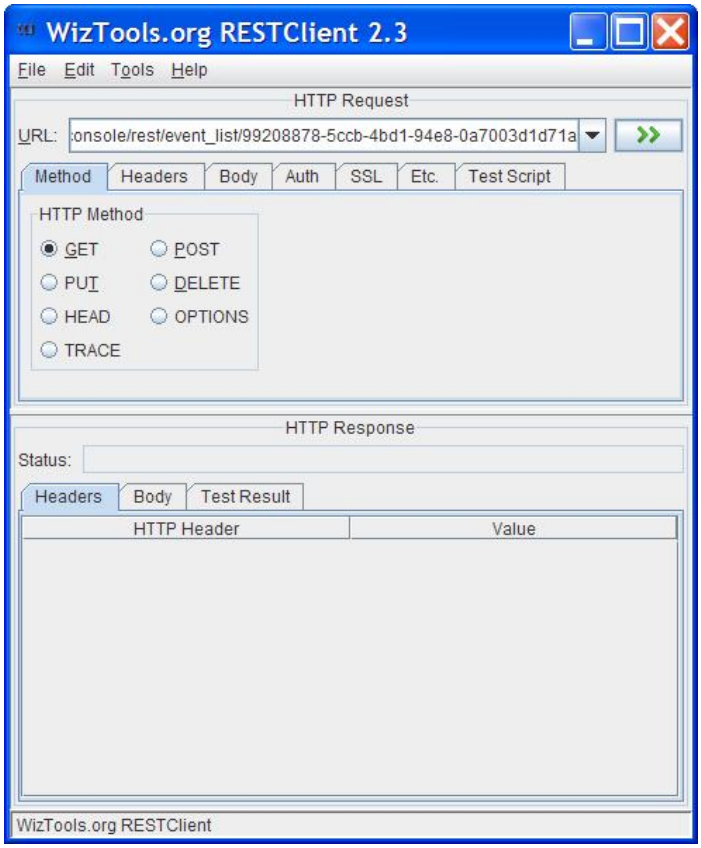
RESTClient ユーザ・インタフェースは 2 つのパートに分割されています。

- 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://<*server.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。

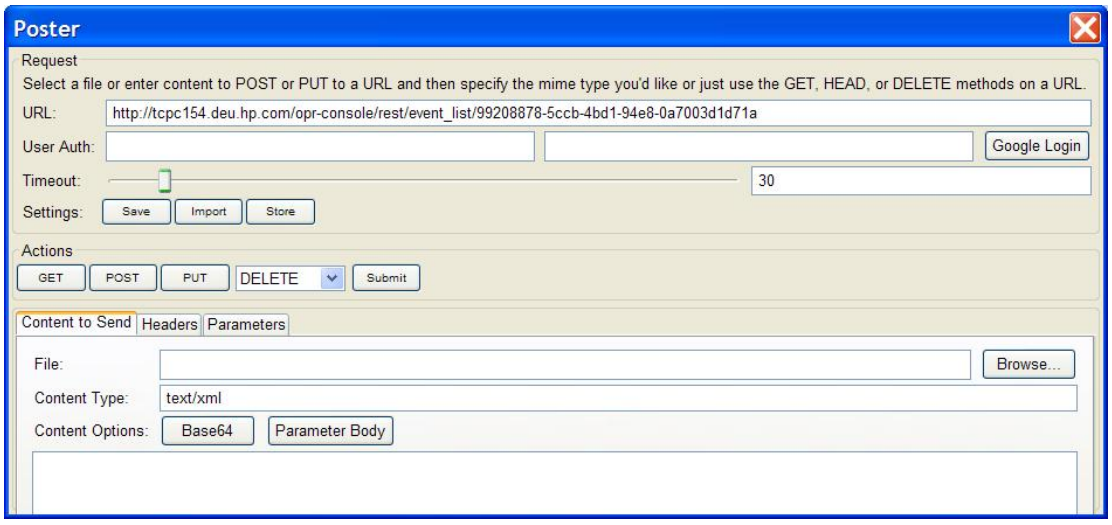
1. 最初に Mozilla Firefox Poster Extension をインストールします（まだインストールされていない場合）。

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。

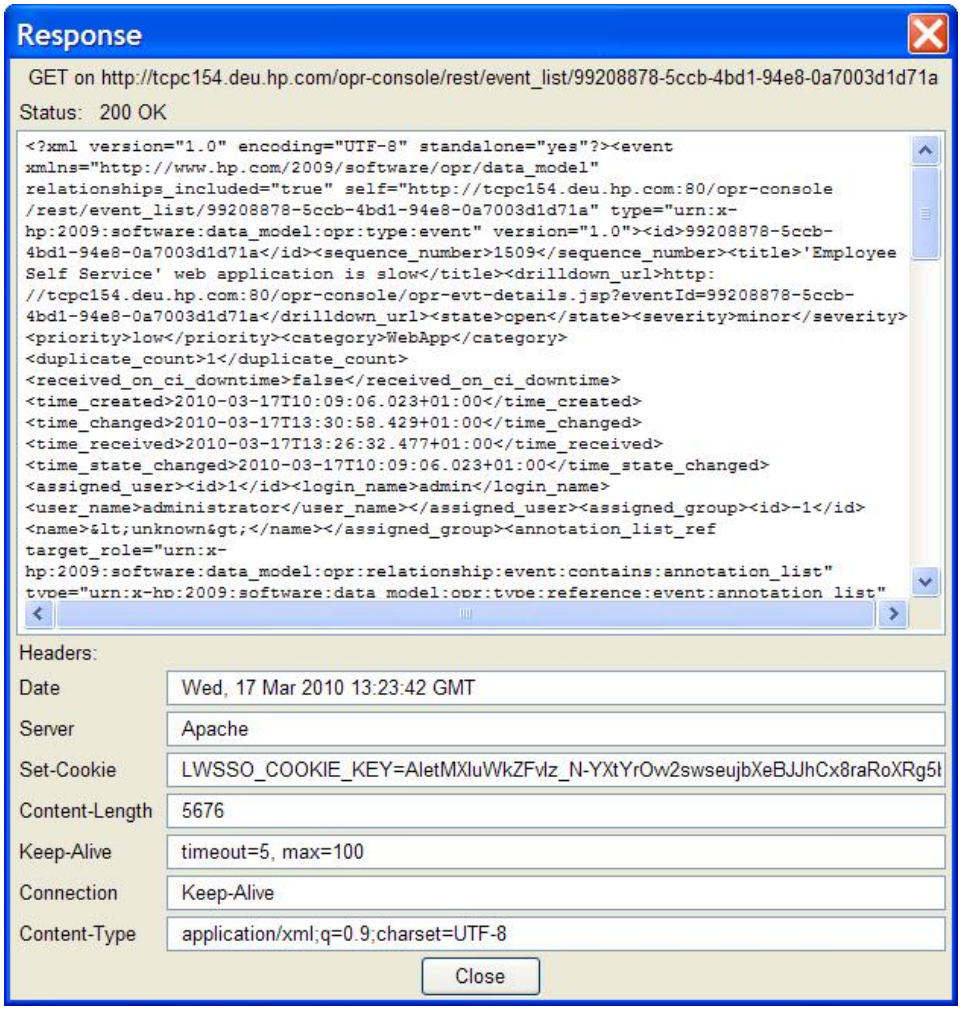
3. イベント・リスト・フィードから変更するイベントのイベント ID を取得します。

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://<*server.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，「編集可能なプロパティ」(239ページ)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。

<*OMi_HOME*>/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。

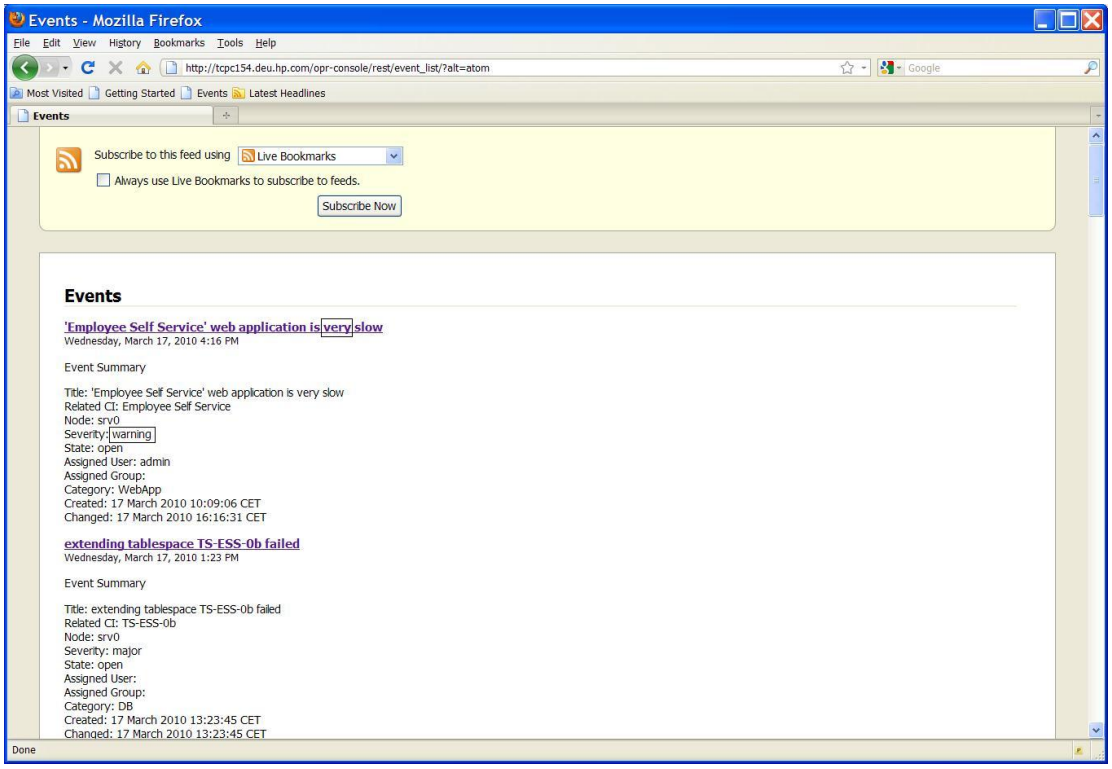
この例では，イベントのタイトルおよび重大度を直接 XML で変更します。たとえば，イベントの元のタイトルが“「'Employee Self Service' web application is slow」であると仮定します。「very」という単語を挿入して，イベントのタイトルを編集します。

...<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。

<*OMi_HOME*>/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。

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 **<*OMi_HOME*>/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。

   <*OMi_HOME*>/opr/bin>RestWsUtil -update update.xml -username <*login name*> -password <*password*> -url http://<*server.example.com*>/opr-console/rest/9.10/event_list/<*event_ID*>

   説明 :

   <*OMi_HOME*> は OMi がインストールされているディレクトリ，<*login name*> は認証に必要なユーザ名，<*event_ID*> は変更するイベントの 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，「REST Web サービス・コマンドライン・ユーティリティ」(211ページ)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，OMi ユーザ管理設定で Event Submission 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，OMi 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）。

   <*OMi_HOME*>/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。

   <*OMi_HOME*>/opr/bin>RestWsUtil -create create.xml -username <*login name*> -password <*password*> -url http://<server.example.com>/opr-console/rest/9.10/event_list

   説明 :

   <*OMi_HOME*> は，OMi 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，「REST Web サービス・コマンドライン・ユーティリティ」(211ページ)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，「イベント Web サービスにアクセスする方法」(187ページ) 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://<*server.example.com*>/opr-console/rest/9.10/event_list/<*event_ID*>/custom-attribute-list/

説明 :

<*server.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://<server.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://<server.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://<server.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://<server.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://<*server.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。イベントの一括更新は，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・リストのアドレスを指定し（「イベント Web サービス・クエリ言語」(222ページ)を参照），更新を PUT 要求のペイロード内で更新を指定すると実行できます。

例 :

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。詳細については，「変更操作のセキュリティ保護」(203ページ)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，「CA SiteMinder を使用した環境」(209ページ)を参照してください。

エラーの詳細度の変更

Web サービスが返すエラー・コードは，管理者が潜在的な問題の原因を特定するときに利用できます。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して，Web サービスが返すエラー・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に移動します。

   ［管理］>**［セットアップと保守］**>**［インフラストラクチャ設定］**

2. アプリケーション**［オペレーション管理］**を選択して，セクション**［オペレーション管理 - Web サービス設定］**>**［エラー応答の詳細］**を見つけます。

3. *オプション:* 標準設定値の **Standard** を別の値に変更します。

4. OMi 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。

   <*OMi_HOME*>/log/opr-ws-response.log

変更操作のセキュリティ保護

OMi 9.10 およびそれ以上では，Web サービスの変更操作は，変更要求（PUT，POST，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（「強化されたセキュリティ保護の無効化」(208ページ)を参照）。

X-Secure-Modify-Token HTTP ヘッダの設定

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 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 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，OMi 9.10 より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 Web サービス・クライアントは，OMi バージョン 9.0x 以下と一緒にインストールされている Web サービス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. 次のように，インフラストラクチャ設定マネージャに移動します。

   ［**管理**］>［**セットアップと保守**］>［**インフラストラクチャ設定**］

   ［**オペレーション管理 - Web サービス設定**］ > ［**安全な変更**］を編集します。

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，OMi 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，OMi 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）。OMi 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 OMi 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。詳細については，「値のタイプ」(235ページ)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

詳細については，「URL エスケープ・コード」(237ページ)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。詳細については，「演算子のエイリアス」(235ページ)を参照してください。

第22章: 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 :<*OMi_HOME*>/opr/bin/RestWsUtil.bat

Linux :<*OMi_HOME*>/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

注: RestWsUtil CLI ツールの実行時にスマート・カードを使用して認証できるように，ActivIdentity などの適切な個人認証ソフトウェアをインストールする必要があります。

リモート・デスクトップ接続を使用して OMi サーバに接続する際，RestWsUtil CLI ツールのリモート実行時に，クライアント・システムに付属するスマート・カードを使用して認証できるように，クライアント・システムに ActivIdentity などの適切な個人認証ソフトウェアをインストールしておく必要があります。

In addition, in the Remote Desktop Connections dialog box, select More **Show Options > Local Resources > More** to open the Local devices and resources dialog box.［**Smart Cards**］チェック・ボックスが選択されていることを確認してください。

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。
-j, -jks <arg>	Specifies the java key store to be used for authentication.
-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。
-sc, -smartcard	スマート・カードまたはセキュリティ・トークンに保存されている証明書を使用して認証を行います。
-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。
-wc, -winCrypto	Windows 証明書ストアを使用して認証を行います。このオプションは，Windows システムでのみ利用できます。

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://server.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 OMi 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 OMi 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 OMi 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 OMi 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 OMi 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 OMi 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 OMi 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 OMi 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要度の更新

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 OMi 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 OMi 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の重大度が critical に設定され，状態が in_progress 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 OMi 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 OMi 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 OMi 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://<server.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://<server.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。

第23章: 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。

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。

「クエリ・フィルタ条件のプロパティ」(231ページ)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。

本項の内容

- 「データ包含」(230ページ)
- 「メディア・タイプ」(230ページ)

HTTP クエリ・パラメータの一覧

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。 詳細については，「日付および時間によるフィルタリング :watermark」(225ページ)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。 詳細については，「イベント属性によるフィルタリング : query」(226ページ)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。 詳細については，「ページング」(227ページ)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。 詳細については，「ページング」(227ページ)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。 詳細については，「順番付け」(229ページ)を参照してください。
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。 詳細については，「順番付け」(229ページ)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 詳細については，「データ包含」(230ページ)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 詳細については，「データ包含」(230ページ)を参照してください。

クエリ・パラメータ	説明
include_history	イベント・サービス（event_list）をクエリするときにイベントに履歴行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 詳細については，「データ包含」(230ページ)を参照してください。
include_cis	イベント・サービス（event_list）をクエリするときに関連する CI，ノード，影響を受けるビジネス・サービスのイベントに構成アイテムのプロパティ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 詳細については，「データ包含」(230ページ)を参照してください。

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，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。

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。

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。

時間形式の要件 :

- **dateTime 形式 :** 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）

- **タイム・ゾーン・エンコーディング：** 「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，"%2B" に置き換えることで URL エンコーディングされている必要があります。その例を次に示します。

  query=time_created%20GT%202009-10-19T17:06:54.453%2B02:00

  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，「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」(237ページ)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

例：

- 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://<*server.example.com*>/opr-console/rest/9.10/event_list?alt=atom&watermark=2010-03-19T13:59:17%2B02:00

- 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://<*server.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

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://<*server.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 NOT，AND，OR で区切ることで組み合わせることができます。これらの論理演算子は指定した順序で解決されます。クエリを異なる順序で処理するには，括弧を使用してください。

例：

- 次の例では，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://<*server.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://<*server.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://<*server.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，NOT 演算子を使用して式を否定する方法を示しています。

  http://<*server.example.com*>/opr-console/rest/9.10/event_list?query=assigned_user%20EQ%20"admin"%20AND%20(title%20EQ%20"My Title"%20OR%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 未満の値が指定された場合，イベント Web サービスはエラーを返します。

次の例では，warning に設定した severity パラメータでフィルタリングされたクエリ結果の 30 項目を表示するページを返します。

http://<*server.example.com*>/opr-console/rest/9.10/event_list?alt=atom&query=severity%20EQ%20"warning"&page_size=30

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://<*server.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://<*server.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）

順番付け

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（「新規イベントを検出する方法」(190ページ)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

「クエリ・フィルタ条件のプロパティ」(231ページ)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，「複雑な属性」(238ページ)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		

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[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）		
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		
node[target_global_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
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		
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		

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
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		
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）		

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
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

「演算子のエイリアス」(235ページ)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

「クエリ・フィルタ条件のプロパティ」(231ページ)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。文字列リテラルの比較では大文字と小文字を区別します。 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（「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」(237ページ)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%20EQ%20'%3CMy title%3E' の代わりに query=title%20EQ%20"$3CMy title$3E" を使用します。 次のクエリは，タイトルに文字列 cpu および value を含むすべてのイベントを返します。URL エンコーディング $25 は SQL ワイルドカード % を表し，これは 0 文字以上の文字に一致します（正規表現 .* に類似）。 query=title%20LIKE%20"$25cpu$25value$25" **注:** OMi 9.0x 以前では，シングルの引用符（'）およびパーセント記号（%）のエスケープ・コードのみがサポートされます。OMi 9.0x では，ドル記号（$）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 以降では，シングル（'）および二重（"）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。
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。「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」(237ページ)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

「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」(237ページ)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%20EQ%20'%3CMy title%3E' の代わりに query=title%20EQ%20"$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

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 (続き)

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
|	%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://<*server.example.com*>/opr-console/rest/9.10/event_list?query=related_ci%5Btarget_id="ffca22eea15268533029f17b4c01b008"%5D

http://<*server.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，「URL エスケープ・コード」(237ページ)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 「編集可能なイベント・プロパティの一覧」(239ページ) 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，「ファイルの場所」(243ページ) 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	
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

編集可能なイベント・プロパティの一覧 (続き)

タグの名前	タイプ	説明	コメント
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	リスト	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	リスト	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	リスト	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。
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，「記録されたプロパティの変更」(241ページ)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
key_pattern	OprStringPropertyChange
node_hint	OprStringPropertyChange

記録されたプロパティの変更 (続き)

名前	ChangeType
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://<*server.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，「日付および時間によるフィルタリング :watermark」 (225ページ)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 :

  <*OMi_HOME*>/opr/api/doc/opr-external-api-javadoc.zip

- イベント Web サービス・スキーマ :

  <*OMi_HOME*>/opr/api/schema/OprDataModel.xsd

第VII部: 外部イベント・プロセスの統合

イベント Web サービスによって，インテグレータを HP Service Manager や BMC Remedy Service Desk などの外部インシデント管理アプリケーションとインタフェースで接続できます。Web サービスでは，次の機能がサポートされています。

- 転送されたイベントの通知の受信
- イベント変更の通知の受信

本項の内容

第24章: 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。

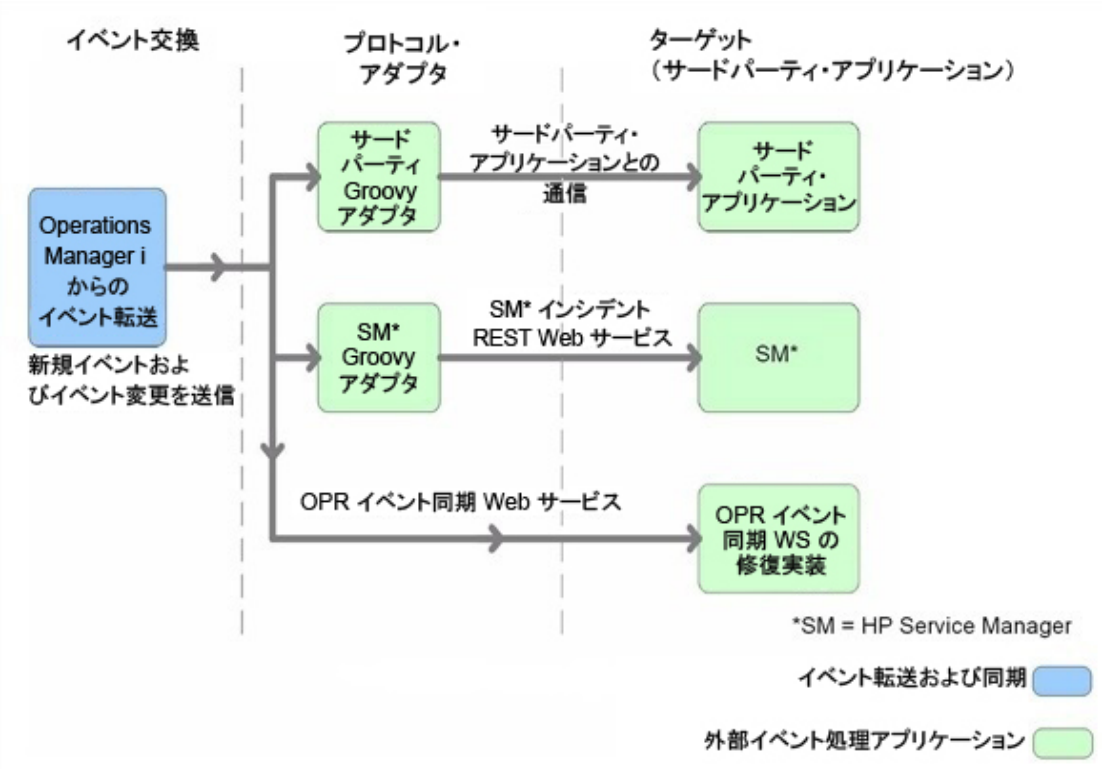
イベント転送および同期プロセス

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。

接続サーバおよび転送ルールの設定方法の詳細については，OMi 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。
- **通知して更新：**［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：**［通知して更新］と同様ですが，イベントに加えられたすべての変更の逆同期を求められるターゲット接続サーバに対して，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 OMi 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・プロセスは次の方法で統合できます。

Groovy スクリプト・アダプタの使用

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。 Groovy スクリプトの開発およびデプロイの詳細については，「Groovy スクリプト」(376ページ)を参照してください。

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（「サンプル Groovy スクリプト :ログファイル・アダプタ」(251ページ) 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，「Groovy スクリプトを使用した外部イベント・プロセスの統合」(251ページ)を参照してください。

イベント同期 Web サービスの使用

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OMiから受信することが可能です。また 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 OMi 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と考えられます。

イベントおよびそれ以降のイベントの変更を転送するために，次の標準 REST Web サービスの HTTP メソッド呼び出しが OMi 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。

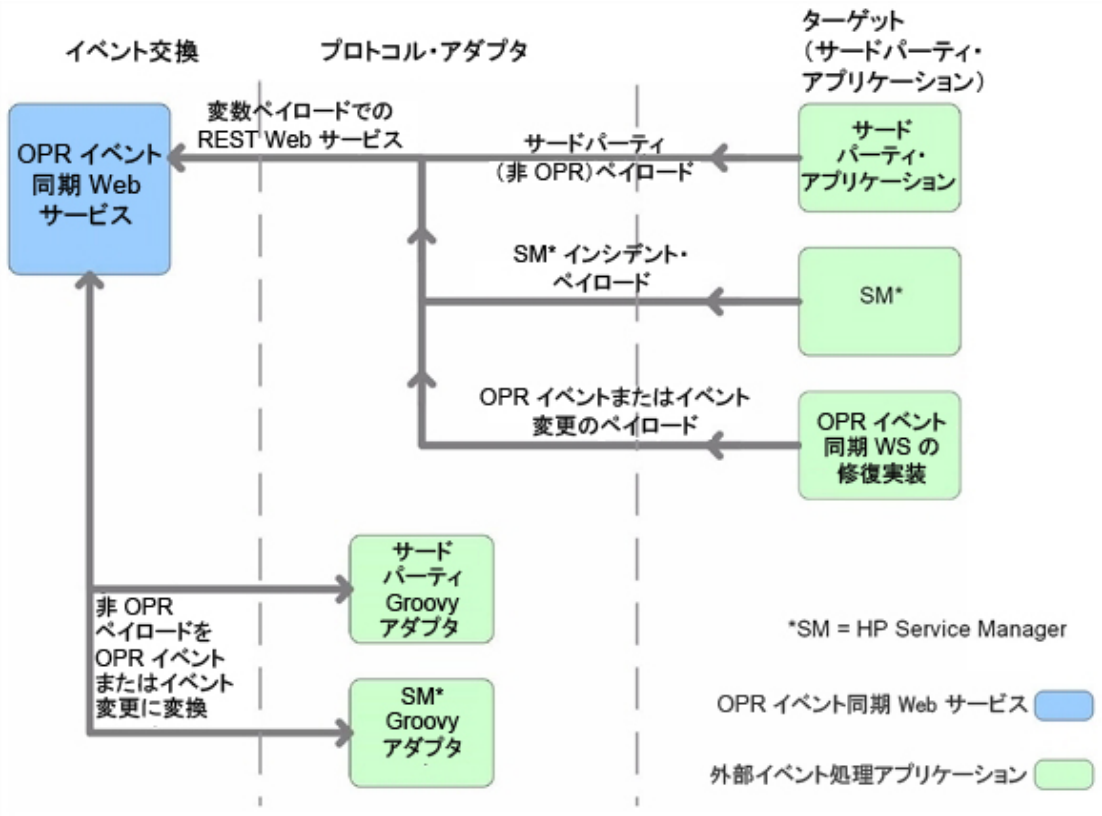
- **GET :** GET 呼び出しが外部イベントの現在の状態を取得するために使用されます。/event/<external_event_ID> パラメータを付加された，ターゲット接続サーバに対して設定されているベース URL は，エンドポイントのアドレスを指定するために使用できます。期待される応答ペイロードは OPR イベント・オブジェクトです。

- **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，「イベント同期 Web サービス・インタフェース」(272ページ)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OMi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。その場合，転送ルールを構成するときに，［同期］転送タイプまたは［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。

また OMi では，外部アプリケーションにより外部イベントの変更を逆同期する場合に使用できる，次の WS PUT メソッドのエンドポイントも利用できます。

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，OMi 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，「イベント同期 Web サービス・インタフェース」(272ページ)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 OMi 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 OMi 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 OMi 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*ユーザ名を使用して OMi でオペレータを設定します（OMi 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 OMi 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，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。

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，「イベント・ブラウザの URL 起動」(174ページ)を参照してください。

第25章: 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 スクリプティングがサポートされています。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 スクリプトの開発およびデプロイの詳細については，「Groovy スクリプト」(376ページ)を参照してください。

「Groovy スクリプトおよびプログラミング」(276ページ)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・スクリプトは次の場所にあります。

Administration > Setup and Maintenance > Connected Servers

Click **Manage Scripts**.

