Server Automation

ソフトウェアバージョン: 10.50

開発者ガイド

ドキュメントリリース日: 2016年7月（英語版）
ソフトウェアリリース日: 2016年7月

Hewlett Packard Enterprise

ご注意

保証

Hewlett Packard Enterprise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。ここでの記載は、追加保証を提供するものではありません。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、Hewlett Packard Enterprise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ーソフトウェアです。これらを所有、使用、または複製するには、Hewlett Packard Enterpriseからの有効な使用許諾が必要です。商用コンピューターソフトウェア、コンピューター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®は、Adobe Systems Incorporated (アドビシステムズ社) の登録商標です。

Microsoft®およびWindows®は、Microsoft Corporationの米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、以下の識別情報が記載されています。

- ソフトウェアバージョン番号: ソフトウェアバージョンを示します。
- ドキュメントリリース日: 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: このソフトウェアバージョンのリリース日を示します。

更新状況、およびご使用のドキュメントが最新版かどうかは、次のサイトで確認できます。https://softwaresupport.hpe.com/

このサイトを利用するには、HPE Passportへの登録とサインインが必要です。HPE Passport IDの登録は、HPEソフトウェアサポートサイトで **[Register]** をクリックするか、HPE Passportのログインページで **[Create an Account]** をクリックしてください。

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

サポート

HPEソフトウェアサポートサイトを参照してください。https://softwaresupport.hpe.com

このサイトでは、HPEのお客様窓口のほか、HPEソフトウェアが提供する製品、サービス、およびサポートに関する詳細情報をご覧いただけます。

HPEソフトウェアオンラインではセルフソルブ機能を提供しています。お客様のビジネスを管理するのに必要な対話型の技術サポートツールに、素早く効率的にアクセスできます。HPEソフトウェアサポートのWebサイトでは、次のようなことができます。

- 関心のあるナレッジドキュメントの検索
- サポートケースの登録とエンハンスメント要求のトラッキング
- 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
- HPE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

一部のサポートを除き、サポートのご利用には、HPE Passportユーザーとしてご登録の上、サイン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。HPE Passport IDを登録するには、HPEサポートサイトで **[Register]** をクリックするか、HPE Passportのログインページで **[Create an Account]** をクリックします。

アクセスレベルの詳細については、次のWebサイトをご覧ください。https://softwaresupport.hpe.com/web/softwaresupport/access-levels

HPE Software Solutions Nowでは、HPESWソリューションおよび統合ポータルWebサイトにアクセスできます。このサイトでは、お客様のビジネスニーズに合ったHPE製品ソリューションをご覧いただけます。また、HPE製品間の統合リストとITILプロセスのリストも用意しています。このWebサイトのURLはhttps://softwaresupport.hpe.com/です。

目次

開発者ガイドの概要

Server Automationプラットフォームは、SAの統合と拡張を容易にするためのAPIとランタイム環境のセットです。Server Automationプラットフォームは、監査コンプライアンス、Windowsパッチ管理、OSプロビジョニングなどのコアサービスを公開します。ランタイム環境は、Global File System (OGFS) にアクセス可能なGlobal Shellスクリプトを実行します。

ここでは、次の内容について説明します。

概要