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。

<OMi_HOME>/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： <OMi_HOME>/opr/examples/external-event-adapter に含まれます。

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

OMi 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 OMi 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詳細については，OMi 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. 接続サーバ・マネージャに移動します。

   Administration > Setup and Maintenance > Connected Servers

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 OMi 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，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. 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，OMi 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し，さらに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 OMi 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，OMi 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 OMi 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. ［**完了**］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。

フィルタの設定の詳細については，OMi 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. イベント転送マネージャに移動します。

   ［管理］ > ［イベント処理］ > ［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，「サンプル Groovy スクリプト :ログファイル・アダプタ」(251 ページ) 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，「Javadoc API ドキュメント」(256ページ)を参照してください。 Groovy スクリプトの開発およびデプロイの詳細については，「Groovy スクリプト」(376ページ)を参照してください。

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（「OPR イベント・スキーマ」(256ページ)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。

<OMi_HOME>/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。

<OMi_HOME>/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。

<OMi_HOME>/opr/api/schema/OprDataModel.xsd

イベント統合 Groovy スクリプト

イベント統合 Groovy スクリプトはデータベースに保存され，次の場所からアクセスできます。

Administration > Setup and Maintenance > Connected Servers

［接続サーバ］表示枠で，  ボタンをクリックし，［イベント転送スクリプト構成］ダイアログ・ボックスを開きます。このダイアログ・ボックスから，既存のスクリプトの選択や新しいスクリプトの作成を行うことができます。

すぐに使用できるスクリプトは次のとおりです。

ログファイル・アダプタ（タイプ）: sample:LogfileAdapter

Service Manager アダプタ（タイプ）: sm:ServiceManagerAdapter

アップグレード・アダプタ（タイプ）: upgrade:UpgradeAdapter

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，destroy，ping メソッド

Groovy スクリプトの開始と終了，Groovy スクリプト・パラメータの検証方法を本項に示します。

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，「init() メソッドのプロパティ」(257ページ)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	OMi 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。

プロパティ	説明
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。

「ping() メソッドのプロパティ」(258ページ) 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	OMi 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。

プロパティ	説明
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，「forwardEvent() メソッドのプロパティ」(259ページ)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	
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。

プロパティ	説明
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.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。
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	
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，「forwardEvents() メソッドのプロパティ」(261ページ)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，「forwardChange() メソッドのプロパティ」(262ページ)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	説明
読み取りプロパティ	
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。

プロパティ	説明
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。
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，「forwardChanges() メソッドのプロパティ」(263ページ)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。
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。

プロパティ	説明
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，「receiveChange() メソッドのプロパティ」(265ページ)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。

プロパティ	説明
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。
書き込みプロパティ	
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。

プロパティ	説明
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 OMi 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，「receiveChanges() メソッドのプロパティ」(269ページ)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，「getExternalEvent() メソッドのプロパティ」(270ページ)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 :外部イベントのタイトル。
description	オプション :外部イベントの説明。
assignedUser	オプション :外部イベントに割り当てられたユーザ。
assignedGroup	オプション :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 :イベントの重要度。
priority	オプション :イベントの優先度。
state	オプション :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

第26章: 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・インタフェースは，OMi 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。

Web サービスに関する詳細情報については，「イベント同期 Web サービス・インタフェースの参照情報」(352ページ)を参照してください。

第27章: 外部イベント・プロセスの統合 :よくある質問

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，「イベントの転送およびイベント変更の同期」(245ページ)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，「イベント同期 Web サービス・インタフェース」(272ページ)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，「Groovy スクリプトを使用した外部イベント・プロセスの統合」(251ページ)を参照してください。
- Groovy スクリプト使用の概要については，「Groovy スクリプト」(376ページ)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。

<*OMi_HOME*>/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。

<*OMi_HOME*>/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。<*OMi_HOME*>/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。利用でき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 LogfileAdatper サンプルを参照してください。

1. 接続サーバ・マネージャを開きます。

   Administration > Setup and Maintenance > Connected Servers

2. ［接続サーバ］表示枠で，ボタンをクリックして［イベント転送スクリプト設定］ダイアログ・ボックスを開きます。

3. sample.LogfileAdatper Groovy スクリプトを開きます。

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 : *<DNS name of your OMi node>*
- 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，sample.LogfileAdapter 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?

- 次の場所で新しいスクリプトを作成します。

Administration > Setup and Maintenance > Connected Servers

Click **Manage Scripts**.

- 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。

  <*OMi_HOME*>/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，ログインしている場合，コントロールがローカルの OMi 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;
  // これが event または event_change の場合，チェックする
  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.")
  // XML に存在する場合，イベント・プロパティを更新する
  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
    }
  }
...
```

外部処理の重要度を OMi イベントの重要度にマッピングしたいのですが，Groovy で簡単にそれを行う方法はありますか。

次のようなマップを使用してください。

```
// 重大度にマッピングする
 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，「Groovy スクリプト・メソッド」(256ページ)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，OMi 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，「init() メソッドのプロパティ」(257ページ)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，「ping() メソッドのプロパティ」(258ページ)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，「forwardEvent() メソッドのプロパティ」(259ページ)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:${extId}\n\n"""
  m_logfile.append(msg)
  // 外部参照 ID を設定する
  args.externalRefId = extId
  // 例として元のイベントにドリルダウンする
  args.drilldownUrl =
    new URL("http://${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, 「forwardChange() メソッドのプロパティ」(262ページ)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, 「receiveChange() メソッドのプロパティ」(265ページ)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，「getExternalEvent() メソッドのプロパティ」(270ページ)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://<*server.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。

<*OMi_HOME*>/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。

<*OMi_HOME*>/conf/core/Tools/log4j/jboss/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。

<*OMi_HOME*>/conf/core/Tools/log4j/wde/opr-event-sync-adapter.properties

<*OMi_HOME*>/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。例 :

```
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)
```

第28章: 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。次の各段階では，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 OMi 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，「Groovy スクリプト・メソッド」(256ページ)を参照してください。

次の設定手順が，通知の転送タイプの統合を実装するために必要です。

1. 「WSDL から Java コードを生成する」(287ページ)
2. 「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」(288ページ)
3. 「外部イベント作成のテスト」(290ページ)

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 1.7 以降

   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。

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

OMi 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 OMi 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。

接続したサーバの構成を行う場合，Groovy スクリプトも作成してサービスにアクセスします。

接続サーバの設定方法の詳細については，『OMi 管理ガイド』のサーバ接続の項を参照してください。

1. 接続サーバ・マネージャに移動します。

   Administration > Setup and Maintenance > Connected Servers

2. ［ **新規**］ボタンをクリックして［サーバ接続の新規作成］ダイアログを開きます。

3. ［**表示名**］フィールドに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。
［表示名］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。

   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，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. 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. 外部サーバとの接続を確立するために使用する**統合タイプ**を選択します。

   a. ［**スクリプト アダプタの呼び出し**］を選択します。

   b. ［**スクリプト名**］で［**スクリプトの管理**］をクリックし，［イベント転送スクリプト構成］ダイアログ・ボックスを開きます。

   c. sample:LogfileAdapter Groovy スクリプトを選択し，  ［**項目を複製**］をクリックしてスクリプトのコピーを作成します。

   d. スクリプト・エディタの［一般］タブで，コピーの名前を TestAdapter に変更します。

   e. ［スクリプト］タブで 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。

   f. ［詳細設定］タブで，TestAdapter スクリプトに必要なすべての JAR ファイルを追加します。

g. ［**OK**］をクリックして TestAdapter スクリプトを保存し［イベント転送スクリプト構成］ダイアログ・ボックスを閉じます。

h. 接続サーバ・ウィザードの［統合タイプ］ページで［**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］フラグがオンに設定されると，OMi 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。

   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ります。

   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。

   ［**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. OMi 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。

第29章: 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 OMi 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。

- 「接続サーバとしての HP Service Manager サーバの設定」(292ページ)
- 「イベント転送ルールの設定」(295ページ)
- 「HP Service Manager サーバの設定」(298ページ)

接続サーバとしての HP Service Manager サーバの設定

OMi イベントと HP Service Manager のインシデントの間でイベントとイベントの変更を同期するには，ターゲットの Service Manager インスタンスを正確に識別するように OMi 内で接続サーバを設定する必要があります。そのための最初の手順として，HP Service Manager を接続サーバ・マネージャでターゲット接続サーバとして設定します。

接続サーバの設定方法に関する詳細については，OMi オンライン・ヘルプの接続サーバの項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. 接続サーバ・マネージャに移動します。

   Administration > Setup and Maintenance > Connected Servers

2. ［新規］（  ）ボタンをクリックして，［**サーバ接続の新規作成**］ダイアログを開きます。

3. ［**表示名**］フィールドに，ターゲットの 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，ターゲットの HP Service Manager サーバの［表示名］として「Service Manager 1」と入力すると，Service_Manager_1 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。

   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，Service_Manager_1）。この名前は後で HP Service Manager サーバを設定して OMi 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 のような外部インシデント・マネージャ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. 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. Service Manager の Groovy スクリプト・アダプタは HP Service Manager との統合用に用意されているため，［**スクリプト アダプタの呼び出し**］を選択します。

   b. ［スクリプト名］フィールドで，**sm:ServiceManagerAdapter** を選択します。

   c. ［**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・サーバへのアクセスに必要なユーザ名とパスワードです。

8. OMi ユーザ・インタフェースで実行する次の手順では，資格情報（ユーザ名，パスワード，ポート番号）を入力して HP Service Manager ターゲット・サーバに接続し，イベントをそのサーバに転送します。［**送信接続**］ダイアログで，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)**］フィールドにパスワードを再入力します。

   c. ［**ポート**］フィールドで，OMi との統合のために HP Service Manager 側で設定したポートを指定します。ポート番号を調べるには，次のように入力します。

      ◦ 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 ファイルには，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・エントリが記載されています。1 つは標準設定ポート番号が 13080 の httpPort，もう 1 つは標準設定ポート番号が 13443 の httpsPort です。ポートの実際の値は，設定に応じて，これらの標準設定の値と異なります。［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。

［セキュア HTTP を使用］が選択されている場合は，「サーバから検索」のリンクをクリックするか，ローカル・ファイルに証明書がある場合には「ファイルからインポート」のリンクをクリックすることにより，ターゲット・サーバの SSL 証明書のコピーをダウンロードしてインストールします。

e. ［**同期してコントロールを移す機能をサポート**］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機能をサポート］フラグが設定されている場合，OMi 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すことができます。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［同期してコントロールを移す機能をサポート］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［同期してコントロールを移す機能をサポート］フラグが設定されていない場合，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。

f. 接続をテストします。

g. ［**次へ**］をクリックします。

9. OMi イベントから HP Service Manager インシデントを自動生成することに加えて，HP Service Manager をドリルダウンできるようにする場合は，インシデントのドリルダウンを実行する対象の 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 のインシデントの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/設定の指示に従って，HP Service Manager 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，使用している設定済みのポートを指定して 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い場合，Web 層クライアントは，イベントとイベントの変更を HP Service Manager に転送し，また HP Service Manager からイベントの変更を受け取るサーバに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せず，Web 層クライアントが HP Service Manager サーバ・マシンにインストールされていない場合，Web ブラウザでは要求された URL を見つけることができなくなります。

［**次へ**］をクリックします。

10. 次の手順では，HP Service Manager から OMi にイベントの変更を逆同期することができるようにします。このためには，HP Service Manager サーバの資格情報を入力して，OMi 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**］ダイアログで，［**イベント BackSync をサポート**］チェックボックスを選択し，OMi をホストしているサーバに接続するために HP Service Manager サーバで必要とされるパスワード（この例では Myqwer1_）を入力します。

    注: このパスワードを書き留めます（この例では，Myqwer1_）。この名前は，後で OMi をホストしているサーバと通信できるように HP Service Manager サーバを設定するときに必要になります。このパスワードは，手順 3 で設定したサーバ名（Service_Manager_1）と組になります。

    b. ［**完了**］をクリックします。ターゲットの HP Service Manager 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，HP Service Manager に自動的に転送されるイベントを決定するイベント転送ルールを設定します。

フィルタの設定の詳細については，OMi 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. 転送ルール・マネージャに移動します。

   ［管理］ > ［イベント処理］ > ［自動化］ > ［イベント転送］

2. ［新規作成］（ ）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（この例では，Forward Critical (Sync and Transfer Control)）を入力します。

   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。HP Service Manager でステータスを使用できるようにするには，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 フィルタの選択**］ダイアログ・ボックスで，次のいずれかを実行します。

   - 既存フィルタを選択する
   - 次の手順に従って新規ファイルを作成する

     i. ［新規作成］（  ）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**］フィールドに新規フィルタの名前（この例では，**FilterCritical**）を入力します。

     重要度が「**致命的**」の場合を除いて，すべての重要度レベルのチェック・ボックスを選択解除します。

     ［**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**］で，前の項「接続サーバとしての HP Service Manager サーバの設定」(292ページ)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。［**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，「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」(245ページ)の項を参照してください。

HP Service Manager からイベント・ブラウザの URL 起動の設定

イベント・ブラウザの URL 起動を使用して HP Service Manager から OMi 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ようにするため，オペレータは，適切な権限を持つ OMi の有効なユーザとして設定されている必要があります。

- **ユーザ・アカウントの要件**

  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 のオペレータが HP Service Manager にログインして URL 呼び出しを実行するために使用するのと*同じ*ユーザ名を使用して，各ユーザを OMi で設定します。（各 OMi ユーザのパスワードは，空または任意の文字列に設定できます）。HP Service Manager へのログインに成功すれば，OMi ユーザは，追加の認証なしでイベント・ブラウザを起動できます。

  SSO 認証を使用するように HP Service Manager が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のと同じユーザ名を使用して各ユーザを設定し，有効なパスワードを指定します。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を入力する必要があります。

- **必要なユーザ権限**

  各 OMi ユーザには，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］権限を付与する必要があります。またオプションとして，各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もできます。

注: 有効なユーザ名がない場合，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。

「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」(250ページ)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. 次のようにして，Groovy スクリプト sm.ServiceManagerAdapter に移動します。

   a. 接続サーバにアクセスします。

      Administration > Setup and Maintenance > Connected Servers

   b. ［接続サーバ］表示枠で，ボタンをクリックして［イベント転送スクリプト設定］ダイアログ・ボックスを開きます。

   c. sm.ServiceManagerAdapter Groovy スクリプトを開きます。

2. Groovy スクリプト内で次のテキストを見つけます。

   private static final String SM_WEB_TIER_NAME = 'webtier-9.30'

3. webtier-9.30 の値を，HP Service Manager の 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 は次のようになります。

   http:// < *HP Service Manager の Web 層サーバの FADNS* > / < HP Service Manager への Web パス > / < *URL クエリのパラメータ* >

   < *HP Service Manager の 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 の完全修飾 DNS 名です。この URL 部は，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指定した値（「接続サーバとしての HP Service Manager サーバの設定」(292ページ)を参照）に応じて，（http:// と一緒に）自動的に追加されます。

   ドリルダウン URL の例を次に示します。

   http://smserver.example.com/SM930/index.do?ctx=docEngine&file=probsummary&query=number%3D

   この例では，webtier-9.30 を SM930 で置き換える必要があります。URL の他のすべての部分は自動的に設定されます。

4. HP Service Manager の 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** に設定します。

   詳細については，HP Service Manager のオンライン・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，OMi 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表示枠から，次のように移動します。

   ［**カスタマイズ**］ > ［**統合マネージャ**］

2. **［追加］**をクリックして新しい設定を追加します。

3. ［統合テンプレート］フィールドから **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. オプション :ログ・レベルを適切な値に変更します。

   オプション :説明を変更します（This is for SMOMi integration など）。

   ［**次へ**］をクリックします。

5. ［全般パラメータ］タブで，既存のエントリを次の値で置き換えます。

名前	値	カテゴリ
omi.server.url	http://<*gateway_FQDN*>/opr-gateway/rest/9.10/《》synchronization/event/	一般
username	Service_Manager_1 （これは，「接続サーバとしての HP Service Manager サーバの設定」(292 ページ)の項ですでに設定済みの HP Service Manager ターゲット・サーバの名前です）。	ヘッダ
omi.eventdetail.baseurl	http://<*gateway_FQDN*>/opr-console/opr-evt-details.jsp?《》eventId=	一般

6. ［**セキュア パラメータ**］タブで，「接続サーバとしての HP Service Manager サーバの設定」(292ページ)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接続にパスワードを設定します。例において，これは HPqwer1_ です。

   ［**次へ**］をクリックします。

7. ［統合インスタンス］の［フィールド］ダイアログで，［**次へ**］をクリックします。

8. ［統合インスタンス］の［マッピング］ダイアログで，［**完了**］をクリックします。

注: ルールを必ずアクティブにします。ルールをアクティブにするには，ルールを選択して［**有効化**］をクリックします。

マッピングおよびカスタマイズ

Groovy スクリプトに独自のカスタム属性を追加し，追加したカスタム属性を HP Service Manager の適切なフィールドにマップできます。OMi から HP Service Manager への属性のマッピングを変更することもできます。マッピングは HP Service Manager の BDM マッピング・マネージャで実行します。

［**システム管理**］ > ［**継続的な保守**］ > ［**BDM マッピング管理**］

属性のマッピングの詳細については，HP Service Manager のオンライン・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，定義したフィルタに一致する OMi をホストするサーバにイベントを送信し（例示したフィルタでは，重要度値は Critical），イベントが予期したとおりに 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. OMi 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. コマンド・プロンプトを開き，次のディレクトリに変更します。

   <*OMi_HOME*>\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. イベントがイベント・ブラウザに表示されることを確認します。
5. ［**転送**］タブを選択します。
6. ［外部 ID］フィールドで，有効な HP Service Manager インシデント ID を参照する必要があります。
7. 次に，インシデントが HP Service Manager のインシデントの詳細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適切に設定されている場合，［**編集**］ボタンをクリックします。ブラウザ・ウィンドウが開くと，HP Service Manager のインシデントの詳細にインシデントが直接表示されます。

イベント・ドリルダウン接続が設定されていない場合，次の手順を実行します。

a. イベント・ブラウザの［**転送**］タブで，［**外部 ID**］フィールドのインシデント ID をコピーするか書き留めます。

b. HP Service Manager ユーザ・インタフェースで次の場所に移動します。

［**インシデント管理**］ > ［**インシデントの検索**］

c. ［**インシデント 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。この手順で，インシデントの詳細にインシデントが表示されます。

8. HP Service Manager の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更（現在は closed）が OMi に同期しなおされることを確認します。アクティブなイベント・ブラウザでは HP Service Manager でクローズしたイベントを表示できませんが，履歴ブラウザには表示されます。

属性の同期

標準設定では，すべての属性が HP Service Manager から OMi へ同期しなおされるわけではありません。OMi から HP Service Manager へ 1 回かぎり一方向に更新される属性もあれば，双方向に同期される属性もあります。

一方向の同期 : OMi から HP Service Manager

次の属性は，OMi から HP Service Manager に 1 回かぎり転送されます。つまり，イベントが最初に作成されたとき，イベントのコントロールの転送の設定が接続サーバ・マネージャに行われます。

- タイトル
- 重要度
- 優先度
- Operator:イベントを転送したオペレータがイベントに割り当てられる
- カテゴリ

- サブカテゴリ
- 関連 CI

上の属性の場合，HP Service Manager から OMi に同期しなおされません。

双方向の同期

OMi と HP Service Manager 間の双方向の同期をサポートする属性 :

- 説明
- ライフサイクル状態（ライフサイクル状態は，状態がクローズ済みに変更されたときのみ更新されます）
- ソリューション
- イベントの注釈は，HP Service Manager のアクティビティ・ログに同期されます。
- イベント詳細の［転送］タブのコンテンツ

Groovy スクリプトを使用した属性の同期

更新される属性に関するすぐに使用できる動作を変更するには，Groovy スクリプトで指定できます。Groovy スクリプトで，HP Service Manager の更新されるフィールドおよび OMi の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可能です。

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ついてのヒントをいくつか紹介します。次に，カスタマイズ可能な Groovy スクリプトの例を選び出して示します。変更可能なその他のアイテムは，Groovy スクリプトの構成セクションで確認できます。

Groovy スクリプトの構成セクションで，OMi と HP Service Manager の間で同期される属性の定義および変更を行うことができます。Groovy スクリプトの構成セクションには，ライフサイクル状態，重要度，優先度の標準設定値のマッピングも含まれます。これらも変更でき，入力要求と出力要求のマッピングを別々に定義できます。

詳細設定は，必要に応じて Groovy スクリプトの別の部分で実行できます。

Groovy スクリプトの構成セクションの開始と終了は次のようにマークされます。

```
//
// Groovy スクリプトをカスタマイズするための構成セクション
// 開始
...
```

...

//

// Groovy スクリプトをカスタマイズするための構成セクション

// 終了

Groovy スクリプトを変更する前に，元の（製品付属の）スクリプトのコピーを作成します。これは，Groovy スクリプトの新しいバージョンとともに，パッチ，サービス・パック，ホットフィックスが配信され，元のスクリプトを上書きする可能性があるためです。カスタマイズした Groovy スクリプトを安全な場所にコピーしたことを確認します。パッチ，サービス・パック，ホットフィックスとともに配信された変更を新しい Groovy スクリプトにマージすることが必要になる場合があります。

OMi から HP Service Manager へのマッピングは BDM 1.1 インシデント Web サービスの仕様に準拠しています。HP Service Manager への BDM 1.1 インシデント Web サービスのマッピングは BDM マッピング・マネージャの HP Service Manager で指定します。BDM マッピング・マネージャの詳細については，HP Service Manager のオンライン・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。

属性の同期の制御

ブール変数を true または false に設定して，特定の属性の更新が OMi と HP Service Manager の間でどのように同期されるかを制御できます。

次に 2 つの例を示します。

- private static final SyncTitleToSMOnUpdate = false;

  Groovy スクリプトのこの行によって，OMi で行われたタイトルの変更の HP Service Manager への同期が無効になります。

- private static final Boolean SyncTitleToOPROnUpdate = false;

  Groovy スクリプトのこの行によって，HP Service Manager で行われたタイトルの変更の OMi への同期が無効になります。

タイトルは HP Service Manager の必須の属性であり，インシデントの作成中に OMi で指定したタイトルを使用して，上記のフラグと独立して設定されます。

BDM ライフサイクル状態への OPR ライフサイクル状態のマッピング

Groovy スクリプトを変更して，HP Service Manager の OMi の（OPR）ライフサイクル状態を（BDM）ライフサイクル状態にマップできます。

次に 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 "closed" のみが BDM ライフサイクル状態 "closed" にマップされます。null は，HP Service Manager の状態が変更されていないことを示します。

- private static final Map OPR2BDMLifecycleState = ["open":null, "in_progress":null, "resolved":"resolved", "closed":"closed"];

  この例では，OPR 状態 resolved によって，BDM 状態が resolved に設定されます。

  インシデントの作成中にインシデントの初期のライフサイクル状態が設定されるため，OPR ライフサイクル状態 open を SM ライフサイクル状態 open にマップする必要はありません。

OPR ライフサイクル状態への BDM ライフサイクル状態のマッピング

次の構成行で，BDM ライフサイクル状態から既知の OPR ライフサイクル状態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したときに（OPR）イベントがクローズします。null は，HP Service Manager のインシデントの状態が変更された場合にイベントの状態が変更されていないことを示します。

BDM ライフサイクル状態 open を OPR ライフサイクル状態 open にマップする場合，次のことが起こります。HP Service Manager でインシデントをクローズして再度開くと，OMi の対応するイベントが再度開かれます。

構文エラー

Groovy スクリプトをカスタマイズしているときに構文エラーが起こるときは，ログ・ファイル opr-event-sync-adapter.log を確認してエラーを解決する情報を見つけます。ログ・ファイルは次の場所にあります。

<*OMi_HOME*>/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の付近には 2 つのセクションがあり，そこで Service Manager との OMi イベント同期の標準設定動作を変更します。

アクセス方法

Administration > Setup and Maintenance > Connected Servers

Click the  button.

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'
```

OMi 管理者ユーザ

BSM_ADMINISTRATOR_LOGIN_NAME 変数を使用して，OMi 管理者ユーザの名前を指定します。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，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**: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 をイベント state にマップします
- **MapOPR2SMUrgency**: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 をイベント severity にマップします
- **MapOPR2SMPriority**: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イベント・プロパティを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，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，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，OMi 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イベントの重要度

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

変更発生時にイベントの重要度に同期する 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，OMi のイベントで対応するプロパティが変更されるたびに OMi 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，OMi のイベントで対応するプロパティが変更されるたびに OMi 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。

- 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

OMi 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。OMi 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 :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 "

マッピング・テーブル: OMi イベントから BDM インシデント・プロパティ

標準の統合により，次の OMi 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。

OMi の UI（イベント）	OMi の UI（転送）	OMi 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	-	OMi 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。
-	外部 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。

OMi の UI（イベント）	OMi の UI（転送）	OMi 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
タイトル	-	title	name	brief.description	タイトル	インデント作成時に設定されます。標準的な構成 :OMi 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。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。
説明	-	description	description	action	説明	OMi イベントのタイトルと説明を組み合わせにできます。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。
ソリューション	-	solution	solution	解決	ソリューション	標準的な構成 :双方向の同期です。
ライフサイクル状態	-	state	incident_status	problem.status	ステータス	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。

OMi の UI（イベント）	OMi の UI（転送）	OMi 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
重要度	重要度	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。
優先度	優先度	priority	priority	priority.code	優先度	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。
割り当てられたユーザ	-	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。
-	割り当てられたユーザ	-	has_assigned party display_label	assignee.name	Assignee	［転送］タブのみに表示するために，OMi が Service Manager Incident の Web サービスに対して，リアルタイムでクエリを行います。OMi 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。

OMi の UI（イベント）	OMi の UI（転送）	OMi 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_ transferred_ to initiated_by	is_ recorded_ by party display_ label	opened.by	開いているユーザ	標準設定では同期しません。
割り当てられたグループ	割り当てられたグループ	assigned_ group name	has_ assigned_ group functional_ group display_ label	assignment	Assignment Group	［**転送**］タブのみに表示するために，OMi が Service Manager Incident の Web サービスに対して，リアルタイムでクエリを行います。OMi 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。
カテゴリ	-	category	category	sub_category	領域	カテゴリは Service Manager インシデントの作成時に OMi 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。

OMi の UI（イベント）	OMi の UI（転送）	OMi 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
サブカテゴリ	-	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 OMi 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。
関連 CI	-	related_ci	is_ registered_ 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 OMi によってのみ設定されます。
注釈	-	annotation_ list	activity_log description	update.action	Activity Log	OMi から Service Manager へのみ同期します。OOTB 同期が有効化されます。

OMi の UI（イベント）	OMi の UI（転送）	OMi 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
カスタム属性	-	custom_attribute_list	activity_log description	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 OMi 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。
要因	-	cause-> control_transferred_to external_id	is_caused_by -master_reference_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，OMi 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。

OMi の UI（イベント）	OMi の UI（転送）	OMi 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
Symptoms	-	symptom_list	activity_log description	update.action	Activity Log	OMi から Service Manager へのみ同期します。標準的な構成 :同期が有効化されています。
重複数	-	duplicate_count	activity_log description	update.action	Activity Log	OMi から Service Manager へのみ同期します。標準的な構成 :同期が有効化されています。

第30章: エラー処理

イベントおよびそれ以降の変更を転送する OMi イベントの同期インタフェースは，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，転送要求が再試行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。

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 「Web サービス統合」(322ページ) 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

OMi 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，「Groovy スクリプトの統合」(321ページ) を参照してください。

第VIII部: Web サービス・インタフェース

多くの Web サービスを使用して，インテグレータは外部アプリケーションから OMi 機能にアクセスできます。

最初の項では，どの Web サービスを使用する場合でも当てはまる一般的な注意点についていくつか詳述します。章のその他の部分は，利用可能な各 Web サービスについての項と，参照情報および例で構成されます。

第31章: すべての Web サービスの参照情報

本項では，使用可能な Web サービス・インタフェースのいずれかを使用する場合の考慮事項について説明します。

認証

Web サービスの使用時には，認証およびセキュリティに関して以下の点を考慮してください。

- Web サービスは，通常の OMi ユーザのみが実行できます。
- Web サービス要求の実行時に想定されるユーザ権限は，OMi ユーザ・インタフェースを使用して同じ操作を実行し，資格情報が Web サービス認証で使用されるユーザとしてログインしているときのユーザ権限と同一です。
- ネットワーク暗号化とデータの整合性は，HTTPS プロトコルを使用する場合にのみサポートされます。

次のコード・サンプルは，ユーザ名/パスワードの組み合わせを使用した基本認証の JAVA 実装を示しています。

例：

```
import org.apache.commons.codec.binary.Base64;
import javax.ws.rs.core.Response;
import javax.ws.rs.core.MediaType;
/** 要求 URL を作成する（大文字化された変数が適切な値に設定されていると想定）。 **/
String url = "http://" + WS_SERVER_HOSTNAME + ROOT_PATH + REQUEST;

byte[] encodedUserPassword =
    Base64.encodeBase64((username + ":"+ password).getBytes());
Response response =
            client.target(url).request(MediaType.APPLICATION_XML_TYPE).
            header("Authorization", "Basic " +
            new String(encodedUserPassword)).get();
```

次の方式の認証もサポートされています。

- OMi でのログインに有効なユーザ名とパスワードの組み合わせを手動で入力します。この方法は，たとえば Poster などのプラグインを使用する場合のように，Web サービスへのアドホック・アクセスで使用できます。
- ライトウェイト・シングル・サインオン（LWSSO）

- Windows 認証（WinAuth）
- 共通アクセス・カード（CAC）

エラー処理

OMi によって直接呼び出される Web サービス・エンドポイントを実装する場合，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。**次の HTTP ステータス・コードは，要求が正常に処理されたことを示します。
  - 200（OK）
  - 201（作成済み）
  - 202（受け入れ）
- **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，要求が成功するかタイムアウトするまで，1 分間に 1 回行われます（標準設定は 12 時間で，［インフラストラクチャ設定］で変更できます。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。

安全な変更トークン

Web サービスを使用して割り当てを作成，変更，または削除するには，HTTP セッションで，安全な変更トークンを X-Secure-Modify-Token という名前のヘッダとして渡す必要があります。

変更操作の保護は，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設定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（「強化されたセキュリティ保護の無効化」(332ページ)を参照）。

X-Secure-Modify-Token HTTP ヘッダの設定：

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に推奨されるアプローチでは，/opr-web/rest での Web サービスの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，サーバ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，secureModifyToken クッキーで指定されている値に設定します。

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。

  例：secureModifyToken クッキーの取得

```
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。

  例：X-Secure-Modify-Token HTTP ヘッダの設定

```
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，OMi 9.10 より前のバージョンはこの HTTP ヘッダを必要としません）。

  例：secureModifyToken クッキーの取得

```
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**

例 : 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。

  例：初期クッキーの取得

```
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。

  例 :X-Secure-Modify-Token HTTP ヘッダの設定

```
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 ヘッダを設定していると Web サービス・クライアントは，OMi バージョン 9.0x 以下と一緒にインストールされている Web サービス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。

   ［管理］ > **［セットアップと保守］** > **［インフラストラクチャ設定］**

［**オペレーション管理 - Web サービス設定**］ > ［**安全な変更**］を選択します。

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，OMi 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，OMi 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）。OMi 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 OMi 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。

Web サービス設定

［インフラストラクチャ設定］では，Web サービスのいくつかの設定を定義できます。

［**管理**］ > ［**セットアップと保守**］ > ［**インフラストラクチャ設定**］

［**オペレーション管理 - Web サービス設定**］を選択します。

これらの設定の影響の詳細については，インフラストラクチャ設定マネージャの各設定を参照してください。

第32章: Monitoring Automation Web サービス・インタフェース

REST ベースの Monitoring Automation Web サービスにより，インテグレータは，外部アプリケーションから次の Monitoring Automation 機能にアクセスできます。

- 使用可能な管理テンプレートをリストし，CI タイプまたは管理テンプレート ID 別にフィルタリングできるようにします。
- すべての管理テンプレート割り当てをリストします。
- 管理テンプレート割り当ての結果として作成されたすべてのデプロイメント・ジョブをリストします。
- 特定の CI に対するすべての管理テンプレート割り当てをリストします。
- 特定の管理テンプレートのすべての割り当てをリストします。
- 管理テンプレートの特定の割り当てのステータス情報とパラメータ情報を見つけます。
- 管理テンプレート割り当てを作成，更新，削除します。

次の各項には，Monitoring Automation Web サービスの使用法の詳細が記載されています。

- 「Monitoring Automation Web サービス・インタフェースの使用」(334ページ)
- 「Monitoring Automation Web サービス・インタフェースの参照情報」(341ページ)
- 「例」(346ページ)

Monitoring Automation Web サービス・インタフェースなど，OMi との Web サービス・インタフェースの使用に関する詳細については，「すべての Web サービスの参照情報」(325ページ)を参照してください。

Monitoring Automation Web サービス・インタフェースの使用

本項では，Monitoring Automation Web サービス・インタフェースの使用時に考慮すべきいくつかの点について説明します。

ログ記録

Web サービスが実行するすべてのアクションは，次のログ・ファイルに記録されます。

<*OMi_HOME*>/log/jboss/opr-webapp.log

<*OMi_HOME*>/log/jboss/opr-configserver.log

応答に埋め込まれるオブジェクト参照

応答に返されるオブジェクトには常に，参照オブジェクトを伴うその後の要求に使用できるいくつかの参照が含まれています。

id および <target_id>

XML タグ id および <target_id> は，それぞれオブジェクト ID と参照オブジェクトの ID を表します。

self および <target_href>

XML タグ self および <target_href> は，それぞれオブジェクト URL と参照オブジェクトの URL を表します。

type および <target_type>

XML タグ type および <target_type> は，それぞれオブジェクトの参照と参照オブジェクトの参照を表します。

<link>

XML タグ **<link>** は，詳細出力要求，作成中の割り当てのデプロイメント・ジョブのリスト要求など，考えられるナビゲーションについて追加で指定します。

例

たとえば，GET 要求への応答が，管理テンプレート・バージョンのリストである場合，リストの各管理テンプレート・バージョンは，<management_template_version_ref_version /> ノードに対応します。管理テンプレート・ノードは，次のような出力になります。

```
<management_template_version_ref version="9.20"
  type="urn:x-hp:2009:software:data_model:opr:type:reference:
      management_template_version">

  <target_id>d6f9cf24-244a-bce7-1e37-c5446c81e773</target_id>
  <target_type>urn:x-hp:2009:software:data_model:opr:type:
      management_template_version</target_type>
  <target_href>
      http://mambo8.mambo.net:80/opr-config-server/rest/ws/9.20/
      management_template_version_list/d6f9cf24-244a-bce7-1e37-c5446c81e773
   </target_href>
```

```
  <display_label>MTCIAttResolution</display_label>
  <major_version>1</major_version>
  <minor_version>0</minor_version>

</management_template_version_ref>
```

type，<target_id>，<target_type>，<target_href> タグは太字で強調表示されています。

応答でオブジェクトをループさせると，参照を使用して個々のオブジェクトの新しい要求の作成ができます。たとえば特定の管理テンプレート・バージョンのすべての割り当てを取得したりできます。

パラメータ

本項では，パラメータの説明と管理テンプレートを CI に割り当てる場合のパラメータ設定方法ついて説明します。

パラメータを扱うときの注意

Web サービス要求を実行時に MA パラメータについて次のことに注意します。

- パラメータ値は，string，numeric，password または enum 型をとります。
- 特定の XML タグは，詳細モードでのみ表示されます。たとえば，次のタグです。
  - enum パラメータの可能列挙値。
  - numeric パラメータの最小および最大値。
  - エキスパート・フラグ。
- Web サービスは，割り当てを扱う Web サービス要求に応答する場合のみパラメータを返します。管理テンプレートに含まれたパラメータのリストが必要な場合，管理テンプレートのドラフト割り当てを要求します。
- GET /assignment_list/draft/ci/*<CIID>*/management_template_version/*<MTVersionID>* 要求への応答には，ID *<CIID>* を持つ CI に関連するパラメータのみが含まれます。この CI には適合しないバージョン ID *<MTVersionID>* の管理テンプレートに含まれるパラメータは除外されます。
- 割り当て作成時，パラメータ仕様にはパラメータの ID とそのコンテキストが必要です。HP は，プログラムを使用してコンテキストを作成するよりも，コンテキストを応答から GET assignment_list/*<assignmentID>* または GET assignment_list/draft/… 要求へコピーすることをお勧めします。

  次の情報は，パラメータ・コンテキストについての補足です。

  - パラメータのコンテキストは，パラメータが定義されている設定オブジェクトを指定します。割り当て中の管理テンプレートに定義されるように，一般的にコンテキストは，管理テンプ

レートに含まれるアスペクトがどのようにサブ CI を CI にリンクするかを示すトポロジ・パスと，パラメータを含むアスペクトのバージョン ID からなります。

- 詳細モードのドラフト応答で取得できるコンテキストには，パラメータ参照の解決でユーザの役に立つパラメータを含むアスペクトのラベルと説明が含まれています。

- 割り当て作成要求からパラメータを除外すると，その標準設定値が使用されます。
- HP は，割り当て作成時にできるだけ少数のパラメータを設定することをお勧めします。パラメータ数は，割り当て中に設定され，パラメータ用の適正な標準設定値を作成することで最小化できます。管理テンプレートのパラメータ付けの詳細については，『Monitoring Automation オンライン・ヘルプ』を参照してください。
- Web サービスを使用した割り当てを作成する場合，割り当て対象の CI が既知である必要があります。したがって，Web サービスによる応答では，パラメータ値に影響するいずれの条件も解決され，条件付きではありません。例として，条件付きの値（Windows は 10，Unix は 20）を持つパラメータ値を考えます。Web サービスから割り当て情報を取得すると，値は解決され，Windows 上の CI の応答では 10 と指定され，Linux 上の CI では 20 と指定されます。
- MA ユーザ・インタフェースの調整機能を使用して変更したパラメータ値と対照的に，Web サービスを使用して設定したパラメータ値は自動割り当てルールで作成された割り当てによって上書きすることができます。このため，HP はアクティブな自動割り当てルールとともに割り当て Web サービスを使用しないことをお勧めします。
- 異なる管理テンプレート・バージョンへの割り当てをアップグレードをする場合，Web サービス割り当てで設定したすべてのパラメータ値は，新しい管理テンプレート・バージョンで指定された値で上書きされます。
- 単純なパラメータとマルチインスタンスのパラメータという 2 つのタイプのパラメータがあり，後者では XML タグ <is_instance_parameter> が true に設定されます。単純なパラメータには値がありますが，インスタンス・パラメータには代わりにインスタンス・リストがあります。リストの各インスタンスは，インスタンス値と依存パラメータのリストからなります。

  例：依存パラメータ usageThreshold（単位 %）および messageSeverity（warning や critical など多くの重要度インジケータ）を持つマルチインスタンス・パラメータ fileSystem について考えます。fileSystem は，2 つのインスタンス，C: および D: を監視するために使用されます。C: の場合，重要度 critical のメッセージを作成するしきい値は 90% で，一方 D: の場合，重要度 warning のしきい値は 95% です。

  Web サービス要求でマルチインスタンス・パラメータを使用する場合，次の点に注意します。

  - 複数のインスタンス定義が指定されていない場合，マルチインスタンス・パラメータは 1 つのインスタンスのみに作成されます。

  - マルチインスタンス・パラメータの定義で複数のインスタンスが指定される場合，各インスタンスにシーケンス番号を付与し，ポリシー・テンプレート条件が正しい順番で処理されるようにできます。
  - 依存パラメータが依存関係にあるマルチインスタンス・パラメータと関連づけられた別のインスタンスに XML タグ <ui_order> の値で定義されたより高位の優先度があれば，そのインスタンスによって，依存パラメータは上書き可能です。一般的に，最も高い優先度を持つインスタンスの ui_order 値は 0 です。

- 既存割り当てに関連した情報に対する Web サービス要求への応答には次のフラグを含むことができます。
  - is_default:パラメータが上書きされた場合，値 false に設定されます。
  - is_tuned:パラメータ値が，Monitoring Automation 調整ユーザ・インタフェースを使用して変更された場合に指定されます。
  - mandatory:管理テンプレートがパラメータを必須と定義する場合に指定されます。
  - readonly:管理テンプレートがパラメータを読み取り専用と定義する場合に指定されます。
  - expert:管理テンプレートがパラメータをエキスパート・パラメータと定義する場合に指定されます。
- パラメータの標準設定は，属性 value を使用してリテラルとして指定するか，または CI 属性の名前に設定した値を持つ属性 model_property をシンボリックに使用して，値を指定できます。
  - シンボリックな表記を使用する場合，パラメータの標準設定値は割り当てが参照する CI またはサブ CI から計算されます。
  - リテラルは，計算された値よりも優先されます。
  - HP は，プロパティの CI タイプと分解性に関してあいまいさを避けるため model_property を変更しないことをお勧めします。
- 次のパラメータ属性は変更可能です。
  - 値
  - model_property
  - マルチインスタンス・パラメータの追加インスタンスの作成。

  その他すべてのパラメータ属性は情報提供のみです。
- 割り当て作成を要求する場合，必須パラメータの値を指定する必要があります。

- 割り当て作成時間に，入力検証が行われます。検証で失敗すると（たとえば，必須パラメータが指定されていない），Web サービスは記述エラーを返します。

パラメータ値として渡されるパスワードのセキュリティ

管理テンプレート・パラメータ値で渡されるパスワードには次のようなセキュリティ上の配慮が行われます。

- パスワードはプレーン・テキストで Web サービスに渡されます。セキュリティ確保のため，Web サービス・クライアントと Web サービス・サーバ間の通信には SSL 接続を使用することをお勧めします。
- パスワードは，プレーン・テキストで渡されますが，データベースに格納されるときには自動的に暗号化されます。
- GET 要求を実行してパスワード・パラメータを取得する場合，クライアントは実際の値ではなく，プレースホルダ文字列 ***** を受信します。

  POST 要求のドラフトを表す応答で返される標準設定パスワードにもこれが適用されます。ただし，パスワード・パラメータが値 ***** で送信されると，Monitoring Automation はデータベースに格納された値を使用するので，応答を送信する前にプレースホルダを実際のパスワードに置き換える必要はありません。

推奨される POST 要求作成の手順

XML 要求および応答に使用するエンコーディングは，要求のタイプに依存しません。したがって，次のように応答を割り当て作成の POST 要求に再利用できます。

割り当てに関連する HTTP 要求および応答に使用される XML エンコーディングは同じ形式を使用します。したがって，次のように GET 応答を割り当て作成時の POST 要求に再利用できます。

1. 次のように，割り当て作成の GET 要求を実行します。
   - 少数のパラメータの値のみ既存割り当てとは異なる割り当てを作成するには，/assignment_list/*<assignmentID>* 要求から開始して，ID *<assignmentID>* を持つ割り当てのすべてのパラメータ値を含む応答を得る方法が役に立ちます。
   - ゼロから割り当てを開始するには，GET /assignment_list/draft/ci/*<CIID>*/management_template_version/*<MTVersionID>* 要求を使用します。これにより，割り当て作成要求に必要な入力のドラフトを表す応答を得ます。ドラフトには，割り当てし，標準設定値に初期化するための管理テンプレート *<CIID>* に含まれるすべてのパラメータが含まれます。
2. 作成する割り当てで変更すべきパラメータ値を上書きして応答を変更します。
3. 変更した応答を POST /assignment_list として送信し，新しい割り当てを作成します。

この方法を使用すると，自動化アルゴリズムは，変更に関係のない XML 要素に依存しません。したがって，手動で入力を作成するのと比較するとコードの頑強性が非常に向上します。

割り当てが作成されたことを確認するには，新しい割り当て，またはすでに作成されているはずのデプロイメント・ジョブの GET 要求を実行するか，OMi MA ユーザ・インタフェースを使用して割り当てを検査してください。

スキーマおよびクラス情報

XML スキーマ

次のファイルには，オブジェクト情報のエンコーディングに使用する XML スキーマが含まれています。

<OMi_HOME>/opr/api/schema/OprDataModel.xsd

JAXB 注釈クラス

次のファイルには，JAXB 注釈クラスが含まれます。

<*OMi_HOME*>/lib/opr-external-api.jar

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。利用でき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。

詳細および最小モード

Web サービス要求の入力，出力データにおける詳細レベルは，verbose または minimal モードを指定して決定できます。

- minimal モードでは，入力および出力データには，オブジェクトの定義に必要な最小セットの XML タグが含まれます。最小セットは，verbose モードで使用される XML タグのセットの真のサブセットです。
- 標準設定では，minimal モードです。
- verbose モードを使用するには，Web サービスを呼び出すときに URL の末尾に ?view=verbose を加えてください。
- minimal モードを使用する場合，XML 応答には XML タグ <link rel="verbose"> が含まれます。このタグの値は，呼び出し時の応答ファイルの詳細バージョンを指定する URL です。

Monitoring Automation Web サービス・インタフェースの参照情報

本項では，Monitoring Automation（MA）Web サービス・インタフェースの参照情報を提供します。

要求構文

Monitoring Automation Web サービス・インタフェースには，次の構文の URL を使用してアクセスします。

```
http[s]://<serviceURL>/<request>
```

<*serviceURL*>	Monitoring Automation Web サービスの URL : <*Server*>:<*port*>/opr-config-server/rest/ws/<*version*> <*Server*> は，OMi サーバの名前，<*version*> は Web サービス・インタフェースのバージョン（例 : 9.20）です。
<*request*>	サービス要求。可能なサービス要求の全一覧および構文については，「ヘッダ構文」(343ページ)および「要求の参照情報」(344ページ)を参照してください。

注: すべての要求では認証が必要とされ，POST 要求と DELETE 要求では安全な変更トークンを指定したヘッダが必要とされます。詳細については，「ヘッダ構文」(343ページ)を参照してください。

ヘッダ構文

Monitoring Automation Web サービス・インタフェースでは，次のヘッダを使用します。

ヘッダ名	ヘッダ値
承認	スペースが 1 つ続く文字列 Basic，および認証で使用されるエンコードされたユーザ名とパスワード。 RESTEasy クライアントで org.apache.commons.codec.binary.Base64 コーデックを使用する JAVA コードの例 : import org.apache.commons.codec.binary.Base64; byte[] encodedUserPassword = Base64.encodeBase64 ((“myUserName" + ":"+"myPW").getBytes()); response = client.target(URL). header("Authorization", "Basic " + encodedUserPassword). get();

X-Secure-Modify-Token	Set-Cookie で設定されている安全な変更トークンの値。
	
	トークンを抽出して渡す JAVA コードの例 :
	
	Object header = response.getHeaders().get("Set-Cookie");
	for(String cookie :(List<String>)header)
	{
	cookie = cookie.substring(0, cookie.indexOf(";"));
	String cookieName = cookie.substring(
	0, cookie.indexOf("="));
	String cookieValue = cookie.substring(
	cookie.indexOf("=") + 1, cookie.length());
	if(cookieName.equalsIgnoreCase("secureModifyToken"))
	break;
	}
	response = client.target(URL). header("Authorization", "Basic " +
	encodedUserPassword). header(X-Secure-Modify-Token",cookieValue). post(xml);

要求の参照情報

Monitoring Automation（MA）Web サービス・インタフェースでは，次の要求をサポートしています。

要求	要求タイプ	MA Web サービスのアクション
/assignment_list [/*<assignmentID>*]	GET	**/*<assignmentID>* が省略されている場合 :**データベースにある現在のすべての割り当てをリストします。 **/*<assignmentID>* が指定されている場合 :**ID が *<assignmentID>* の割り当てを取得します。
	POST	送信した XML データで指定されている管理テンプレート割り当てを作成します。 **注 :** • /*<assignmentID>* は省略する必要があります。 • この操作には，次のライセンスが必要です。**HP Monitoring Automation for Composite アプリケーション**。
	DELETE	ID が *<assignmentID>* の管理テンプレート割り当てを削除します。 **注 :** • /*<assignmentID>* を指定する必要があります。 • Web サービスを使用して削除できるのは，管理テンプレート割り当てのみです。
/assignment_list/ci/*<CIID>*	GET	ID が *<CIID>* の CI に割り当てられているすべての管理テンプレートをリストします。

要求	要求タイプ	MA Web サービスのアクション
/assignment_list/draft/ci/*<CIID>*/management_template_version/{*<MTID>* | *<MTVersionID>*}	GET	割り当ての作成のドラフトを取得します。 • *<MTID>* が使用されている場合は，指定した管理テンプレートの最新バージョンが割り当てられます。 • *<MTVersionID>* が使用されている場合は，指定した管理テンプレート・バージョンが割り当てられます。 応答に対して必要な変更を加え，/assignment_list 要求を使用してその応答を POST し，割り当てを作成することができます。 この手順の詳細については，「すべての Web サービスの参照情報」(325ページ)および「例」(346ページ)を参照してください。
/assignment_list/management_template/*<MTID>*	GET	ID が *<MTVersionID>* の管理テンプレートの割り当てをすべてリストします。
/assignment_list/management_template_version/*<MTVersionID>*	GET	バージョン ID が *<MTVersionID>* の管理テンプレートの割り当てをすべてリストします。
/deployment_job_list[/{*<DeploymentJobID>* | assignment/*<AssignmentID>*}]	GET	***<DeploymentJobID>* と *assignment/<AssignmentID>* のどちらも指定されていない場合 :**データベースにあるすべてのデプロイメント・ジョブをリストします。 ***<DeploymentJobID>* が指定されている場合 :**ID が *<DeploymentJobID>* のデプロイメント・ジョブを取得します。 ***<assignment/AssignmentID>* が指定されている場合 :**ID が *<AssignmentID>* の割り当ての結果として作成されたすべてのデプロイメント・ジョブをリストします。

要求	要求タイプ	MA Web サービスのアクション
/management_template_list[/{*<MTID>* | /ci_type/*<CIType>*}]	GET	***<MTID>* が省略されている場合 :**データベースにあるすべての管理テンプレートをリストします。 ***<MTID>* が指定されている場合 :**ID が *<MTID>* の管理テンプレートのすべてのバージョンをリストします。 **/ci_type/<CIType> が指定されている場合 :**CI タイプが *<CIType>* の CI に割り当て可能なすべての管理テンプレートをリストします。
/management_template_version_list[/*<MTVersionID>*]	GET	***<MTVersionID>* が省略されている場合 :**データベースにあるすべての管理テンプレートのすべてのバージョンをリストします。 ***<MTVersionID>* が指定されている場合 :**バージョン ID が *<MTVersionID>* の管理テンプレートを取得します。

例

本項では，Monitoring Automation（MA）Web サービス・インタフェースの使用例をいくつか示します。

各使用例では，ワークフローが順に示されています。これらの使用例の多くでは，次のディレクトリに格納されているコード・サンプルを使用しています。

<*OMi_HOME*>/opr/examples/assignment-ws-client
<*OMi_HOME*>/opr/examples/mgmt-template-ws-client

前提

それぞれの例は，次の前提に基づいています。

- CI は，OMi によって監視されるトポロジの一部であり，ネットワークを介して接続することができる。
- 認証および安全な変更トークンが必要に応じて正しく指定されている。詳細については，「すべての Web サービスの参照情報」(325ページ)を参照してください。
- 認証で使用する OMi アカウントには，要求された操作を実行するために十分な権限が付与されている。

- カスタマには，要求された操作を実行するために必要な権限が付与されている。ライセンス要件の詳細については，「Monitoring Automation Web サービス・インタフェースの参照情報」(341 ページ)を参照してください。

シナリオ 1 : 新規 CI を監視する

推奨されるワークフロー :

1. 次の Web サービス要求を発行して，CI タイプが myType の CI に割り当て可能なすべての管理テンプレートをリストします。

   GET /management_template_list/ci_type/myType

2. 応答には管理テンプレート・バージョンのリストが含まれています。最新の管理テンプレート・バージョンは，latestMTV として示されます。

   a. 次の Web サービス要求を発行して，管理テンプレート・バージョンの割り当てをシミュレートします。

      GET /assignment_list/draft/ci/myCI/management_template_version/latestMTV

   b. 応答で，変更されるすべてのパラメータ値を更新します。

   c. 次の要求を使用して応答を送信し，割り当てを作成します。

      POST assignment_list

シナリオ 2 : CI に対するすべての割り当てを削除する

推奨されるワークフロー :

1. 次の Web サービス要求を発行して，ID が myCI の CI に対するすべての割り当てをリストします。

   GET /assignment_list/ci/myCI

2. 応答には割り当てのリストが含まれています。myAssg として返される割り当てごとに，次の要求を発行して削除します。

   DELETE /assignment_list/myAssg

   注: CI が削除されると，Monitoring Automation がその CI に対するすべての割り当てを自動的に削除するため，これらの手順を実行する必要はありません。

シナリオ 3 : CI の監視を一時的に無効にする

推奨されるワークフロー :

1. 次の Web サービス要求を発行して，ID が myCI の CI に対するすべての割り当てをリストします。

   GET /assignment_list/ci/myCI

2. 応答には割り当てのリストが含まれています。返される割り当てごとに is_enabled フラグを false に設定することによって，応答に変更を加えます。

3. 次の要求を使用して応答を送信し，割り当てを更新します。

   POST assignment_list

割り当てを有効にするには，このワークフローを繰り返します。ただし，is_enabled フラグは true に設定します。

シナリオ 4 : 既存の割り当てのパラメータ値を変更する

推奨されるワークフロー :

1. 次の Web サービス要求を発行して，ID が myAssg の割り当てを取得します。

   GET /assignment_list/myAssg

2. 応答には，パラメータ値ブロックが含まれています。パラメータ値ブロックで，値が myParm のノード parameter_value > parameter > display_label を検索することにより，myParm として示される必要なパラメータを検索します。このノードにより，ユーザ・インタフェースで myParm と呼ばれるパラメータを定義します。

3. 必要に応じてノード value の内容を変更することにより，応答に変更を加えます。

4. 次の要求を使用して応答を送信し，割り当てを更新します。

   POST assignment_list

シナリオ 5 : 新しい管理テンプレート・バージョンに対する割り当てを更新する

推奨されるワークフロー :

1. 次の Web サービス要求を発行して，ID が myMTV の管理テンプレート・バージョンのすべての割り当てをリストします。

   GET /assignment_list/management_template_version/myMTV

2. 応答には割り当てのリストが含まれています。返される割り当てごとに，myAssg として示されます。

   a. 応答から，管理テンプレートが割り当てられている CI の ID を判別します。

   b. 応答から，割り当てに含まれている各パラメータの値を判別します。新しい管理テンプレート・バージョンで同じ標準設定値 myMTV を使用する場合は，is_default フラグが false に設定されている変更されたパラメータについてのみ値を判別すれば十分です。

   c. 次の Web サービス要求を発行して，新しい管理テンプレート・バージョン（バージョン ID が newMTV として示される）の割り当てをシミュレートします。

      GET /assignment_list/draft/ci/myCI/management_template_version/newMTV

   d. myMTV の割り当てから取得したパラメータ値を，newMTV のドラフト応答に含めます。

   e. 次の要求を使用して変更したドラフト応答を送信し，割り当てを作成します。

      POST assignment_list

   MA は，割り当てる前に古いバージョンの管理テンプレートの既存の割り当てをすべて削除することによって，常に 1 種類のバージョンの管理テンプレートのみが特定の CI に割り当てられるように自動的に処理します。

シナリオ 6 : 必須の割り当てパラメータを判別する

推奨されるワークフロー :

1. 次の Web サービス要求を発行して，管理テンプレート・バージョンの割り当てをシミュレートします。

   GET /assignment_list/draft/ci/myCI/management_template_version/latestMTV

2. 応答には，パラメータ値ブロックが含まれています。各パラメータについて，必要なフラグの値を判別します。

3. required フラグが指定されていて値が true になっている場合，そのパラメータは必須であり，ドラフトに基づいて割り当てを作成するときに指定する必要があります。

シナリオ 7 : デプロイメントに成功したかどうかを確認する

推奨されるワークフロー :

1. 次の要求を発行して，ID が myAssg の割り当てを作成した結果として開始されたデプロイメント・ジョブがあるかどうかチェックします。

   /deployment_job_list/assignment/myAssg

2. 応答について次の点を調べます。

   - その応答にデプロイメント・ジョブが含まれていない場合，割り当ては正常にデプロイされました。
   - 応答に failed 状態のデプロイメント・ジョブが含まれている場合，デプロイメントは失敗しました。ジョブの内容に示されている問題をすべて修正し，割り当てを再作成してください。
   - 応答に running 状態のデプロイメント・ジョブが含まれている場合，デプロイメントは終了していません。デプロイメントが終了するために必要とされる妥当な時間が経過するまで待ち，running 状態のすべてのジョブが完了するまでワークフローを繰り返してください。

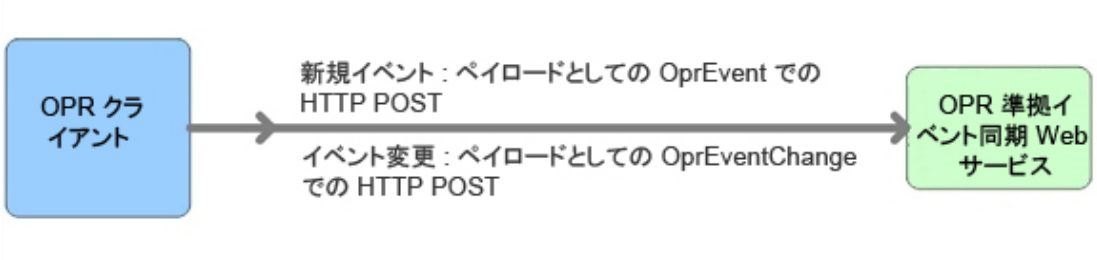
第33章: イベント同期 Web サービス・インタフェースの参照情報

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，OMi 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 OMi 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，OMi 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 OMi 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

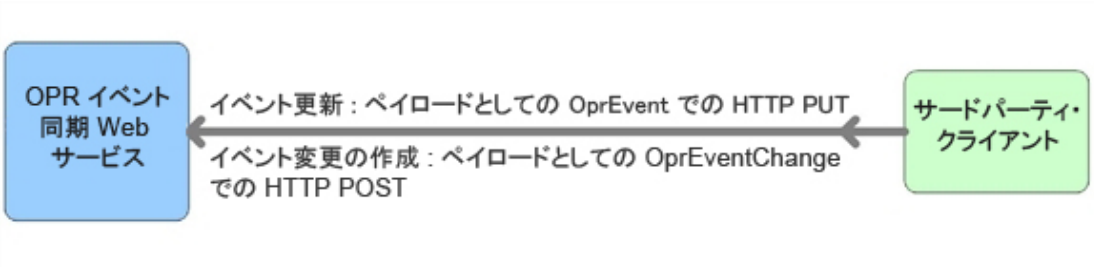
サードパーティ・アプリケーションに対して ping 要求を送信できます。サードパーティ・アプリケーションに ping 要求を送信する場合，次のデータが関連します。

- 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://<*server.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://<*server.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。

「イベント更新 :ログファイル・アダプタの例」(359ページ)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://<*server.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。

「イベント変更の作成 :ログファイル・アダプタの例」(362ページ)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://<*server.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://<*server.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://<*server.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。

「event_list に対するイベント更新の例」(370ページ)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://<*server.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。

「イベントの送信の例」(371ページ)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://<*server.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。

「event_change_list のイベント変更作成の例」(372ページ)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 OMi のユーザがアクセス可能なあらゆるイベントについて，OMi 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（「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」(187ページ)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，OMi 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	可	可	
assigned_user	可		NULL に設定
assigned_group	可		NULL に設定
cause	可		NULL に設定
custom attribute	可	可	
description	可	可	NULL に設定
priority	可		
severity	可		
solution	可		NULL に設定
state	可		
title	可		NULL に設定
control_transferred_to	可		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（「1 つの呼び出しによる複数のプロパティの変更」(361ページ)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，「REST Web サービス・コマンドライン・ユーティリティ」(211ページ)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://<*server.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。

<OMi_HOME>/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。

<OMi_HOME>/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。

例 :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>
```

例 :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。

例 :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>
```

例 :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。

例 :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（「1 つの呼び出しによる複数のプロパティの変更」(370ページ)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://<*server.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://<*server.example.com*>/opr-gateway/rest/9.10/synchronization/event_change/<*event 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。

例 :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>
```

例 :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。

例 :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>
```

例 :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。

例 :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>
```

例 :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。

例 :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>
```

例 :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。

例 :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。

例 :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>
```

重要度

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。

例 :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。

例 :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>
```

例 :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。

例 :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>
```

例 :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。

例 :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://<*server.example.com*>/opr-gateway/rest/9.10/synchronization/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://<*server.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>
```

第IX部: Groovy スクリプト

OMi では，スクリプトベースのカスタマイズがサポートされます。スクリプトを使用すると，必要な機能が OMi で使用できない，または標準の OMi ツールと設定オプションを使用しても追加できない場合に，OMi プロセス・フローに機能を追加できます。

本項の構成

本項には，Groovy スクリプトの追加場所と追加方法が説明されています。本項には，次の内容が含まれます。

- いくつかの異なるエリアでカスタマイズが可能です。各エリアにスクリプトを開発，デプロイする方法については，「スクリプトの開発およびデプロイメント」(384ページ)の項に詳細が解説されています。
- いずれのスクリプトにも適用可能な参照情報は，「参照情報」(410ページ)の項に記載されています。
- スクリプト開発時のベスト・プラクティスに関するガイダンスは，「ベストプラクティス」(379ページ)の項に記載されています。

カスタマイズ・スクリプトの概要

次の側面がカスタマイズ・スクリプトの開発を成功させるためには重要です。

- **スクリプト実行トリガ**

  トリガによりスクリプトは実行されます。たとえば，スクリプトはイベントの発生とともに実行されるように設定できます。トリガは，★「スクリプトの開発およびデプロイメント」(384ページ)のさまざまなタイプのカスタマイズに関する項において詳述されています。★の「プロセス・フロー」見出し下のさまざまなタイプのカスタマイズに関する項において詳述されています。

- **スクリプト実行プロセス場所**

  スクリプトが実行されるときのプロセス・フロー内の場所です。たとえば，イベントベースのスクリプトは，イベントの処理前または後に実行されるように設定できます。利用可能な場所については，★「スクリプトの開発およびデプロイメント」(384ページ)のさまざまなタイプのカスタマイズに関する項において詳述されています。★の「プロセス・フロー」見出し下のさまざまなタイプのカスタマイズに関する項において詳述されています。

- **プログラミング言語**

  OMi では，Groovy スクリプトがサポートされています。Groovy は，Java プラットフォームで動作する動的なオブジェクト指向のプログラミング言語です。Java プラットフォームのスクリプト言語として使用される場合，Groovy は Java 仮想マシン（JVM）バイトコードに動的にコンパイルされるので，外部 Java コードやライブラリと互換性があります。Groovy は Java に似ており，多くの Java コードは構文上 Groovy で有効です。Groovy の構文の詳細は，このガイドの範囲外ですが，参考になる多くのソースがインターネット上にあります。まず初めは，Wikipedia のGroovy (programming language) を検索し，記事に書かれた参照リストを使用するとよいでしょう。公式 Web サイトは次の URL です。

  http://groovy.codehaus.org/

  インストール，デプロイメント，Groovy コンソールを使用したデバッグに関する OMi 専用の情報，Groovy コンソールの入手に関する詳細については「Groovy コンソール」(410ページ)を参照してください。

- **スクリプト形式**

  スクリプトを追加可能なさまざまなプロセスの場所で，OMi プロシージャがスクリプトを呼び出し，正常にその実行ができるように，スクリプトは特定の形式に従う必要があります。テンプレート・スクリプトと使用例がインストール時に入手できます。さまざまなタイプのカスタマイズで使用される形式については，★「スクリプトの開発およびデプロイメント」(384ページ)のさまざまなタイプのカスタマイズに関する項において詳述されています。★の「形式」見出し下のさまざまなタイプのカスタマイズに関する項において詳述されています。

- **情報関数**

  スクリプトを使用して，管理された構成アイテム（CI）に関する情報を，標準情報セットに追加する場合が多くあります。追加すべき情報は，様々なサービスやプロセスから取得する必要があります。該当するアプリケーション・プログラミング・インタフェース（API）から関数を使用します。たとえば，イベント情報を強化する場合，「RTSM Java API」から関数を使用し，CI から情報を取得できます。利用可能な OMi および外部の API のリストを「利用可能な API」(414ページ)に示します。関数の使用例については，★「スクリプトの開発およびデプロイメント」(384ページ)のさまざまなタイプのカスタマイズに関する項において詳述されています。★の「例」見出し下のさまざまなタイプのカスタマイズに関する項において詳述されています。

- **ベスト・プログラミング・プラクティス**

  ベスト・プラクティスおよび避けるべきことの詳細は，「ベストプラクティス」(379ページ)に述べられています。従うべき特定のベスト・プラクティスについては，★「スクリプトの開発およびデプロイメント」(384ページ)のさまざまなタイプのカスタマイズに関する項において詳述されています。★の「...固有のベスト・プラクティス」見出し下のさまざまなタイプのカスタマイズに関する項において詳述されています。

- **スクリプト管理**

スクリプトは，OMi インスタンスが実行されているサーバにあり，OMi コンソールを使用して該当プロセス用に設定されます。スクリプトの設定方法は，それを使用する場所により異なります。スクリプトを含める，アクティブ化するために必要な手順は，★「スクリプトの開発およびデプロイメント」(384ページ)のさまざまなタイプのカスタマイズに関する項において詳述されています。★の「スクリプト管理」見出し下のさまざまなタイプのカスタマイズに関する項において詳述されています。

第34章: ベストプラクティス

本項は，高品質なスクリプトの開発に役立つヒントについて説明します。

パフォーマンスの改善

カスタマイズのスクリプトは処理フローの一部として実行されます。カスタマイズは処理の実行を低速にする原因になります。総処理時間に対して増加した時間が短かければ問題になりませんが，総処理時間が後続の処理の実行間隔の大部分を占めるまで増加すると，イベント同期やトポロジ同期などのすべての OMi プロセスに悪影響を及ぼす可能性や中断する可能性もあります。したがって，カスタム・スクリプトの実行にかかる絶対時間をできるだけ短くすることが重要です。

データのキャッシュによるパフォーマンスの改善

パフォーマンスの改善において，取得に時間のかかるデータをキャッシュする方法は有効です。同期プロセスに，RTSM データ，ユーザ・データベースのデータ，ファイルのデータなどの外部リソースからデータを取得する場合の例を示します。

次は，キャッシュに関する操作を示しています。

1. 外部リソースに接続します。
2. リソースからデータを読み込み，メモリのデータ構造に保存します。通常，ここでのアクセス時間のほうがより短くなります。メモリ内のデータをキャッシュ・データと呼びます。
3. カスタム・プロセスでデータを使用します。
4. 外部リソースから切断します。
5. キャッシュ・データをメモリから削除します。

通常，外部リソースへの接続と読み込みの手順は最も時間がかかります。

キャッシュを実装する場合は，次の点を考慮してください。

- 解決するより回避するほうが望ましいため，外部リソースへの依存度を最小限にしてください。
  - 外部データの使用を検討するとき，本当に必要かどうかを検討してください。
  - 内部リソースのデータを利用可能または誘導可能な場合，外部リソースを使用しないでください。
  - 外部リソースのデータを利用可能または誘導可能な場合，最小限の数のリソースを使用してください。

  - さまざまな API のデータを利用可能または誘導可能な場合，パフォーマンスが最速の API を使用してください。

- 再利用しそうなデータのみをキャッシュします。CI データは再利用する可能性があるため，多くの場合，キャッシュします。ただし，利用可能なすべての CI データを常にキャッシュすることはお勧めしません。特定のタイプの CI に関連付けられたイベントのみをスクリプトが使用する場合，関連するイベントのみをキャッシュするためにフィルタを適用できます。再読み込みされるデータのみをキャッシュすることとコードを複雑にすることは，バランスを取るようにしてください。コードを複雑にすると，多くの if および case 条件を評価する必要があるために，プロセスのパフォーマンスを低下させる可能性があります。

- 外部リソースへの接続が必要になるのは 1 回のみです。スクリプト全体で使用するデータを格納するリソースへの接続に最適な場所は，スクリプト・クラスの init() メソッドです。切断は，destroy() メソッドで実行します。process() メソッド内のコードからリソースに直接アクセスできます。

- さまざまな場所で多様なデータが必要になるため，データの読み込みは通常コードの使用中に実行されます。ただし，同じデータを繰り返し再利用する場合，init() メソッドを使用して，接続，ローカルの配列へのデータの読み込み，切断をすぐに行うことができます。

- リソースをオープンし，メモリにデータを格納するために，システム・リソースが使用されます。リソースの使用状況を最適化するために，process() メソッドで最初にデータにアクセスするときに接続し，読み込んだ後に切断することもできます。これによって，1 回接続する場合にかぎり，同じ程度の速度になります。ただし，いずれにしても，システム・リソースの使用を最小限にするよりも，絶対時間でスクリプト・パフォーマンスを最適化するほうが望ましいです。

- デバッグ終了後，不要なオーバーヘッドを回避するために，実運用環境に必要のないすべてのログ用のコードをスクリプトから削除します。

低速の API および関数を使用した回避

- パフォーマンスが重要なコードに，低速なパフォーマンスが特徴として知られている API を使用しないでください。特に注意が必要なのは，イベント・パイプラインにカスタム・スクリプトを挿入したときの API のパフォーマンスです。

- JAVA API を使用して RTSM にアクセスします。JAVA API は，SOAP ベースの Web サービス API などのその他の API よりもパフォーマンスが優れています。

- System.exit() などのシェル関数の使用は許可されていません。シェル関数を呼び出すと，Groovy スクリプトを実行する JVM が停止されて再起動されます。

- 時間がかかる処理は，可能なかぎりバックグラウンドのスレッドで実行します。

コードの品質の改善

コードの品質は，コードを使用した結果のスクリプトの統合および保守の難易度に直接影響を与えます。

スクリプトのデバッグによる潜在的な問題の抑制

問題のあるスクリプトの実行により，コードを実行している場合であっても，パフォーマンスの問題が起こる場合があります。実運用環境で安全に使用できる堅牢なスクリプトを作成するには，徹底的なテストとデバッグが必要です。

- スクリプトの検証およびテスト中に Groovy コンソールを使用します（「Groovy コンソール」(410ページ)を参照）。
- スクリプトのデバッグ時，コードの動作を追跡できるログ用ステートメントを多数挿入します。HP は，デバッグ情報のログの記録を debug() メソッドにカプセル化することをお勧めしています。これにより，実運用環境でのログの記録の削除や無効化が容易になります（「パフォーマンスの改善」(379ページ)を参照）。例外として，init() および destroy() メソッドの呼び出しが挙げられます。HP は，デバッグ環境および実運用環境の両方でのスクリプトのロードとアンロードを正常に追跡できるように，info() メソッドを使用してログを記録することをお勧めしています。
- スクリプトの今後の保守を容易にするために，コメントと詳細な変数名を使用してください。
- 例外で実行時エラーが発生した場合，プロセスが中断される可能性があります。try や catch のブロックを使用して例外を取得し，例外ハンドラで明確に終了するようにします。
- init() メソッドで接続するとき，リソースにアクセスする前にデータベース処理の妥当性を確認します。スクリプトの外部の理由でリソースが切断された場合，この例外の処理で再接続して接続を適切に回復できます。
- オブジェクトが null を評価するときに NullPointerException を回避するために，オブジェクトのプロパティを参照するときは ? を使用します。? を使用してプロパティを参照すると，例外をスローするのではなく，参照を含む式全体が null を評価します。

  例：次のコード・スニペットは assignedUser オブジェクトが null を評価する場合に現在も機能します。
  ```
  if (event.assignedUser?.userName) {...}
  ```
- 可能なかぎりスクリプトの透明性を高め，今後のスクリプトの保守を容易にするために，`def` ステートメントの使用を回避します。HP は，def ステートメントを使用するのではなく，実際のオブジェクト・タイプを宣言することをことをお勧めしています。これにより，例に挙げたオブジェクトのプロパティおよびメソッドを容易に見つけることができます。透明性の高いスクリプ

トのメリットにはほかに，デバッグの容易さがあります。ただし，これは def ステートメントの使用が合理的な場合に関してです。例として，前のバージョンで利用できないクラス・タイプの引数を 1 つ持つ関数を使用したスクリプトの下位互換性を検討します。引数がタイプ def として宣言されている場合，クラスがそのバージョンで定義されていなくても，現在も前のバージョンでスクリプトを読み込むことができます。この場合，問題になっている引数の実際のオブジェクト・タイプを指定するコメントを追加します。

- 内部スクリプトのエラー機能を使用して，実行時の問題の発生によってスクリプトが起こすプロセスの中断を防止します。スクリプト・クラスが getInternalScriptError() という名前のパブリック・メソッドを使用していて，メソッドが 10 以上の値を返す場合，スクリプトはスクリプトを実行するプラットフォーム・サービスによってスキップされます。この機能を使用するには，次のコードを挿入します。

  ```
  int internalScriptError = 0;
  def int getInternalScriptError() { return internalScriptError; }
  def setInternalScriptError() { internalScriptError++; }
  ```

  このコードは，スクリプト・エラー・レベルを保持するメンバ変数 internalScriptError を 0 に初期化します。プログラマは，エラーが発生するたびに setInternalScriptError() メソッドを呼び出してエラー・レベルを高くできます。たとえば，setInternalScriptError() メソッドが呼び出される場合，データベースへの接続に失敗するたびに，スクリプトは 10 回試行した後にスキップされ，処理時間は使用されなくなります。

コード作成の簡易化

次のヒントは OMi の Groovy スクリプトの作成を可能なかぎり容易にします。

外部編集ツールの使用

プラットフォームでスクリプトを設定するときに使用される OMi の管理画面は，スクリプトの管理に役立ちますが，スクリプトの編集，保守，テスト，デバッグには十分ではありません。より万能な編集環境を作成するには，外部編集ツールを使用する必要があります。

- 行番号およびテキスト検索の機能が，コード行を参照するログ・ファイルのマッチングに役立ちます。これらが OMi コンソールに含まれるスクリプト・フォームでまったく利用できない場合，外部エディタがスクリプト開発中に役立ちます。

- 多くのスクリプトでは，Notepad++，vi，GNU Emacs などのシンプルなツールで十分です。
  - Notepad++ は，自動インデント・プラグインのオプションだけでなく，行番号と Java 構文の強調表示を特色とする Notepad を改良したフリーウェアです。Notepad++ は，http://notepad-plus-plus.org/download/ から取得できます。

  - vi は，UNIX 用に開発されたビジュアル・テキスト・エディタであり，UNIX システムにインストールされています。vi の無料のオープン・ソース・ソフトウェアを実装できます。詳細については，http://en.wikipedia.org/wiki/Vi を参照してください。

  - GNU Emacs は，拡張とカスタマイズが可能なテキスト・エディタであり，構文の強調表示，Unicode への完全対応，多くの任意の拡張機能を特色とします。GNU Emacs は，http://www.gnu.org/software/emacs/ から取得できます。

- 構文チェックが必要なさらに複雑なスクリプトの場合は，Groovy コンソールを使用します。Groovy コンソールのインストールおよび使用の詳細については，「Groovy コンソール」(410ページ)を参照してください。

- 大きなイベントでは，Service Manager アダプタ・スクリプトなどのスクリプトの転送は，Eclipse や IntelliJ などの，より包括的なスクリプト・エディタへ変更するときに役立つことがあります。

スクリプト・テンプレートの使用

スクリプト・テンプレートとサンプルを，スクリプトが保存されているディレクトリから取得できます。新しく作成するのではなく，テンプレートから新しいスクリプトを開始すると，スクリプト形式のルールに従うのが容易になり，対応忘れも少なくなります。

コピーと貼り付けの注意

電子メール，PDF ファイル，Microsoft Word 文書からコピーして貼り付ける場合は注意してください。

- コピーと貼り付けを使用すると，余分な改行や Groovy エンジンで異なる扱いを受ける改行のために，コードが無効になる可能性があります。Groovy エンジンはステートメントの終了にセミコロンを必要としません。また，改行をステートメントの終了として処理することもあります。意図しない改行によって，コードが意図しない動作になる可能性があります。

- コードをコピーして貼り付ける場合，行ごとにそのコードを検査して 2 行にわたるステートメントの分割を再結合します。これは，Groovy エンジンが誤って 2 行の別のステートメントとして解釈する可能性があるためです。

第35章: スクリプトの開発およびデプロイメント

本項では，トリガ，プロセス・フロー，スクリプト管理，ベスト・プラクティス，スクリプトを使用できる領域の例について説明します。次のタイプのスクリプトを対象とします。

イベント処理インタフェース・スクリプト

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。EPI プロセスにとって外部のデータを使用するイベントの変更および強化などを実行できます。たとえば，外部の SQL データベースからデータを追加することができます。

注: EPI スクリプトはイベントの転送には使用しません。イベント転送のためのスクリプトの詳細については，「イベント転送スクリプト」(400ページ)を参照してください。

プロセス・フロー

バージョン 9.2x のプロセス・フローは，次のとおりです。

- hpbsm_opr-scripting-host で実行されます。
- ログ情報が次のファイルに書き込まれます。

  <*OMi_HOME*>/log/opr-scripting-host/opr-scripting-host.log

バージョン 0.0x および 9.1x のプロセス・フローは，次のとおりです。

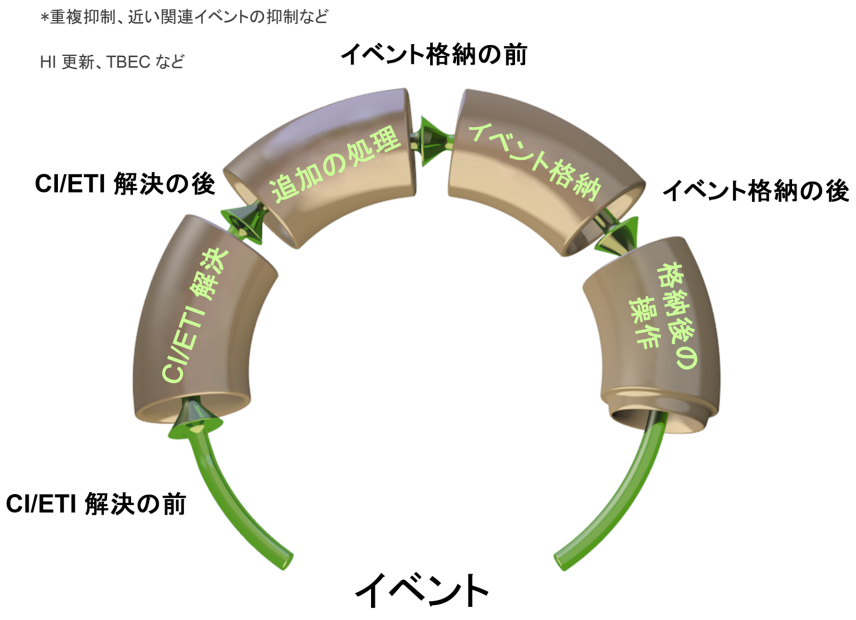
- hpbsm_opr-epi-server で実行されます。
- ログ情報が次のファイルに書き込まれます。

  <*OMi_HOME*>/log/opr-epi-server/opr-epi-server.log

EPI スクリプトは，スクリプティング・ホスト・プロセスが実行されるたびに自動的にトリガされます。OMi コンソールでスクリプトを設定するときに，スクリプトが実行されるイベント・パイプライン・プロセス・フローでの場所を設定できます。下図に示すように，いくつかの場所を選択可能です。

1. Before CI/ETI Resolution:スクリプトは，イベントがイベント・パイプラインに入るとき，CI/ETI 解決の前に実行されます。
2. After CI/ETI Resolution:スクリプトは，CI/ETI 解決の直後に実行されます。
3. Before Storing Events:スクリプトは，すべてのイベント処理の実行が完了後，イベントがデータベースに格納される前に

   実行されます。
4. After Storing Events:スクリプトは，イベントがデータベースに格納された後に実行されます。この場所に設定されるすべてのスクリプトは，読み取り専用モードで実行されます。



スクリプト管理

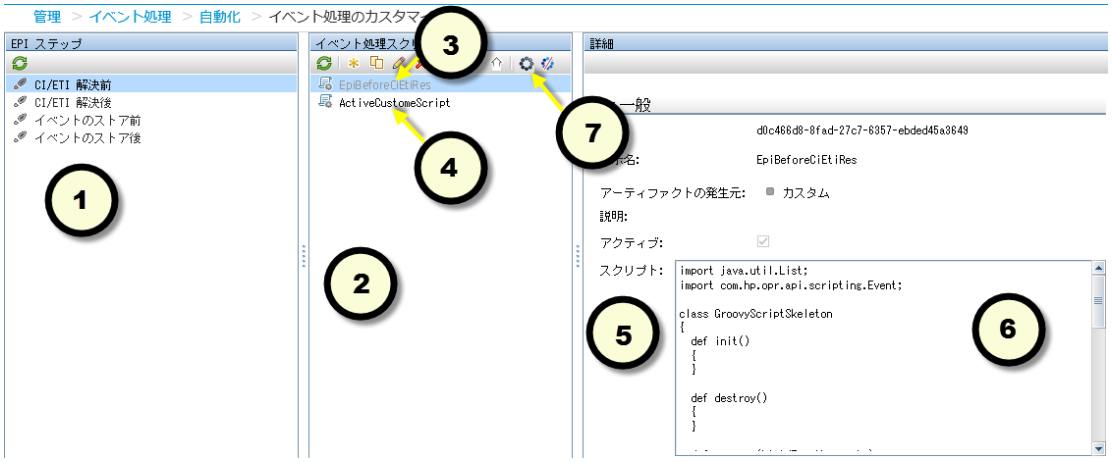
スクリプトは，データベースに格納されており，OMi コンソールからアクセスできます。

イベント転送スクリプトを設定し，アクティブ化するには，次の手順を実行します。

1. OMi で，次のようにイベント処理のカスタマイズに移動します。

   ［管理］ > ［イベント処理］ > ［自動化］ > ［イベント処理のカスタマイズ］

   イベント処理のカスタマイズ画面を下図に示します。



画面には，次の 3 つの表示枠があります。

a. *EPI ステップ*①

   この表示枠には，スクリプト化されたアクションをイベント処理に追加できるイベント・パイプライン内の可能性のある場所が一覧表示されます。一覧表示された場所は，前述の図に示した場所に対応します。

b. *イベント処理スクリプト*②

   この表示枠には，選択した場所の設定が完了したすべてのスクリプトが一覧表示されます。［EPI ステップ］表示枠で場所を選択すると，スクリプトのリストがグレーまたは黒で表示されます。グレーのスクリプトは非アクティブ，黒のスクリプトはアクティブです。③④

c. *詳細*

   この表示枠には，スクリプト・コードのエディタ・フィールドを含む実際のスクリプトのプロパティが表示されます。③

標準設定では，いくつかのサンプル・スクリプトとプレースホルダがデータベースに用意されています。新しいスクリプトを書くときには，必ずすべての必要なメソッドが利用可能になるように，開始点として，まずサンプル・クラスを使用することを HP は推奨します。

2. スクリプトの作成完了後に，スクリプトを選択して［**アクティブ化**］アイコン をクリックするか，またはスクリプトを右クリックして［**項目のアクティブ化**］を選択して，アクティブ化します。これで，イベント・パイプラインが処理されるたびにスクリプトが実行されます。

形式

EPI スクリプトを書く場合，次のルールを適用する必要があります。

1. スクリプトを，クラスに含める必要があります。

   a. そのクラスに対し独自の名前を選択できます。設定を行ったクラス名は OMi 管理コンソールの［イベント処理スクリプト］表示枠に追加されます。

   b. このクラスは，プロパティやメソッドを継承しません。

   次は，クラス宣言の例です。
   ```
   class EpiBeforeStoringEvents{ ... }
   ```

2. スクリプト・クラスにパブリックな init() および destroy() メソッドが含まれている場合，スクリプト・クラスがインスタンス化および破棄されるときに，これらのメソッドがそれぞれ呼び出されます。これらのメソッドを使用して，一度だけ呼び出されるカスタマイズした初期化およびクリーンアップ・アクションを実装できます（「ベストプラクティス」(379ページ)も参照）。

3. スクリプト・クラスには，process(eventList) という名前のパブリック・メソッドが必要です。このメソッドは，スクリプトが実行されるときにプラットフォームによって呼び出されます。引数 eventList は，for ステートメントを使用して一つ一つ処理できる Event オブジェクトの配列を受け取ります（本項の例を参照）。

4. スクリプト・クラスが getInternalScriptError() という名前のパブリック・メソッドを使用していて，メソッドが 10 以上の値を返す場合，スクリプトはスクリプトを実行するプラットフォーム・サービスによってスキップされます。この機能を使用するには，次のコードを挿入します。
   ```
   int internalScriptError = 0;
   def int getInternalScriptError() { return internalScriptError; }
   def setInternalScriptError() { internalScriptError++; }
   ```
   このコードは，スクリプト・エラー・レベルを保持するメンバ変数 internalScriptError を 0 に初期化します。プログラマは，エラーが発生するたびに setInternalScriptError() メソッドを呼び出

してエラー・レベルを高くできます。

5. 必要に応じてその他の任意のメソッド，プロパティおよびメンバを含めることができます。

EPI スクリプト特有のベスト・プラクティス

OMi 処理のパフォーマンスに対する EPI スクリプトの影響を最小化するには，次の側面を考慮します。

- 一度にロードするスクリプト数に注意してください。スクリプトをロードすると，スクリプトがアンロードされるまでオーバーヘッドが発生します。したがって，一度にロードするスクリプトが多すぎるとパフォーマンス問題が起きる可能性があります。
- 可能ならばバックグラウンド・スレッドやキャッシュ情報を使用します（「ベストプラクティス」(379ページ)も参照）。
- RTSM（またはその他の外部ファイルやデータベース）からのデータが必要な場合，init() メソッドで RTSM への接続を開き，destroy() メソッドで接続を閉じます（「ベストプラクティス」(379ページ)も参照）。
- スクリプトの process() メソッドの実行に必要な処理時間を最小化します。特に， EPI スクリプトは速度重視で書きます。「ベストプラクティス」(379ページ)も参照してください。
- DNS 可用性，ネットワーク可用性，ファイル・アクセスの遅さなど外部リソースに問題がある場合，特に process() メソッドがそれらのリソースを使用する場合，EPI スクリプトはイベント処理全体に悪い影響を与えます。
- 専用の JAVA API を使用して RTSM にアクセスします（「ベストプラクティス」(379ページ)も参照）。
- OMi コンソールでスクリプトを設定する場合，スクリプトの実行順を設定できます。［スクリプト パネル］ツールバーの上下アイコンを使用してスクリプトの順番を上げ下げできます。標準設定では，スクリプトは読み取りおよび書き込みとマークされています。スクリプトを読み取り専用にするには読み取り専用アイコンを使用します。スクリプトが読み取りおよび書き込みか，読み取り専用かは，他のスクリプトとの関係で，スクリプトが実行される時期に対して次の影響があります。
  - イベントを変更する必要があるスクリプトには，データベースに対する読み取りおよび書き込みアクセスが必要です。スクリプトは，読み取りおよび書き込みとマークされ，潜在的な競合を避けるように設定された順番で実行されます。
  - 読み取り専用モードのスクリプトは，読み取りおよび書き込みスクリプトと並行して（非同期に）実行することができます。

つまり，読み取り専用スクリプトがイベントのスループットに与える影響は，読み取りおよび書き込みスクリプトよりも小さくなります。したがって，可能な限りスクリプトは読み取り専用にマークします。スクリプトを読み取りおよび書き込みにマークするには，読み取りおよび書き込みアイコンを使用します。

注: After Storing Events プロセス・ステップにあるスクリプトは，常に読み取り専用モードで実行されます。データベースに格納されたイベントは，EPI スクリプトにより変更されることはないからです。

- イベント・フィルタを使用するスクリプトのイベント・スループット全体に対する影響はやや大きくなります。フィルタの使用とフィルタなしでロードする必要のあるイベント数とのバランスを取ります。
- OMi コンソールで，スクリプトごとにタイムアウト値を設定できます。
  - 読み取りおよび書き込みモード（同期）で実行中のスクリプトは，タイムアウト時間が過ぎると停止し，スクリプトがイベントに行った変更はすべてロール・バックされます。
  - 読み取り専用モード（非同期）で実行されているスクリプトは，タイムアウト時間を過ぎると停止するだけです。

  OMi パイプライン全体に対し，問題が発生しているいずれのスクリプトの潜在的な影響が最小化するように，それによりプロセス中断のリスクが最小化するように，EPI スクリプトのタイムアウトの設定をすることを HP はお勧めします。

  タイムアウトの設定は，特に外部リソースを使用しているスクリプトにとって重要です。最適なタイムアウト値は，スクリプトが依存する外部リソースへのアクセスに問題がある条件下でスクリプトをテストして確認する必要があります。 スクリプトのテスト時に，次のログ・ファイルから各スクリプトが受信する各イベントに対する実行時間を見ます。
- メモリにあまり多くのデータを格納しないでください（JVM 統計も参照）。
- 必要がなくなったらリソースを開放し，デバッグ中はメモリ・リークに注意してください。
- リソース・ハンドルおよび再接続の有効性が必要か確認します（「ベストプラクティス」(379 ページ)も参照）。

例

OMi の外部のエスカレーション・データベースで，重大なイベントのエスカレーションと管理が行われていると状況を仮定し，次のサンプル・スクリプトを当てはめます。スクリプトは，エスカレーション・データベースからエスカレーションされたイベントの所有者を取得し，それを OMi イベント情報に追加します。

```
import com.hp.opr.api.scripting.Event;
```

```
import external.api.scripting.dataBase;

class actionOnStatus
  /* スクリプトは常にクラスとして宣言する必要があります。
   * スクリプト・クラスの名前は，OMi 管理コンソールのイベント処理スクリプトの
   * 名前として使用されます。*/
{
  int internalScriptError = 0;
          // 注 : getInternalScriptError メソッドが 10 以上の値を返す場合，
          // EPI サーバはスクリプトを
          // 実行しません。
  def int getInternalScriptError()
  { return internalScriptError; }

  def setInternalScriptError()
          // internalScriptError を増分させるには，このメソッドを使用します。
  { internalScriptError++; }

  Database externalDB;
          // イベントの強化に使用するデータ，
          // この場合エスカレーション・プロセスについての
          // 情報を含む
          // 外部データベースへの
          // ハンドル。
  def connectionString = "MyCloud,MyUsername,MyPwd";


  def init()
  {

    // Groovy スクリプトのコンパイル時に実行すべきアクション。
    // EPI がアクティブ化されるときに一度発生します。
    externalDB = connectDB(connectionString);
    if(!externalDB)
    {
      throwDatabaseException(
        "Could not connect to external dastabase"
        + " with connection string "
        + connectionString + "."
      );
      setInternalScriptError();
    }

  }

  def destroy()
  {
    // スクリプト・キャッシュを破棄する前に完了すべきアクション。
    externalDB.disconnect()
  }


  /* EPI サーバがイベント・パイプラインを処理するたびに
   * process() メソッドが呼び出されます。
   *
   * この例では，process() メソッドは外部データベースからの
   * パイプラインのすべての重大イベントに対する関連 CI のエスカレーション所有者を
   * 取得し，その情報を使用してイベントの OmUser プロパティ
   * を設定します。*/

  def process(eventList)
          // eventList 引数は，イベント・パイプラインの
          // イベントを含む配列を渡します。
```

```
{
  try
  {
    for( Event in eventList )
    {
      String stringOmUser = "";
      String stringQuery = "";

      int eventSeverity = Event.getSeverity();
      if( eventSeverity == CRITICAL)
      {
        def RecordSet rsEventInfo;

        stringQuery = "SELECT escalationOwner AS eo "
          + "FROM servers WHERE hostname = "
          + """" + Event.getRelatedCiHint() + """ "
          + "SORT BY contactPriority ASC";

        rsEventInfo = externalDB.Open(StringQuery);
        if( rsEventInfo.RecordCount > 0 )
        {
          stringOmUser =
          "Primary escalation owner: "
          + rsEventInfo.Fields("eo");
        } else {
          stringOmUser =
          "No escalation owner for CI: "
          + Event.getRelatedCiHint();
        }

        setOmUser(stringomUser);
      }
      // このイベントが処理されるときに完了すべきこと以外が
      // すべてここに来ます。
    }
  }

  catch( DatabaseException e )
  {
    setInternalScriptError();

    // このイベントが処理されるときに完了すべきこと以外が
    // この例外が発生するときに完了すべきこと以外がすべてここに来ます。
  }


  finally
  {
    // すべてのイベントの処理後に完了すべきアクション。
    // この例では，何もありません。
  }

}
```

カスタム・アクション・スクリプト

カスタム・アクションは，OMi イベント・ブラウザで選択したイベントに対して実行することができます。

トリガ

カスタム・アクションは，特定のイベントに対しオペレータが手動で起動します。カスタム・アクションを起動するには，次の手順を実行します。

1. OMi で，イベント・ブラウザとアクション・ウィンドウを含むパースペクティブを選択します。たとえば標準設定のイベント・パースペクティブまたは状況パースペクティブです。
2. アクションを適用するイベントを選択します。選択したイベントで利用可能なすべてのアクションがアクション・ウィンドウに一覧表示されます。
3. 実行するアクションをクリックします。

プロセス・フロー

OMi バージョン 9.2x のプロセス・フローは，次のとおりです。

- カスタム・アクション・スクリプトは，OMi ゲートウェイ・サーバの次のワーク・ディレクトリで実行されます。
  hpbsm_opr-scripting-host
- ログ情報が次のファイルに書き込まれます。
  <*OMi_HOME*>/log/opr-scripting-host/opr-scripting-host.log

OMi バージョン 9.1x のプロセス・フローは，次のとおりです。

- カスタム・アクション・スクリプトは，OMi ゲートウェイ・サーバの次のワーク・ディレクトリで実行されます。
  hpbsm_opr-ctxm-server
- ログ情報が次のファイルに書き込まれます。
  <*OMi_HOME*>/log/opr-ctxm-server/opr-ctxm-server.log

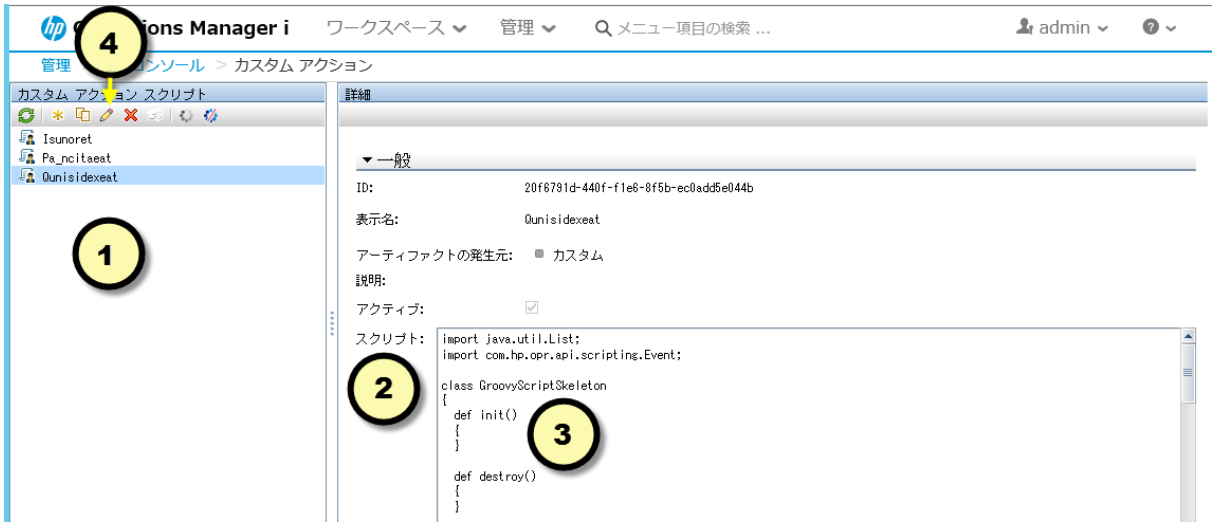
スクリプト管理

カスタム・アクション・スクリプトを設定し，イベント・ブラウザから利用可能にするには，次の手順を実行します。

1. OMi で，次のようにカスタム・アクションに移動します。

   ［管理］ > ［操作コンソール］ > ［カスタム アクション］

   カスタム・アクション画面を下図に示します。



画面には，次の 2 つの表示枠があります。

a. *カスタム・アクション・スクリプト* ①

この表示枠には，設定済みスクリプトが表示されます。アクティブなスクリプトは黒で，非アクティブはグレーで表示されます。

b. *詳細* ②

この表示枠には，スクリプト・コードを含む実際のスクリプトのプロパティが表示されます。③.

2. スクリプト・ウィンドウで直接コードを編集することはできません。次のように［カスタム・アクション・スクリプト］表示枠のツールバーを使用して，スクリプトにアクセスする必要があります。④

- 既存のスクリプトを編集するには，［**編集**］アイコン をクリックします。スクリプト・ウィザードが表示されます。［**スクリプト**］タブをクリックして，編集するスクリプトにアクセスします。

- 新しいスクリプトを設定するには，［**新規**］アイコン をクリックします。スクリプト・ウィザードが表示され，スクリプトと名前や説明など関連情報を入力できます。ウィザードには，スクリプト・クラスのスケルトンがあるため，開始点として使用できます。

- スクリプトをアクティブ化するには，［**アクティブ化**］アイコン をクリックします。

例

```
import java.util.List;
import com.hp.opr.api.scripting.Event;

class IsunoretAction
{
   def process(event) // Process the event passed
                // イベント・ブラウザから渡されたイベントの処理。
    {
       // 例 : イベントのサービス ID は,
          // 手動で定義済みの値に設定される。

          event.setOmServiceId("Isunoret");
    }
}
```

証明書スクリプト

バージョン 9.2x からは，証明書要求をスクリプトを使用して自動的に処理できます。次の関数が利用可能です。

- 要求の許可。
- 要求の拒否。
- 要求を無視し，手動による許可または拒否にする。

証明書要求を扱う Groovy スクリプトは，インストールにより利用可能になります（「例」(395ページ)を参照）。Groovy スクリプトは，カスタム・スクリプトの基盤として使用可能です。

プロセス・フロー

証明書スクリプト・クラスは，証明書要求のときに初めてインスタンス化されます。証明書要求ごとに process() メソッドが呼び出されます。

- ゲートウェイ・サーバの次のディレクトリにある証明書処理スクリプトの一部として，カスタム・アクション・スクリプトが実行されます。

  opr-scripting-host

- ログ情報が次のファイルに書き込まれます。

  <*OMi_HOME*>/log/opr-scripting-host/opr-scripting-host.log

スクリプト管理

スクリプトは，データベースに格納されており，OMi コンソールからアクセスできます。

証明書スクリプトを設定するには，次の手順を実行します。

1. OMi で，次のように証明書要求マネージャに移動します。

   Administration > Setup and Maintenance > Certificate Requests

   ［**証明書要求**］タブに，ドロップダウン・リストで選択した時間範囲に受信されたすべての証明書要求が表示されます。

2. Click the **Autogrant Script** tab.標準設定の自動許可スクリプトが表示されます。自動許可スクリプトは，要求の nodeName 属性の値に基づき証明書の許可または拒否を行います。標準設定スクリプトは，標準設定ではアクティブ化されていません。

   a. スクリプトをカスタマイズするには，［**編集**］アイコン  をクリック，またはスクリプト・ウィンドウをダブルクリックし，必要な変更を行います。

   b. スクリプトをアクティブ化するには，［**アクティブ化**］アイコン  をクリックします。

証明書スクリプトとして設定可能なのは 1 つのスクリプトだけです。設定されるとすぐにスクリプトはアクティブ化されます。 証明書スクリプトが必要ない場合は，［非アクティブ化］アイコン  をクリックして，自動許可sスクリプトを非アクティブ化してください。

例

次の例では，要求者のサブネットに基づき，証明書要求を許可または拒否します。

```
import java.net.InetAddress;
import java.util.Date;
import java.util.List;
import com.hp.opr.api.scripting.CertificateRequest;

class AutoGrantTest
{
 def init()
 {
 }

 def destroy()
 {
 }

 Boolean boolInSubnet( String IPAddress )
 {
  // 渡された IP アドレスが
  // 許可されたサブネットにあるか確認。
  String stringAllowedSubnet = "128.0";
```

```
  ByteOne = IPAddress.indexOf('.');
  ByteTwo = IPAddress.indexOf('.', ByteOne + 1 );

  String stringSubnet;
  stringSubnet = IPAddress.substring( 0, ByteTwo );

  return( stringSubnet.equals( stringAllowedSubnet ) );
 }

 def process(List<CertificateRequest> requests)
 {
  // 最初の要求オブジェクトを取得。
  def request = requests.get(0);

  if ( boolInSubnet( request.getIpAddress() ) )
  {
   request.deny();
  }
  else
  {
   request.grant();
  }
 }
}
```

サービス状況スクリプト

Service Health スクリプトは，主要管理指標（KPI）を計算するために使用するビジネス・ロジック・ルールを実装するために使用されます。KPI は，管理対象ネットワーク，コンポーネント，これらに関連付けられたサービスの状況ステータスを評価するために，Service Health Analyzer（SHA）で使用されます。

トリガ

サービス状況スクリプトは，Marble プロセスの一部として実行されます。

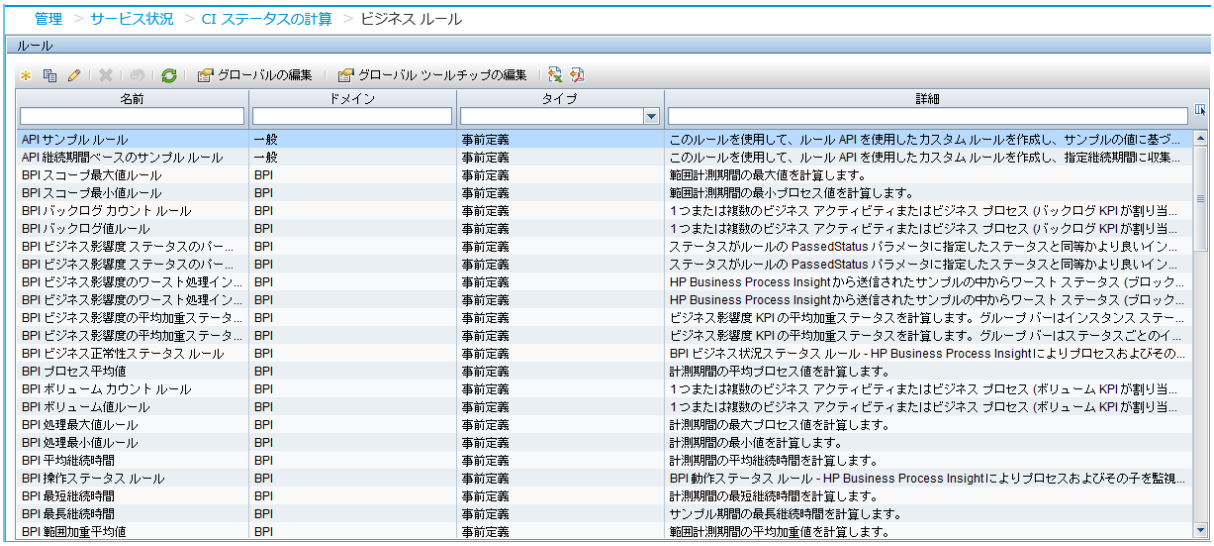
スクリプト管理

ビジネス・ルールを設定し，KPI の計算に使用するには，次の手順を実行します。

1. OMi で，［ビジネス ルール］に移動します。

   ［管理］ > ［サービス状況］ > ［リポジトリ］ > ［ビジネス ルール］

2. ［ビジネス ルール］画面は下の図とよく似ています。



画面にビジネス・ルールのリストが表示されます。［**タイプ**］列に表示されているとおり，ビジネス・ルールの一部は定義済みです。［**新規作成**］アイコン をクリックして新しいルールを追加するか，ルールを選択し，［**編集**］アイコン をクリックしてルールをカスタマイズできます。ビジネス・ルールの作成の詳細については，OMi ヘルプを参照してください。

3. ルールの作成後，KPI に含めることができます。KPI の作成やカスタマイズを行うには，［**リポジトリ**］タブをクリックし，［**KPI**］リンクを選択して KPI のリストを表示します。［**新規作成**］アイコン をクリックして新しい KPI を作成するか，KPI を選択し，［**編集**］アイコン をクリックして KPI をカスタマイズします。新しいルールまたはカスタマイズしたルールが，［KPI の編集］ダイアログの［選択されていないルール］リストまたは［選択されているルール］リストに表示されます。KPI の作成の詳細については，OMi ヘルプを参照してください。

トポロジ同期スクリプト

トポロジ同期パッケージにより，ユーザ定義 Groovy スクリプトをトポロジ同期プロセスに含めることができます。

スクリプトを使用して，トポロジ同期プロセスの一部として次のカスタム・アクションを起動できます。

- モデルが定める変換とエンリッチメントの制限事項にとらわれることなくサービス階層を操作する。
- 同期後の操作を実行する。
- HPOM のトポロジ・モデルを変える必要なくトポロジ・データを変更する。

プロセス・フロー

トポロジ同期スクリプトは次の 2 つの方法で起動できます。

- **自動** : スクリプトは hpbsm_wde プロセスの一部として自動的に起動されます。

  ログ情報が次のファイルに書き込まれます。

  <*OMi_HOME*>/log/wde/opr-svcdiscserver.log

- **手動（非推奨）** : トポロジ同期スクリプトは，次のバッチ・ファイルを実行してコマンド・ラインから手動で起動できます。

  opr-startTopologySync.bat

  ログ情報は次の場所に書き込まれます。

  <*OMi_HOME*>/log/opr-topologysync

注意: 自動的に起動されるスクリプトの実行が同期プロセスに悪影響を及ぼす場合，トポロジ同期プロセスとイベント同期プロセスの両方が中断される危険性があります。「ベストプラクティス」(379ページ)に記述されたベスト・プラクティスに従って，できるかぎりリスクを最小限に抑えてください。

スクリプト・ファイルの名前から，トポロジ同期プロセスで実行される場所が決まります。プロセス内の 3 つの場所でスクリプト・ファイルが起動されます。スクリプト・ファイルの名前は次の規則に従います。

1. 存在する場合，HPOM ノードとサービスがマップされる前に，preEnrichment.groovy という名前のスクリプトが実行されます。
2. 存在する場合，HPOM ノードとサービスがマップされてから uCMDB にアップロードされるまでの間に，preUpload.groovy という名前のスクリプトが実行されます。
3. 存在する場合，ノードとサービスが uCMDB にアップロードされた後に，postUpload.groovy という名前のスクリプトが実行されます。

スクリプト管理

トポロジ同期スクリプトはトポロジ同期パッケージに不可欠な部分であり，パッケージをインストールするとデータベースに保存されます。トポロジ同期プロセスは，ファイル・システム内のスクリプトを参照することなくデータベースに保存されているスクリプトを実行するため，ファイル・システム内のスクリプトに変更や追加を行った場合はトポロジ同期パッケージを再度アップロードする必要があります。

スクリプトを追加，削除，変更するには，次の手順を実行します。

1. 完成したスクリプトまたは変更したスクリプトを次のフォルダに配置します。

   <*OMi_HOME*>/conf/opr/topology-sync/sync-packages/<*Sync Package Name*>

   スクリプトを削除するには，次の手順を実行します。

   a. sdtool コマンドを使用してスクリプトをデータベースから削除します。

   b. スクリプトをファイル・システムから削除します。

   > **注:** 標準インストールでインストールしたスクリプトはファイル・システムに存在します。これらのスクリプトはシステムで使用されるため，削除しないでください。

2. sdtool コマンドを使用して同期パッケージをデータベースにアップロードします。

トポロジ同期スクリプト特有のベスト・プラクティス

トポロジ同期スクリプトの開発時は，次の点を考慮してください。

- マルチサーバ環境では，スクリプトを常に同期するための特別な配慮が必要になります。新しいスクリプトや変更したスクリプトを含む同期パッケージをデータベースにアップロードした後に，パッケージのアップロードに使用されるサーバのファイル・システムに保存されているスクリプトに応じて，すべてのサーバ上の同期プロセスが変更されます。ただし，他のすべてのサーバのファイル・システムに保存されているスクリプトがデータベース内のスクリプトに一致しなくなるため，サーバのうちの 1 台をスクリプトのアップロードに使用すると，同期が以前の好ましくない状態に戻る可能性があります。これを回避するために，HP はトポロジ同期スクリプトの管理に同じサーバを使用することをお勧めします。

- 同じ同期パッケージに含まれるプレアップロード・スクリプトおよびポストアップロード・スクリプトは，変数に関してスコープを共有します。そのため，preUpload.groovy スクリプト内の変数に割り当てられた値が，同じパッケージ内の postUpload.groovy スクリプトで使用されます。別の同期パッケージをまたがってスクリプトが同じスコープを共有することはないため，同じ同期パッケージに属さない場合にかぎり，スクリプトで同じ変数名が使用される場合があります。

- スクリプトは常に scriptInterface オブジェクトおよび syncData オブジェクトにアクセスします。これらのオブジェクトで提供される API の関数によって，内部オブジェクトがカプセル化されます。内部オブジェクトに直接アクセスするのではなく，常にこれらのオブジェクトを参照してください。たとえば次のように使用します。
  myCI = createCI(caption:"myCI")
  CI クラスを明示することなく ICi 実装を返すには，次のように使用します。
  new CI(...) myCI
  「利用可能な API」(414ページ)も参照してください。

- クラスではなく，Java インタフェースのみを使用します。さらに，一連の特殊なファクトリ関数を使用できます。「利用可能な API」(414ページ)も参照してください。

例

文字列 "disk" を含むサービスのキャプションにプレフィックス ">>>" およびサフィックス "<<<" を追加して強調表示し，変更の件数が電子メールで管理者に報告するとします。

プレフィックスとサフィックスは OM ノードがマップされた後に追加します。次の preUpload.groovy スクリプトは，キャプションを変更し，変数 iChanges の変更の件数を保存します。

```
iChanges = 0;

// syncData オブジェクトの getConfigurations() メソッドを使用して
// すべての解決済みのサービスを取得し，配列サービスに保存する。
arrayServices = syncData.getConfigurationItems();


// サブ文字列 "disk" に関して配列サービスのすべての要素を確認し，
// 見つかった場合はプレフィックスとサフィックスを追加して，
// カウントを加算する。
arrayServices.findAll{ it.getCaption().indexOf( "disk" ) >= 0 }.each
{
   stringCaption = svc.getCaption();
   svc.setCaption( ">>> " + stringCaption + " <<<" );
   iChanges++;
}
```

postUpload.groovy スクリプトの一環として，アップロード完了後に電子メールが送信されます（最後の例の sendmail コマンドは Windows 標準プラットフォームで利用できません）。このスクリプトは preUpload.groovy のスクリプトと同じ同期パッケージに含まれるため，変数 iChanges は同様にスコープ内です。

```
msg = "Sync Ready!${iChanges} CIs modified!";
scriptInterface.exec( "sendmail Administrator " + msg );
```

イベント転送スクリプト

イベント転送スクリプトは上位のアプリケーションにイベントを転送します。イベント転送スクリプトは，外部システムへのアダプタを作成するためにおもに使用します。

イベント転送スクリプトは，次のタイプの操作を扱う必要があるため，他のスクリプトよりも複雑になる傾向があります。

- パイプラインに到着したイベントの転送。
- 更新をイベントに転送。

- 変更を受信しイベントに転送。
- 転送されたイベントを受信する外部サーバ上で実行されているプロセスに対するイベント相関処理のサポート。
- Web サービスの呼び出しから収集された情報，コマンド・ライン・ツールで入手した情報，スクリプトによりログ・ファイルに書き込まれた情報など，スクリプトに必要な外部情報の交換のサポート。

注: イベント・スクリプトはイベントを強化するためには使用しません。イベント強化のためのスクリプトの詳細については，「イベント処理インタフェース・スクリプト」(384ページ)を参照してください。

他のスクリプトと違ってイベント転送スクリプトでは，具象クラス OprEvent を使用します。

OprEvent クラスでは，イベント情報の変更に使用するコードを厳密に receiveChange() メソッドのカスタマイズに限定します。このメソッドは，ターゲット・サーバがイベント変更を通知するときに呼び出されます。

次のコードは，イベント転送スクリプトの実装に有効な宣言です。

```
package com.hp.opr.api.ws.adapter;
import com.hp.opr.api.ws.model.event.OprEvent;

public abstract interface ExternalProcessAdapter
{
    void init(final InitArgs args);
    void destroy();

    Boolean ping(final PingArgs args);

    /*次のメソッドはカスタマイズ可能です*/
    Boolean receiveChange(final ReceiveChangeArgs args);

    Boolean getExternalEvent(final GetExternalEventArgs args);
    Boolean forwardEvent(final ForwardEventArgs args);
    Boolean forwardChange(final ForwardChangeArgs args);
    String toExternalEvent(final OprEvent event);

}
```

OprEvent クラスは，インポートされ，スクリプト・クラス（ここでは ExternalProcessAdapter としますが，他の名前の場合もあります）には，標準の init() と destroy() メソッドがあります。次のメソッドにより，イベント転送が実装されます。

- ping() メソッドは接続サーバとの通信を試行し，成功した場合は true を返します。
- カスタマイズ可能な receiveChange() メソッドは外部の変更を受信し，外部サーバが行った変更の

同期を，OMi が実行できるようにします。receiveChange() メソッドは，イベント同期 Web サービスにより呼び出され，次の HTTP 要求に応答します。

- 次に対する PUT 要求。

  /opr-gateway/rest/synchronization/event/*<event ID>*

- 次に対する POST 要求。

  /opr-gateway/rest/synchronization/event_change/*<event ID>*

HTTP ヘッダ・コンテンツタイプを次の値のいずれかに設定する必要があります。

- application/xml
- application/json
- application/soap+xml
- text/xml
- text/plain

注: 接続サーバで，同期がサポートされない場合，receiveChange() メソッドは引数の HTTP 応答コードを 403（禁止）に設定し，false を返します。

実行が成功し，引数で渡した値が更新された場合，receiveChange() メソッドは true を返します。

- getExternalEvent() メソッドは，外部サーバ上のプロセスに管理を移したイベントの，イベント・ブラウザで実行されたユーザ要求に応答して，イベント詳細を取得します。イベント詳細に対応した値が受信され，メソッドの引数 GetExternalEventArgs により渡されます。
- forwardEvent() メソッドは，メソッドの引数 ForwardEventArgs で渡されたイベントを外部プロセスに転送します。
- forwardChange() メソッドは，メソッドの引数 ForwardChangeArgs で渡されたイベントに対し，このメソッドが最後に呼び出し成功して以降に発生したイベント変更を転送します。特定のイベントに対して初めてこのメソッドが呼び出された場合，イベントが転送されて以降のすべての変更が含まれます。
- toExternalEvent() メソッドは，イベントを外部オブジェクトに変換します。イベント同期 Web サービスにより呼び出され，次の HTTP 要求に応答します。

  次に対する GET 要求。

  /opr-gateway/rest/synchronization/event/*<event id>*

プロセス・フロー

イベント転送スクリプトは，ゲートウェイ・サーバ上のイベント転送プロセスの一部としてターゲット・サーバから変更が到着するときに，トリガされます。

バージョン 9.2x からは，イベント転送スクリプトはデータベースに格納されます。9.20 より前の OMi バージョンでは，スクリプトはゲートウェイ・サーバ上のファイル・システムに格納されていました。

OMi バージョン 9.2x のプロセス・フローは，次のとおりです。

- イベント転送スクリプトは，次のワーク・ディレクトリで，ゲートウェイ・サーバ上で実行されているプロセスの一部として実行されます。

  hpbsm_opr-scripting-host

- ログ情報が次のファイルに書き込まれます。

  <*OMi_HOME*>/log/opr-scripting-host/opr-event-sync-adapter.log

OMi バージョン 9.1x のプロセス・フローは，次のとおりです。

- イベント転送スクリプトは，次のワーク・ディレクトリで，ゲートウェイ・サーバ上で実行されているプロセスの一部として実行されます。

  hpbsm_opr-ctxm-server

- ログ情報が次のファイルに書き込まれます。

  <*OMi_HOME*>/log/opr-ctxm-server/opr-event-sync-adapter.log

OMi バージョン 9.0x のプロセス・フローは，次のとおりです。

- イベント転送スクリプトは，次のワーク・ディレクトリで，ゲートウェイ・サーバ上で実行されているプロセスの一部として実行されます。

  hpbsm_wde

- ログ情報が次のファイルに書き込まれます。

  <*OMi_HOME*>/log/wde/opr-event-sync-adapter.log

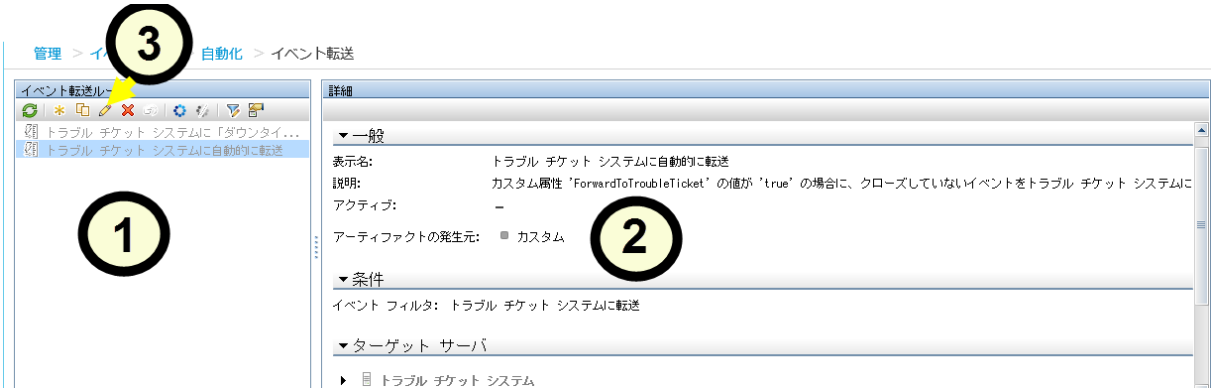
スクリプト管理

9.20 より前の OMi では，イベント転送スクリプトはゲートウェイ・サーバ上のファイル・システムに格納されていました。バージョン 9.20 からは，データベースに格納されています。イベント転送スクリプトを設定し，アクティブ化するには，次の手順を実行します。

1. OMi で，次のようにイベント転送に移動します。

   ［管理］ > ［イベント処理］ > ［自動化］ > ［イベント転送］

   イベント転送画面を下図に示します。



   画面には，次の 2 つの表示枠があります。

   a. イベント転送ルール ①

      この表示枠には，設定済みイベント転送ルールの一覧が表示されます。アクティブなスクリプトは黒で，非アクティブはグレーで表示されます。

   b. 詳細 ②

      この表示枠には，選択したルールのプロパティが表示されます。

2. スクリプト・ウィンドウで直接コードを編集することはできません。次のように［イベント転送ルール］表示枠のツールバー ③ を使用して，スクリプトにアクセスする必要があります。

   - 既存のルールを編集するには，［**編集**］アイコン をクリックします。［イベント転送ルールの編集］ダイアログが表示されます。ルールをカスタマイズし，［**OK**］をクリックします。

   - 新しいルールを設定するには，［**新規**］アイコン をクリックします。［イベント転送ルールの編集］ダイアログが表示されます。ルールの情報を入力し，［**OK**］をクリックします。

   - ルールをアクティブ化するには，［**アクティブ化**］アイコン をクリックします。

標準設定では，次のルールがインストールされます。

- トラブル・チケット・システムに「ダウンタイムを開始しました」イベントを自動的に転送
- トラブル・チケット・システムに自動的に転送

例

サンプル・スクリプト LogfileAdapter がシステムで使用できます。

外部命令取得スクリプト

外部命令取得スクリプトは，ナレッジ・ベースやデータベースなどの外部インタフェースからイベント命令を取得するために使用されます。OMi は，指定したイベント・フィルタの少なくとも 1 つが，イベント・ブラウザで選択したイベントと一致している場合に命令取得スクリプトの実行をトリガします（命令は，［イベント詳細］ペインの［**命令**］タブで選択した場合に取得されます）。

命令取得スクリプトは，OMi ゲートウェイ・サーバの opr-scripting-host プロセスによって実行されます。ログ情報は，<*OMi_HOME*>/log/opr-scripting-host ファイルに書き込まれます。

OMi には，外部命令インタフェース用のサンプル命令取得スクリプトおよびフィルタが含まれている定義済みのコンテンツ・パックが用意されています。コンテンツ・パックにはサンプル・ポリシーも含まれています。この初期スクリプトに基づいて，ビジネス・ニーズに合ったカスタム・スクリプトを作成できます。コンテンツ・パックの詳細については，OMi 管理ガイドを参照してください。

プロセス・フロー

OMi によってスクリプトの実行がトリガされると，イベントがスクリプトに渡されます。スクリプトは，命令インタフェース名（プロパティ 'event.instructionInterfaceName'）やパラメータ文字列（'event.instructionParameterString' プロパティ）など，指定した命令パラメータを含め，イベントのすべてのプロパティにアクセスできます。命令テキストを生成する外部プログラムに渡す任意のオプションをパラメータとして入力できます（ポリシー・テンプレートで設定するポリシー変数やポリシー・パラメータなど）。

［**利用可能な命令**］フラグは，命令がイベントで使用できるかどうかを示します（選択したイベントの［I］カラムに  アイコンが表示されます）。このプロパティは，イベントの作成時に true に設定され，エージェントの対応するメッセージ（OMi サーバで取得され，イベントに変換されるメッセージ）に，次のいずれかが含まれる場合にデータベースに保存されます。

- ポリシーで指定した命令テキストの UUID
- 命令テキストの取得に使用される命令テキスト・インタフェースの名前

また，OMi は，イベントがイベント・ブラウザにロードされ前に，すべてのアクティブな外部命令テキスト・インタフェース定義を評価し，イベント・フィルタがイベントと一致するかどうかをチェッ

クします。少なくとも 1 つのイベント・フィルタが一致すると，［**利用可能な命令**］イベント・フラグも true に設定されます。

注: 一致する外部命令テキスト・インタフェース定義を無効化または削除して，OMi サーバを再起動すると，関連イベントの［**利用可能な命令**］フラグの設定は true ではなくなります。

命令テキスト・インタフェースへのマッピングは，次のいずれかの方法で行うことができます。

- すべてに一致する 1 つのイベント・フィルタを作成して，このイベント・フィルタから単一の Groovy スクリプトへの 1 つのマッピングを定義する。スクリプトに渡されるイベントの命令インタフェース名に応じて，さまざまなソースから命令テキストを取得できます。

  ヒント: ［**利用可能な命令**］プロパティが true に設定されているが，実際の命令が含まれていないイベントが数多く生成されないようにするには，命令テキストが使用できるイベントのみに一致するようにイベント・フィルタを定義します。

  イベント・フィルタリングを容易にするために，［**命令インタフェース名**］と［**命令パラメータ文字列**］詳細フィルタ要素を使用できます。このようにして，イベント・ブラウザに表示されるイベントのセットを，指定した命令インタフェースおよびパラメータ文字列があるイベントに制限できます。イベント・フィルタの詳細については，OMi ユーザ・ガイドを参照してください。

- 複数のイベント・フィルタと，Groovy スクリプトへのマッピングを作成する。最初に一致するイベント・フィルタによって，マップされた Groovy スクリプトが呼び出されます。この代替方法を選択する場合，同じイベントのセットに一致しないように各イベント・フィルタを定義します。

  注: OMi では常に，最初にイベントに一致するフィルタが選択されます。イベント・フィルタの評価順序は定義できないため，独立したイベント・フィルタ（異なるイベントのセットに一致するイベント・フィルタ）のみを定義する必要があります。

スクリプト管理

スクリプトは，データベースに格納されており，OMi コンソールからアクセスできます。

命令取得スクリプトを設定するには，次の手順を実行します。

1. 外部命令マネージャに移動します。

   ［**管理**］ > ［**操作コンソール**］ > ［**外部命令**］

   画面には，次の 2 つの表示枠があります。

a. *スクリプト*

このペインには，設定済みスクリプトが表示されます。アクティブなスクリプトは黒で，非アクティブはグレーで表示されます。

b. *詳細*

このペインには，選択したスクリプトの一般プロパティと詳細プロパティ（スクリプト自体も含む）が表示されます。

2. 次の操作を行う場合，［スクリプト］ペインのツールバーを使用します。

- 既存のスクリプトを編集する。［**編集**］アイコン  をクリックして，スクリプト・ウィザードを開き，［**スクリプト**］タブをクリックして編集するスクリプトにアクセスします。
- 新しいスクリプトを設定する。［**新規**］アイコン  をクリックして，スクリプト・ウィザードを開きます。スクリプトと名前や説明など関連情報を入力します。ウィザードには，スクリプト・クラスのスケルトンがあるため，開始点として使用できます。
- スクリプトをアクティブ化 / 非アクティブ化する。［**アクティブ**］  または［**非アクティブ**］  アイコンをクリックします。

スクリプトの出力

イベント（関連する命令を含む）は，OMi イベント・ブラウザで表示できます。命令は，［イベント詳細］ペインの［**命令**］タブに表示されます。

Groovy スクリプトは，プレーン・テキストまたは HTML 形式で出力を返します。命令テキストに URL が含まれている場合，自動的にハイパーリンクに変換されます。http://，https://，ftp://，fttps://，telnet://，mailto: URI スキーム名で始まる URL がサポートされています。外部 Web サイト，サポート・サイト，ドキュメント・リポジトリ，トラブルシューティング情報などの URL を追加できます。

また，www で始まる任意の Web サイトのアドレスを入力することもできます。

イベント・ブラウザでハイパーリンクをクリックすると，新しいウィンドウが開き，スクリプトによって返されたページが表示されます。

例

本項では，2 つのサンプル命令取得 Groovy スクリプトを紹介します。

サンプル命令取得 Groovy スクリプト :

```
import com.hp.opr.api.scripting.Event
```

```
class GroovyScriptSkeleton
{
 // メソッド「getInstructions」が呼び出される前に呼び出されます。
 def init()
 {}
 // メソッド「getInstructions」が呼び出された後に呼び出されます。
 def destroy()
 {}
 // このメソッドは，命令テキストが取得されるイベントをパラメータとして取得し，
 // イベント・ブラウザに表示される命令テキストを返します。
 def getInstruction(Event event)
 {
  if ("iti" == event.instructionInterfaceName)
  {
   // 実行するコマンド。（エージェントから送信される）パラメータ文字列を引数として取得します。
   // そのため，エージェントがエージェント・システムの名前を送信すると，このコマンドによってエージェント・システムが ping さ
れ，
   // ping コマンドの出力が命令テキストとして返されます。
   def command = "ping -n 1 " + event.instructionParameterString
   def proc = command.execute() // コマンドを実行
   proc.waitForOrKill(3000)   // タイムアウトまでに 5 秒間待機
   if (proc.exitValue() == 0)
    return "ping of node '" + event.instructionParameterString + "':" + proc.in.text
   else
   {
    if (event.instructionInterfaceName == null)
     return "No instructions are available for event with ID '" + event.id + "'."
    else
     return "Cannot get instructions for interface '" + event.instructionInterfaceName + "'."
   }
  }
 }
 // このメソッドは，命令テキストが取得されるイベントのリストをパラメータとして取得します。
 // すべてのイベントの命令テキストは，キー（イベントの ID）と値（このイベントの命令テキスト）を含むマップとして返されます。
 def getInstructions(List<Event> events)
 {
  events.collectEntries { event ->
   [ event.id, getInstruction(event) ]
  }
 }
}
```

サンプル命令取得 Groovy スクリプト（HTML 出力）：

```
import com.hp.opr.api.scripting.Event

class GroovyScriptSkeleton
{
 def init()
 {}

 def destroy()
 {}

 def getInstruction(Event event)
 {
  if ("iti" == event.instructionInterfaceName)
  {
   // 外部ソースから命令テキストを取得するコードを追加します。
   return """<html><body><h1>My First Heading</h1><p>My first paragraph.</p><a href="http://www.google.com">This is a link</a>
```

```
    </body></html>"""
   }
   else
   {
    return "Cannot get instruction text for interface '" + event.instructionInterfaceName + "'."
   }
  }

  def getInstructions(List<Event> events)
  {
   events.collectEntries { event ->
    [ event.id, getInstruction(event) ]
   }
  }
}
```

第36章: 参照情報

本項には，Groovy スクリプトに関する参照情報または，情報自体がすでにどこかに記載されている場合には参照情報へのリンクが含まれています。

次の情報が含まれています。

- 「Groovy コンソール」(410ページ)では，Groovy コンソールのインストール方法と OMi 用スクリプト開発の準備方法が説明されています。スクリプトのデバッグについてのヒントも含まれています。
- 「利用可能な API」(414ページ)には，OMi 用 Groovy スクリプト開発時に役に立つ可能性のある内部および外部 API の一覧があります。

Groovy コンソール

Groovy スクリプトの開発とデバッグを行う最善の方法は，Groovy コンソールを使用することです。

インストール

Groovy スクリプトの開発環境を準備するには，次の手順を実行します。

1. Java をインストールします。

   Groovy には 1.4 以降の Java バージョンが必要です。まだインストールしていない場合は，次の URL から最新の Java ディストリビューションを入手してください。

   http://java.sun.com

   インストーラを実行し，環境変数 JAVA_HOME を設定します。Windows では，次の手順を実行します。

   a. ［**コントロール パネル**］を開き，［**システムとセキュリティ**］ > ［**システム**］セクションに移動します。

   b. ［**システムの詳細設定**］をクリックします。［*システムのプロパティ*］ダイアログが開きます。

   c. ［**環境変数**］ボタンをクリックします。［*環境変数*］ダイアログが開きます。

   d. ［**ユーザ環境変数**］ボックス下の［**新規...**］をクリックし，変数 JAVA_HOME を追加して，Java をインストールするパスをその値として割り当てます。

e. ［**システム環境変数**］ボックスの変数 %Path% を選択し，［**編集...**］ボタンをクリックし，変数 %JAVA_HOME% をパスに追加します。

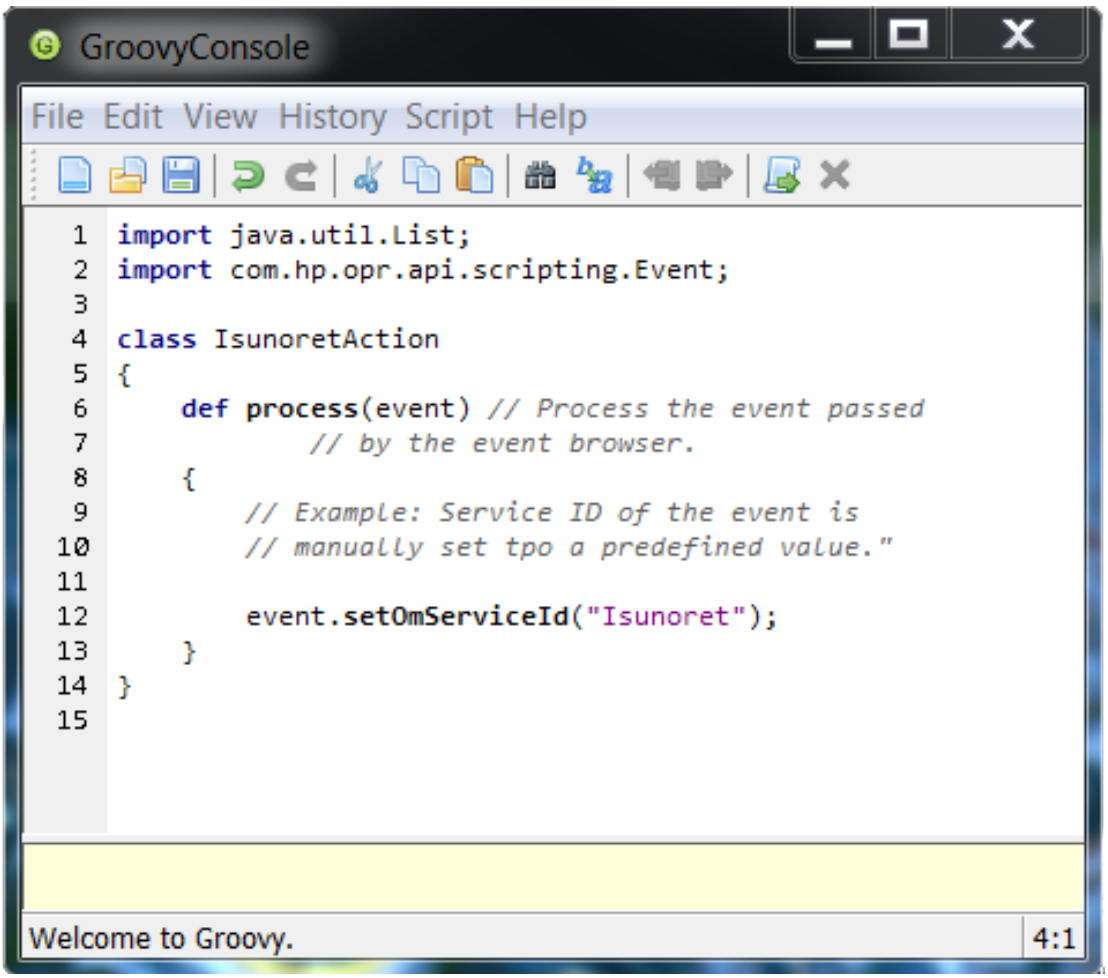
2. Groovy コンソールをインストールするには，次の URL の 公式 Groovy Web サイトのダウンロード・ページから必要なオペレーティング・システムのインストーラを入手します。

   http://groovy.codehaus.org/

   標準設定では，インストーラによってシステムへのインストール・パスを保持する環境変数 GROOVY_HOME が追加され，バイナリへのパスがパスに追加されます。本項では以降 %GROOVY_HOME% をインストール・パスを示すために使用します。

   注: インストール・パスに空白を使用するとインストーラに問題が発生します。HP はパスに空白のないディレクトリへのコンソールのインストールをお勧めします。

   Groovy コンソールを次の図に示します。



3. OMi データ処理サーバまたはゲートウェイ・サーバ以外のサーバで Groovy コンソールを実行する場合，プラットフォーム・ライブラリを定義済みの場所にコピーする必要があります。ライブラリを Groovy コンソールで使用できるようにするには，次の手順を実行します。

   a. テキスト・エディタで %GROOVY_HOME%\conf\groovy-starter.conf ファイルを開きます。

   b. 次の形式の行がアクティブか確認します（必要に応じて，行を追加またはコメント・アウト）。

      load !{user.home}/.groovy/lib/*.jar

   c. 変更した場合は，ファイルを保存します。

   d. user.home に該当するディレクトリにライブラリをコピーします。ここで， user.home はライブラリ・ファイルが格納されるディレクトリです。

   場所がわからない場合は，次のディレクトリにライブラリをコピーします。このディレクトリは常にロードされます。

   %GROOVY_INSTALL%\lib

   OMi をカスタマイズするには，多くの場合，次のライブラリが必要です。

   - Apache Commons Libraries

     <install>/lib/commons-*.jar

     このライブラリには，ログ関数 log4j など一般的な関数が含まれます。

   - JSON Library

     <install>/lib/json.jar

     JSON コードの解析と生成に使用します。

   - <install>/lib/opr-external-api.jar

   - <install>/lib/opr-common.jar

   - Apache Wink REST Web Services APIs

     <install>/lib/wink-*.jar

   - JSR-311 REST APIs

     <install>/lib/jsr311-api.jar

- RTSM Topology APIs

  <install>/lib/ucmdb-api.jar

OMi との互換性

OMi では，次の表に示す Groovy バージョンがサポートされます。これより新しい Groovy バージョンに含まれる機能はサポートされません。

OMi バージョン	最新の互換性のある Groovy バージョン
10.00	2.2.1
9.13，9.2x	1.7.3
9.10，9.11，9.12	1.6.0（EPI およびカスタム・アクション・スクリプト用） 1.7.3（その他のスクリプト用）
9.01	1.6.0（EPI およびカスタム・アクション・スクリプト用） 1.5.6（その他のスクリプト用）

Groovy コンソールを使用したデバッグ

次のヒントは Groovy スクリプトのテストとデバッグを行うときに役立ちます。

- Groovy コンソールはインタプリタです。import せずにクラスを参照するなどの特定のエラーは，コード行が実行されるまではレポートされません。
- スクリプトで使用する各クラスに import ステートメントがあることを確認してください。
- デバッグ中は，広範ログを使用します。ログはデバッグ支援の非常に強力なツールになり得ますが，不必要な情報をログしないように注意します。ログのエントリが多すぎると，ログがすぐにロール・オーバーしてしまうだけでなく，特定の問題の発見が困難になる場合があります。

  注: スクリプトを実運用環境に移す前にログ・コードを非アクティブ化してください。実運用環境で非アクティブ化しないログ情報を使用することは控えめにします。「ベストプラクティス」(379ページ)も参照してください。

- スクリプトを有効にして Apache LogFactory クラスを使用し，Apache にログを行うことができます。このクラスは Apache Common Libraries で定義されています。スクリプト・クラスにログ・ファクトリを含めるには，ログ型のメンバ変数を宣言し，それを関連する OMi パッケージのログ・メソッドの名前に初期化します。たとえば次のようにします。

  ```
  private static Log s_log = LogFactory.getLog("com.hp.opr.epi.MyLogs")
  ```

Apache にログを行う場合の注意：

- 一度だけ初期化が行われるように，static としてメンバ変数を宣言します。
- OMi スクリプトのいずれかを使用している場合，パッケージ名 com.hp.opr.epi を使用します。これにより，ログ・エントリは自動的に EPI スクリプティング・ログ・ファイルに追加されます（「イベント処理インタフェース・スクリプト」(384ページ)を参照）。
- カスタム・パッケージにアクセスするには，パッケージのプロパティ・ファイルが必要です。プロパティ・ファイルは，ディレクトリ conf/core/Tools/log4j のサブディレクトリにあります。サブディレクトリの名前は，スクリプトをトリガするプロセス名に対応しています。これについては，「スクリプトの開発およびデプロイメント」(384ページ)の項に説明されている各スクリプト・タイプのトリガの見出し下で詳述されています。
- スクリプトのログ・レベルを調整するには，該当するプロパティ・ファイルのログ・レベル・エントリを編集します（前述の箇条書き項目を参照）。
- Apache ログ・エントリを作成するには，次の構文を使用します（ログ・メンバと同じ名前を使用した場合）。

  s_log.error("*<Error text>*")

  例：

  s_Log.error("Attempt to get event failed; HTTP status:${response.statusCode}")

  メソッド名として該当するエラー・レベルを指定します（例では error()）。スクリプトの動作を追跡するには，スクリプトの正常ロードを確認できるようにエラー・レベル info() でのログ・エントリを含めることを HP は推奨します。

利用可能な API

本項では，次の利用可能なプラットフォームおよび外部 API について詳述します。

「外部 API」

API	アプリケーション
Java JDK	次のような関数のための**汎用 JDBC アクセス** • データベースへの接続の確立。 • データ・ソースへの接続。 • SQL 例外の取り扱い。 • テーブル作成とデータ入力。 • データの取得と変更。 • データベース・トランザクションの使用。 詳細については， http://docs.oracle.com/javase/tutorial/jdbc/basics を参照。
	一般的な Web サービス・アクセス。 URL.openConnection() 関数の使用
	コマンド・ライン・ツール。

API	アプリケーション
Apache	アプリケーション・バイナリを変更せずに**ランタイムのログ**によるデバッグのサポート : Apache Log4j logging API の使用（「Groovy コンソール」(410ページ)も参照）。 詳細については，http://logging.apache.org/log4j/1.2/ を参照。
	RESTful Web サービスとの通信 : Wink JSR Client API の使用。 詳細については，http://incubator.apache.org/wink/ を参照。
	HTTP プロトコルを利用して，分散通信を使用している環境で情報の取得 : Apache HTTP Components API の使用。 詳細については，http://hc.apache.org/ を参照。
	シンプル・オブジェクト・アクセス・プロトコル（SOAP）をサポートする Web サービスとの通信 : Apache Axis2 API の使用。 詳細については，http://axis.apache.org/axis2/java/core/ を参照。

「EPI スクリプトで使用される関数」

Groovy スクリプトAPI とすべての引数および型の詳細については，製品に含まれる Java API のマニュアルに記載されています。ドキュメントは次のディレクトリにあります。
<OMi_HOME>/opr/api/doc.

EPI スクリプトの開発の詳細については，「イベント処理インタフェース」(148ページ)を参照してください。

「トポロジ同期スクリプトの API」

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 Java API ドキュメントを参照してください。

- *<OMi_HOME>*/opr/api/doc/opr-external-api-javadoc.zip
- *<OMi_HOME>*/opr/api/doc/opr-ts-interfaces-javadoc.zip

次は，外部から見えるタイプと関数のリストです。

- 外部から見えるインタフェース:
  - ICi
  - ICiRelation
  - ICiMessage
  - IScriptingInterface
  - INavigator
  - INode
  - ISyncData
- 外部から見えるクラス:
  - CiMessageCategory
  - CiMessageSeverity

- IScriptingInterface インタフェースは，変数 scriptInterface を介してアクセス可能です。
- ISyncData は，変数 syncData を介してアクセス可能です。

トポロジ同期スクリプトの開発の詳細については，「トポロジ」(23ページ)を参照してください。

「Service Health ルールの API」

Service Health ルールの開発の詳細については，「イベント・タイプ・インジケータと状況インジケータ」(36ページ)を参照してください。

第X部: サービス状況

第37章: サービス状況ルール API

注: OMi 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（「API サンプル・ルール」(422ページ)および「API 継続時間ベースのサンプル・ルール」(424ページ)）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。**OMi 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，『OMi 管理ガイド』のUnderstanding the Service Health Calculation Rules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 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，OMi 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。 Groovy スクリプトの開発およびデプロイの詳細については， 「Groovy スクリプト」(376ページ)を参照してください。

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\API_docs\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。詳細については， 「API グループと兄弟ルール」(420ページ)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。詳細については， 「API サンプル・ルール」(422ページ)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。詳細については， 「API 継続時間ベースのサンプル・ルール」(424ページ)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，「ルール API を使用したルールの作成」(425ページ)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，「ツールチップ・エントリの使用方法」(432ページ)を参照してください。

「ルール API コードからログ・ファイルに書き込む方法」(433ページ)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，『OMi 管理ガイド』のKPIs Repository page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。

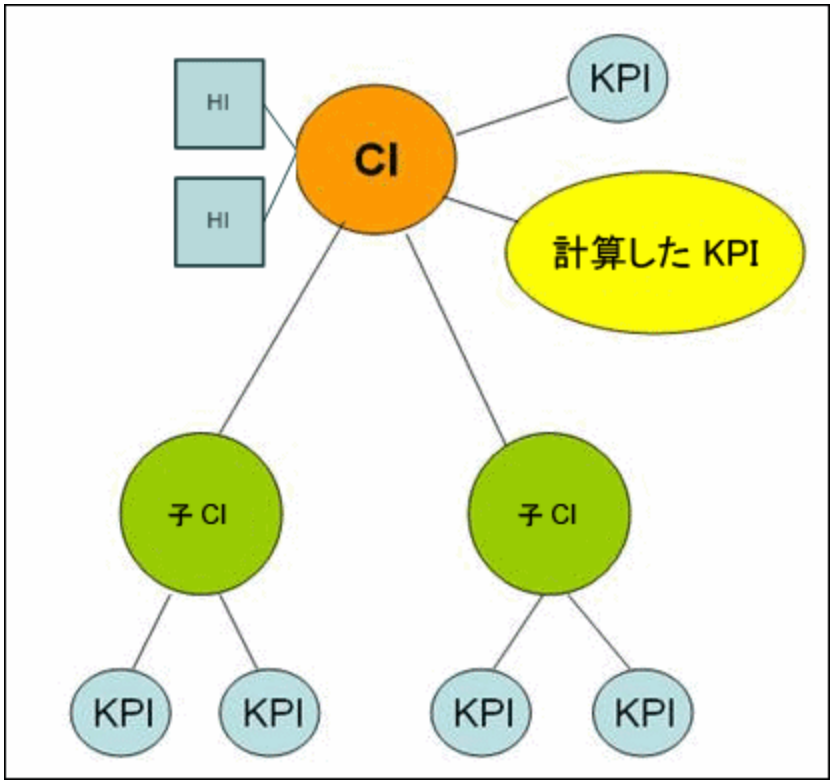
```
public void calculateKPI(CI ci, KPI kpi)
```

- **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\API_docs\Rules_API\index.html

グループと兄弟ルールの詳細な例については，「例 - API グループと兄弟ルール」(438ページ)を参照してください。

API ルールは，「ルール API を使用したルールの作成」(425ページ)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，「［CI インジケータ］タブでの API ルールの定義方法」(426ページ)または「ルール・リポジトリでの API ルールの定義方法」(431ページ)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，「例 - API グループと兄弟ルール」(438ページ)の「例 - 特定の子 CI グループ・ルール」(441ページ)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，「テキスト・ファイル・ベースの API ルールの作成方法」(427ページ)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，「例 - API サンプル・ルール」(435ページ)を参照してください。

API ルールは，「ルール API を使用したルールの作成」(425ページ)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\API_docs\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，「［CI インジケータ］タブでの API ルールの定義方法」(426ページ)または「ルール・リポジトリでの API ルールの定義方法」(431ページ)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，「テキスト・ファイル・ベースの API ルールの作成方法」(427ページ)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，「例 - API サンプル・ルール」(435ページ)を参照してください。

API ルールは，「ルール API を使用したルールの作成」(425ページ)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\API_docs\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）List of Rule Parameters 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，「［CI インジケータ］タブでの API ルールの定義方法」(426ページ)または「ルール・リポジトリでの API ルールの定義方法」(431ページ)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，「テキスト・ファイル・ベースの API ルールの作成方法」(427ページ)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。

詳細については，「［CI インジケータ］タブでの API ルールの定義方法」(426ページ)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 コードはテキスト・ファイル内でだけ表示，変更できます。サービス状況 内では表示も変更もできません。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。

詳細については，「テキスト・ファイル・ベースの API ルールの作成方法」(427ページ)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。

詳細については，「ルール・リポジトリでの API ルールの定義方法」(431ページ)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，『OMi 管理ガイド』のAdding KPIs to CIs, or Modifying KPI Settings - KPIs Tab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，「サービス状況ルール API」(419ページ)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。

   - 「API グループと兄弟ルール」(420ページ)
   - 「API サンプル・ルール」(422ページ)
   - 「API 継続時間ベースのサンプル・ルール」(424ページ)

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 コードはテキスト・ファイル内でだけ表示，変更できます。サービス状況 内では表示も変更もできません。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。

   - 「API グループと兄弟ルール」(420ページ)
   - 「API サンプル・ルール」(422ページ)
   - 「API 継続時間ベースのサンプル・ルール」(424ページ)

   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，『OMi 管理ガイド』のCustomizing KPI and HI Calculation Rules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）。

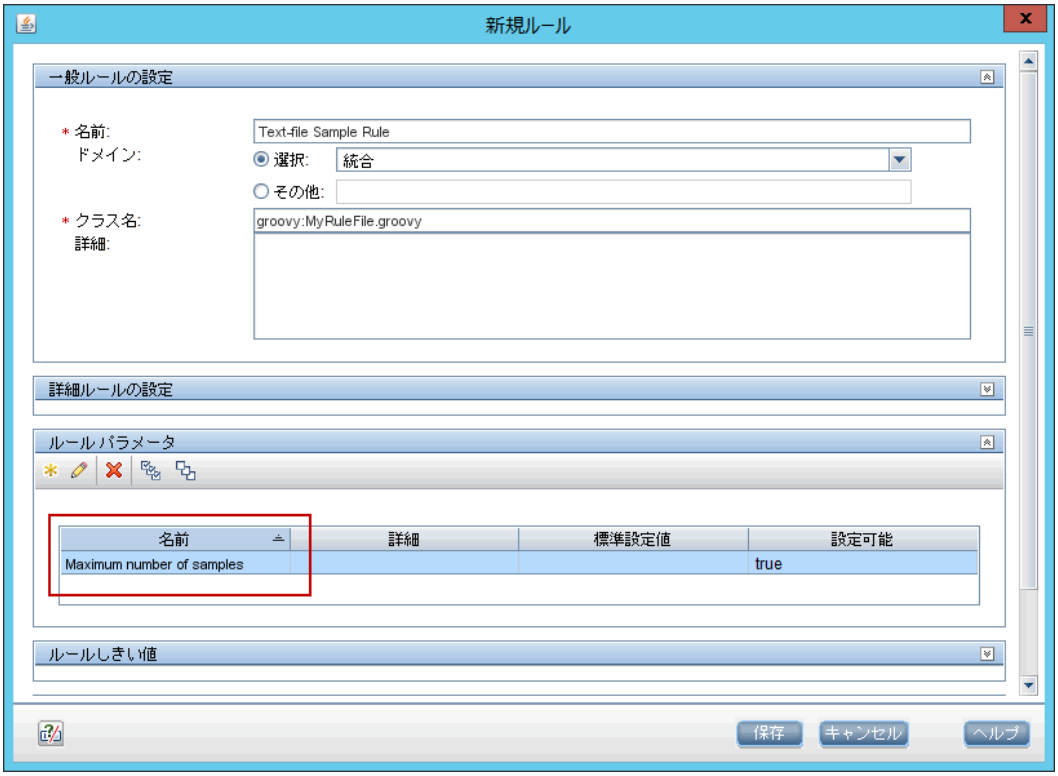
      - **しきい値パラメータ**領域で，［**新規作成**］をクリックします。
      - ［**名前**］フィールドに「**critical**」と入力します。［**タイプ**］フィールドで［**Float**］を選択します。

        ［**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。

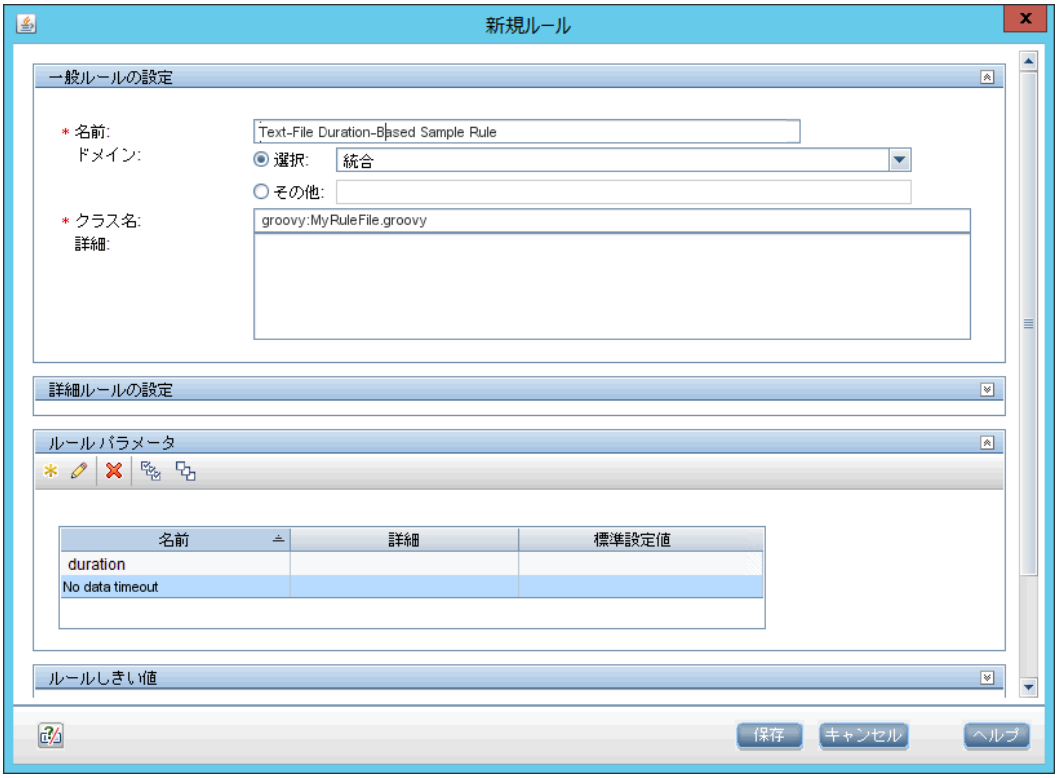
      - ［**OK**］をクリックして保存します。

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，『OMi 管理ガイド』のNew KPI/Edit KPI Dialog Box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，「ツールチップ・エントリの使用方法」(432ページ)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. **OMi-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，『OMi 管理ガイド』のCustomizing KPI and HI Calculation Rules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。

      - 「API グループと兄弟ルール」(420ページ)
      - 「API サンプル・ルール」(422ページ)

- 「API 継続時間ベースのサンプル・ルール」(424ページ)

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\API_docs\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，『OMi 管理ガイド』のNew KPI/Edit KPI Dialog Box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. ビジネス・ルールにアクセスします。

   ［管理］ > ［サービス状況］ > ［リポジトリ］ > ［ビジネス ルール］

   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
ステータスの最終変更	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，『OMi 管理ガイド』のCustomizing Tooltips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 <*OMi_HOME_DPS*>/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，<*OMi_HOME_DPS*>/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
- BLE_ONLINE_ATTRIBUTE

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>
```

API グループと兄弟ルールを使用して CI の CI プロパティの値を取得するには，**marble_dashboard_tql** を再起動する必要があります。

1. RTSM JMX コンソール（<*OMi_HOME_DPS*>:21212/jmx-console/HtmlAdaptor?action=inspectMBean&name=UCMDB:service=TQL%20Services）にアクセスします。
2. **retrieveTqlNames**() を呼び出します。
3. **marble_dashboard_tql** を検索し，TQL を再起動します。

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 - 平均可用性ルール」(435ページ)
- 「例 - 平均パフォーマンス・ルール」(436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」(437ページ)

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) {
   // この計算サイクルのサンプルの総数を保持する。
   def totalSamples = samples.size();
   // 使用可能なサンプルを計算するための変数を作成する。
   def availableSamples = 0;
   /**
/**    * 所定のサンプルを調査する。サンプルの u_iStatus が 0 である場合，
   * サンプルを使用可能とみなす。
   */
   samples.each {Sample currentSample ->
   if (currentSample.u_iStatus == 0) {
      // 使用可能なサンプルの数に加算する。
      availableSamples++;
      }
   }
   if (totalSamples > 0) {
      // KPI 値を設定し，パーセンテージに変換する。
   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) {
   // 使用可能なサンプルを計算するための変数を作成する。
   def availableSamples = 0;
   // 使用可能なサンプルの応答時間を合計するための変数を作成する。
   def totalResponseTime = 0;
   /**
/**    * 所定のサンプルを調査する。サンプルの u_iStatus が 0 である場合，
   * サンプルを使用可能とみなす。
    */
   samples.each {Sample currentSample ->
      if (currentSample.u_iStatus == 0) {
         // 使用可能なサンプルの数に加算する。
         availableSamples++;
```

```
        // 現在のサンプルの dResponseTime の値を totalResponseTime に追加する。
        totalResponseTime += currentSample.dResponseTime
      }
    }
    if (availableSamples > 0) {
      // KPI 値を設定し，パーセンテージに変換する。
      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，『OMi 管理ガイド』のCustomizing Rule Parameters and Thresholds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) {
   // 使用可能なサンプルを計算するための変数を作成する。
   def availableSamples = 0;
   // 使用可能なサンプルの応答時間を合計するための変数を作成する。
   def totalResponseTime = 0;
   /**
    * ルール・パラメータ "Response time limit" の値を
    * サービス状況 管理の KPI に定義された KPI から取得する。
    * This rule parameter is optional, so responseTimeLimit can be null.
    */
   Long responseTimeLimit = kpi.getRuleParameter("Response time limit")
```

```
  /**
/**    * 所定のサンプルを調査する。サンプルの u_iStatus が 0 である場合，
  * サンプルを使用可能とみなす。
   */
  samples.each {Sample currentSample ->
    if (currentSample.u_iStatus == 0) {
      /**
      * ルール・パラメータの値を確認する。
      * null ではなく（ユーザが値を設定済み），
      * サンプルの dResponseTime がルール・パラメータの値より大きい場合，
      * 値は無効である。
      */
      boolean isSampleValid = true;
      if (responseTimeLimit != null) {
        // ResponseTime がルール・パラメータの値を超えるかどうかを確認する。
        if (currentSample.dResponseTime > responseTimeLimit) {
          // サンプルは無効である。
          isSampleValid = false;
        }
      }
      if (isSampleValid) {
        // 使用可能なサンプルの数に加算する。
        availableSamples++;
        // サンプルの dResponseTime の値を totalResponseTime に追加する。
        totalResponseTime += currentSample.dResponseTime
      }
    }
  }
  if (availableSamples > 0) {
    // KPI 値を設定し，パーセンテージに変換する。
    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」(439ページ)
- 「例 - 最悪の兄弟ステータス・ルール」(440ページ)

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) {
  // 計算した KPI のタイプ ID を取得する（サービス状況 KPI リポジトリの定義のとおり）。
  int kpiId = kpi.getType();
  // 計算した CI の子 CI のすべての KPI のリストを取得する。
  // 計算した CI の子 CI は計算した KPI と同じタイプになる。
  List<KPI> childKpiList = ci.getChildrenKPIsByID(kpiId);
  // アクティブなステータスが見つかった場合のみ，
  // 計算した KPI のステータスを設定するための変数を作成する。
  boolean isActiveStatusFound = false;
  // 現在のワースト・ステータスを OK に設定する。下回るステータスが見つかったら更新する。
  Status worstStatus = Status.OK;
  // 子 KPI のリストを調査する。
  childKpiList.each{KPI childKPI->
    // 子 KPI のステータスを取得する。
    Status childKpiStatus = childKPI.status;
    // 子 KPI のステータスがアクティブなステータスかどうかを確認する。
    if(childKpiStatus.isActive()){
      // アクティブなステータスが見つかったことをマークする。
      isActiveStatusFound = true;
      // 子 KPI のステータスが現在のワースト・ステータスを下回るかどうかを確認する。
      if(childKpiStatus.isWorse(worstStatus)){
        // ワースト・ステータスを更新する。
        worstStatus = childKpiStatus;
      }
    }
  }
  // アクティブなステータスが子 KPI で見つかったかどうかを確認する。
  if (isActiveStatusFound) {
    // 計算した KPI ステータスを設定する。
    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) {
    // CI のすべての KPI のリストを取得する。
    List<KPI> ciKpiList = ci.getAllKPIs();
    /**
     * アクティブなステータスが見つかった場合のみ，
     * 計算した KPI のステータスを設定するための変数を作成する。
     */
    boolean isActiveStatusFound = false;
    // 現在のワースト・ステータスを OK に設定する。下回るステータスが見つかったら更新する。
    Status worstStatus = Status.OK;
    // CI の KPI のリストを調査する。
    ciKpiList.each {KPI ciKPI ->
        /**
        * CI の KPI が計算した KPI でないことを確認する。
        * This is needed because getAllKPIs method returns all the KPIs for the CI.
        */
        if (ciKPI !=  kpi) {
            /**
             * ciKPI は計算した KPI の兄弟 KPI を表す。
             * Get the sibling KPI's status.
             */
            Status siblingKpiStatus = ciKPI.status;
            // 必要に応じて worstStatus を更新する。
            if (siblingKpiStatus.isActive()) {
                isActiveStatusFound = true;
                if (siblingKpiStatus.isWorse(worstStatus)) {
                    worstStatus = siblingKpiStatus;
                }
            }
        }
    }
    // 兄弟 KPI にアクティブなステータスが見つかったかどうかを確認する。
    if (isActiveStatusFound) {
        // 計算した KPI のステータスを設定する。
        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) {
    /**
    * 子 CI "tx_10 from virtual_host_3" の Availability KPI を取得する。
    * "tx_10 from virtual_host_3" の RTSM ID は "96c2df2b544683c7f79bb382d1d7b3a9" である。
    *
    * 注 :UI 内で，次の行に次のように書き込むことができる。
    * KPI childKPI = "tx_10 from virtual_host_3"."Availability"
    */
    KPI childKPI = ci.getChildKpiByChildId(KpiType.Availability, "96c2df2b544683c7f79bb382d1d7b3a9");

    // childKPI が null でないかどうかを確認する。この RTSM ID の子 CI が存在する場合や，この CI に可用性 KPI がない場合は null になる。
    if (childKPI != null) {

        // childKPI に値が設定されているかどうかを確認する。
        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) {
   /**
    * タイプが Availability の兄弟 KPI を取得する。
    * If Availability KPI does not exist, null will be returned.
    */
   KPI availabilityKPI = ci.getKPI(KpiType.Availability);
   // タイプが Performance の兄弟 KPI を取得する。
   KPI performanceKPI = ci.getKPI(KpiType.Performance);
   if (availabilityKPI != null && performanceKPI != null) {
      // この CI では両方の KPI が存在する。Check if the KPIs status is active.
      if (availabilityKPI.status.isActive() && performanceKPI.status.isActive()) {
         // KPI のステータスを確認する。
         if (availabilityKPI.status == Status.OK &&
            performanceKPI.status == Status.OK) {
            /**
             * 両方のステータスがアクティブで，両方が OK である。Set this KPI's status to OK.
             */
            kpi.status = Status.OK
         }
         else {
            /**
             * 両方のステータスがアクティブだが，両方が OK ではない。
             * この KPI のステータスを CRITICAL に設定する。
             */
            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) {
   // 計算した KPI のタイプ ID を取得する（サービス状況 KPI リポジトリの定義のとおり）。
   int kpiId = kpi.getType();
   // 計算した KPI と同じ CI タイプで，CI タイプが "bpm_tx_from_location" の
   // 子 CI の KPI のリストを取得する。
   List<KPI> bpmTxFromLocationChildKpiList = ci.getChildrenKPIsByIDAndCiType(kpiId, "bpm_tx_from_
location")
   // 子 KPI から合計値を計算するための変数を作成する。
   // If no child exists or no child has value the variable will remain null.
   Double totalChildValue = null;
   // ログ・ファイルに情報を書き込む。
   logger.debug("DashboardGroupAvgValueByCiTypeRule :number of child CIs with type bpm_tx_from_
location:" + bpmTxFromLocationChildKpiList.size())
   // 子 KPI のリストを調査する。
   bpmTxFromLocationChildKpiList.each {KPI childKPI ->
      // null 値を処理する Utils クラスを使用して子 KPIの値を合計する。
      totalChildValue = Utils.sum(totalChildValue, childKPI.value);
   }
   // Utils クラスを使用して，計算した KPI の値を平均値に設定する。
   // totalChildValue が null の場合，null 値が設定される。
   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) {
   // すべての状況インジケータを取得する。
   List<HI> his = ci.getHIs();
   // アクティブなステータスが見つかった場合のみ，
   // 計算した KPI のステータスを設定するための変数を作成する。
   boolean isActiveStatusFound = false;
   // 現在のワースト・ステータスを OK に設定する。
```

```
    // 下回るステータスがある場合，更新する。
    Status worstHiStatus = Status.OK;
    his.each {HI hi ->
        Status hiStatus = hi.getStatus();
        // 現在の HI ステータスがアクティブなステータスかどうかを確認する。
        if (hiStatus.isActive()) {
            // アクティブなステータスが見つかったことをマークする。
            isActiveStatusFound = true;
            // 子 KPI のステータスが現在のワースト・ステータスを下回るかどうかを確認する。
            if (hiStatus.isWorse(worstHiStatus)) {
                // ワースト・ステータスを更新する。
                worstHiStatus = hiStatus;
            }
        }
    }
    // アクティブなステータスが子 KPI で見つかったかどうかを確認する。
    if (isActiveStatusFound) {
        // 計算した KPI ステータスを設定する。
        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) {
    /**
     * Groovy Closure を CI クラス getChildrenKPIs メソッドと使用して，
     * KPI のリストを CI の子 CI から取得する。
     * 1. KPI タイプは Availability
     * 2. ステータスは MAJOR

     Closure の説明 :
     { KPI childKPI ->
       childKPI.type == KpiType.Availability.getID("DASHBOARD") && childKPI.status == Status.MAJOR
     }
     Closure はタイプが KPI のパラメータ childKPI を定義する。
     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.
    Closure ボディが true を返す各 KPI は，返されたリストの一部になる。
    式 KpiType.Availability.getID("DASHBOARD") は サービス状況 KPI リポジトリから Availability KPI ID を表す int を返す。
    */

   List<KPI> kpiList = ci.getChildrenKPIs {KPI childKPI ->
      childKPI.type == KpiType.Availability.getID("DASHBOARD") && childKPI.status == Status.MAJOR
   }
   // KPI が存在するかどうかを確認する。
   if (kpiList.isEmpty()) {
       // KPI が存在しない。
      // デバッグ・レベルでログ・ファイルに書き込む。
      logger.debug "Closure Rule:no Availability KPI with  MAJOR status exist"
   }
   else {
      // ステータスが MAJOR の Availability KPI が少なくとも 1 つ存在する。
      logger.debug("Closure Rule:At least one Availability KPI  with  MAJOR status exist")
      // 計算した KPI ステータスを CRITICAL に設定する。
      kpi.status = Status.CRITICAL;
   }
}
```

第38章: サービス状況の外部 API

本項の内容

- 「インジケータ・データの取得 API」(446ページ): 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。
- 「状況インジケータの状態リセット API」(452ページ): 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。
- 「サービス状況のデータベース・クエリ API」(453ページ): 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

- 「KPI の経過時間ごとのステータスの取得」(446ページ)
- 「KPI 定義の取得」(449ページ)
- 「インジケータ・ステータスの取得」(450ページ)

サービス・ログ・ファイルは，次の場所にあります。**<OMi_HOME_GW>/log/jboss/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 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）。
- **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，『OMi 管理ガイド』のList of サービス状況 KPIs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「KPI 定義の取得」(449ページ)を参照）
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/<カスタマ ID>/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，『OMi 管理ガイド』のList of サービス状況 KPIs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，『OMi 管理ガイド』のList of サービス状況 KPIs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/<カスタマ ID>/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，『OMi ユーザ・ガイド』のHealth Indicator Componentを参照してください。

HI の状態リセット API を使用すれば，OMi 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）
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。**<OMi_HOME_ GW>/log/jboss/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 Operations Manager i 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	OMi カスタマは 1 を指定する必要があります。
userName	OMi 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。
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。標準設定値は **1**。
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XI部: ダウンタイム REST サービス

ゲートウェイ・サーバで実行されている RESTful Web サービスを使用して，ダウンタイムを取得，更新，作成，削除できます。REST クライアントの HTTP 要求や，HTTP 要求と XML コマンドの組み合わせをブラウザに入力できます。サービス認証は，基本認証に基づいています。

ダウンタイムの詳細については，『OMi 管理ガイド』のDowntime Management Overviewを参照してください。

注: ダウンタイム REST API を使用するには，管理権限かスーパーユーザ権限が必要です。

サポートされている HTTP 要求

ダウンタイム REST サービスでは，次の HTTP 要求がサポートされています。

注: CustomerID は常に 1 です。

アクション	HTTP コマンド
すべてのダウンタイムを取得	http:// < HP OMi サーバ > /topaz/bsmservices/customers/[customerId]/downtimes
特定のダウンタイムを取得	http:// < HP OMi サーバ > /topaz/bsmservices/customers/[customerId]/downtimes/[downtimeId]
http PUT を使用してダウンタイムを更新	http:// < HP OMi サーバ > /topaz/bsmservices/customers/[customerId]/downtimes/[downtimeId] + ダウンタイムの XML
http POST を使用してダウンタイムを作成	http:// < HP OMi サーバ > /topaz/bsmservices/customers/[customerId]/downtimes + ダウンタイムの XML **注 :** ダウンタイムの作成に成功すると，新しく作成されたダウンタイムが XML 形式で返されます。これには，ダウンタイム ID が含まれています。
http DELETE を使用してダウンタイムを削除	http:// < HP OMi サーバ > /topaz/bsmservices/customers/[customerId]/downtimes/[downtimeId]

許可されているダウンタイム・アクション

次のダウンタイム・アクション用にリストされている XML コマンドを使用します。

アクションの詳細	XML コマンド
アクションは必要ありません	<action name="REMINDER">
警告の非表示イベントを閉じる	<action name="SUPPRESS_NOTIFICATIONS"/>
KPI 計算のダウンタイム実行，警告の非表示イベントを閉じる（監視は継続）。	<action name="ENFORCE_ON_KPI_CALCULATION"/>
レポートおよび KPI 計算のダウンタイム実行，警告の非表示イベントを閉じる（監視は継続）	<action name="ENFORCE_ON_REPORTS"/>
レポートおよび KPI 計算のダウンタイム実行，警告の非表示イベントを閉じる（監視は継続）。すべての SLA が含まれます。	<action name="ENFORCE_ON_REPORTS"> <propGroup name="SLA" value="ALL"/> </action>
レポートおよび KPI 計算のダウンタイム実行，警告の非表示イベントを閉じる（監視は継続）。特定の SLA が含まれます。	<action name="ENFORCE_ON_REPORTS"> <propGroup name="SLA" value="SELECTED"> <prop>dda3fb0b20c0d83e078035ee1c005201</prop> </propGroup> </action>
アクティブ モニタリングの停止（BPM および SiteScope），レポートおよび KPI 計算のダウンタイム実行，警告の非表示イベントを閉じる	<action name="STOP_MONITORING"/>

ダウンタイム XML の例

次のフィールドは，指定した最大長を超えることはできません。

- 名前 :200 文字
- 説明 :2000 文字
- 承認者 : 50 文字

> **注:** Oracle で東アジアの言語（中国語，日本語，または韓国語）を使用する場合，最大文字数は上の指定数未満にすることができます。

```
<downtime userId="1" planned="true" id="8898e5a5dbcdc953e04037104bf5737c">
   <name>The name of the downtime</name>
   <action name="ENFORCE_ON_REPORTS">
   </action>
   <approver>The approver name</approver>
```

```
  <category>1</category>
  <notification>
    <recipients>
      <recipient id="24"/>
      <recipient id="22"/>
      <recipient id="21"/>
    </recipients>
  </notification>
  <selectedCIs>
    <ci>
      <id>ac700345b47064ed4fbb476f21f95a76</id>
      <viewName>End User Monitors</viewName>
    </ci>
  </selectedCIs>
  <schedule xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="WeeklyScheduleType">
    <type>WEEKLY</type>
    <startDate>2010-06-10T15:40:00</startDate>
    <timeZone>Europe/Zurich</timeZone>
    <days>
      <selectedDays>WEDNESDAY</selectedDays>
      <selectedDays>THURSDAY</selectedDays>
      <selectedDays>FRIDAY</selectedDays>
      <selectedDays>SATURDAY</selectedDays>
    </days>
    <startTimeInSecs>52800</startTimeInSecs>
    <durationInSecs>300</durationInSecs>
  </schedule>
</downtime>
```

第39章: ダウンタイムのスケジュール例

ダウンタイムのスケジュールを設定する場合は，次の点に留意してください。

- 以前のダウンタイムはサポートされない。次の操作はできない。
  - 過去にスケジュールされたダウンタイムを作成する。
  - 開始しているダウンタイムまたは過去に発生したダウンタイムを削除する。
  - 開始しているダウンタイムまたは過去に発生したダウンタイムを変更する。
- startDate / endDate の日付形式は **yyyy-MM-dd'T'HH:mm:ssZ**。
- 週次および月次のダウンタイムでは，startDate と endDate を深夜 0 時に定義する必要がある。例 :
  - <startDate>2010-07-24T00:00:00</startDate>
  - <endDate>2010-09-04T00:00:00</endDate>

1 回かぎりのダウンタイム・スケジュールの例

```
<schedule xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="OnceScheduleType">
  <type>ONCE</type>
  <startDate>2010-06-08T14:40:00</startDate>
  <endDate>2010-06-08T14:45:00</endDate>
  <timeZone>Asia/Tokyo</timeZone>
</schedule>
```

週次ダウンタイム・スケジュールの例

```
<schedule xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="WeeklyScheduleType">
  <type>WEEKLY</type>
  <startDate>2010-06-10T15:40:00</startDate>
  <timeZone>Europe/Zurich</timeZone>
  <days>
    <selectedDays>WEDNESDAY</selectedDays>
    <selectedDays>THURSDAY</selectedDays>
    <selectedDays>FRIDAY</selectedDays>
    <selectedDays>SATURDAY</selectedDays>
  </days>
  <startTimeInSecs>52800</startTimeInSecs>
  <durationInSecs>300</durationInSecs>
</schedule>
```

月次ダウンタイム・スケジュールの例

```
<schedule xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="MonthlyScheduleType">
  <type>MONTHLY</type>
  <startDate>2010-06-10T14:40:00</startDate>
  <timeZone>America/Montevideo</timeZone>
  <days>
    <selectedDays>WEDNESDAY</selectedDays>
    <selectedDays>THURSDAY</selectedDays>
    <selectedDays>FRIDAY</selectedDays>
    <selectedDays>SATURDAY</selectedDays>
  </days>
  <startTimeInSecs>52800</startTimeInSecs>
  <durationInSecs>300</durationInSecs>
</schedule>
```

第40章: ダウンタイム REST の例

ダウンタイム REST サービス API の使用する際に利用できるよう、次に Java コード例を示します。それぞれの例では，標準の Java コンポーネントのみを使用しています。サーバは，呼び出されるオペレーションごとに，クライアント側でのオペレーション検証に使用することができる HTTP コードを戻します。

次のリストに示す Java コード・ダウンタイム REST の例は，次のディレクトリに txt 形式で格納されています。

\\ < 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\DowntimeREST_JavaAPI\

- **CreateDowntime.java**

  これは，新規ダウンタイムを作成するための Java コードの例です。HTTP **POST** 要求を使用しています（REST サービスでは，POST 要求を使用してエンティティを作成します）。オペレーションが正常に実行されると，システムは新規作成されたダウンタイムを XML 形式で返します。これには，ダウンタイム ID が含まれています。

- **DeleteDowntime.java**

  これは，特定のダウンタイムを削除するための Java コードの例です。HTTP **DELETE** 要求を使用しています（REST サービスでは，DELETE 要求を使用してエンティティを削除します）。オペレーションが正常に実行されても，値は何も返されません。

- **GetAllDowntimes.java**

  これは，すべてのダウンタイムを取得するための Java コードの例です。HTTP **GET** 要求を使用しています（REST サービスでは，GET 要求を使用してエンティティを取得します）。オペレーションが正常に実行されると，システムはすべてのダウンタイムを XML 形式で返します。

- **GetSpecificDowntime.java**

  これは，特定のダウンタイムについてのすべての情報を取得するための Java ファイルの例です。HTTP **GET** 要求を使用しています（REST サービスでは，GET 要求を使用してエンティティを取得します）。オペレーションが正常に実行されると，システムは特定のダウンタイムを XML 形式で返します。

- **UpdateDowntime.java**

これは，ダウンタイムを更新するための Java コードの例です。HTTP **PUT** 要求を使用しています（REST サービスでは，PUT 要求を使用してエンティティを更新します）。オペレーションが正常に実行されても，システムは値を何も返しません。

第41章: 外部ソースからのダウンタイム・データのインポート

ビジネス管理ソリューション（HPSM，HPOM，サードパーティ・ソフトウェアなど）が OMi との統合時にダウンタイム・イベントを作成する場合は，外部システムからダウンタイム情報をインポートしなければならないことがあります。このダウンタイム情報をインポートするには，REST API を使用してミドル・ユーティリティを作成し，外部ソースからイベントを引き出し，それらのイベントを OMi に送信します。

外部ソースから定義をインポートする場合は，インポートのスコープとメカニズムの両方を考慮に入れてください。

インポートのスコープ

ダウンタイムのプロパティは，システムやソフトウェアのプラットフォームごとに異なります。一般的なダウンタイムのプロパティ・セットには，スケジュール情報や構成アイテムなどが含まれます。OMi ダウンタイムには，次のような必須フィールドがあります。

- ダウンタイム名
- CI ID
- スケジュール
- アクション

インポートしたイベントは，OMi ダウンタイムのプロパティに適合するように変換する必要があります。

インポート・メカニズム

OMi へのダウンタイムのインポートは，外部ソースのフォーマットおよびプロパティに対するアクセス権を持つ外部ユーティリティによって実行されます。このユーティリティは，XML 形式で外部のプロパティを変換して OMi ダウンタイムの必須および任意のプロパティに相関させます。

インポート例

Java および Groovy を REST API と組み合わせて使用して作成されたユーティリティの例は，次のサイトからダウンロードできます。

\\ < 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\DowntimeREST_JavaAPI\DTImport.zip

DTImport.zip ファイルには，次のファイルが格納されています。

- **DTImport.jar** - Groovy スクリプトを実行し，スクリプトの依存関係を決定する Java クラスが含まれています（主に，REST サービスへのアクセスに使用する HTTP クライアント）。
- **DTImport.bat** - Java アプリケーションを実行し，OMi URL，ユーザ名，パスワードを送信します。
- **downtimeFiles フォルダ** - ダウンタイム定義の XML ファイルが格納されています。
- **DTImport.groovy** - XML ファイルを読み取り，OMi REST サービスに送信する Groovy スクリプト。

DTImport.bat ファイルは，定義された統合フォルダ（システム・プロパティ integ.home）と一緒に DTImport Java アプリケーションを呼び出します。このアプリケーションは，Scripts フォルダ内のすべての Groovy スクリプトを読み取り，それらのスクリプトを呼び出します。例の DTImport.groovy スクリプトは，すべての dt[n].xml ファイルを読み取り，OMi REST サービスを使用して OMi でダウンタイムを作成します。このファイルの内容を変更して，OMi 9.1 以上との独自のカスタム統合を作成することができます。

ファイルを編集する場合には，次の点に留意してください。

- ユーティリティの論理を変更するには，Groovy スクリプトを編集し，バッチ・ファイルを再実行します。スクリプトをビルドしてコンパイルする必要はありません。
- 統合ディレクトリを変更するには，バッチ・ファイルの **integ.home** を編集します。
- **DTImport.groovy** スクリプトを，Scripts フォルダ内の他のスクリプトと置き換えることができます。
- メイン・メソッドを含む Groovy スクリプトのみを Scripts フォルダに追加します。
- 正しくフォーマットされた XML ファイルを downtimeFiles フォルダに追加します。
- 任意のスケジューラ（Windows タスク・スケジューラなど）を使用してスクリプトを実行します。

例を実行するには，次の手順を実行します。

1. クライアント・マシンに Java Runtime Environment がインストールされていることを確認します。
2. DTImport.zip ファイルを，統合を実行するサーバに抽出します。これは，OMi サーバ，外部ソース，またはほかの任意のサーバにすることができます。
3. DTImport.bat ファイルを編集して，OMi URL および資格情報を更新します。
4. downtimeFiles フォルダの XML ファイルを編集して，システム定義と適合するように変換するダウンタイムのパラメータを更新します。

5. 既存の受信者 ID を入力するには，次の手順を実行します。

    a. Web ブラウザで，JMX コンソールの URL（**http://localhost:29000/jmx-console/**）を入力します。

    b. ユーザ名とパスワードを入力します。

    c. **［通知サービスのテスト］ > showRecipients()** を実行して，受信者 ID を取得します。

6. 既存の CI ID を入力するには，次の手順を実行します。

    a. Web ブラウザに，DPS JMX コンソールの URL（**http:// < データ処理サーバの名前 > :21212/jmx-console**）を入力します。

    b. ユーザ名とパスワードを入力します。

    c. **［モデル サービス］ > retrieveObjectsOfType** を実行して，CIID を取得します。たとえば，CI がビジネス・アプリケーションの場合は，「business_application」と入力します。

7. バッチ・ファイルを実行します。

ドキュメントのフィードバックの送信

このドキュメントに関するご意見は，電子メールでドキュメント・チームまでお寄せください。このシステムで電子メール・クライアントが設定されている場合，上記のリンクをクリックすると，件名の行に次の情報を含む電子メール・ウィンドウが開きます。

OMi 拡張性ガイド（Operations Manager i 10.00）に関するフィードバック

フィードバックを電子メールに追加し，［送信］をクリックしてください。

電子メール・クライアントを使用できない場合は，上記の情報を Web メール・クライアントで新しいメッセージにコピーし，フィードバックを ovdoc-asm@hp.com に送信してください。

ご意見ありがとうございます。

OMi にアクセス!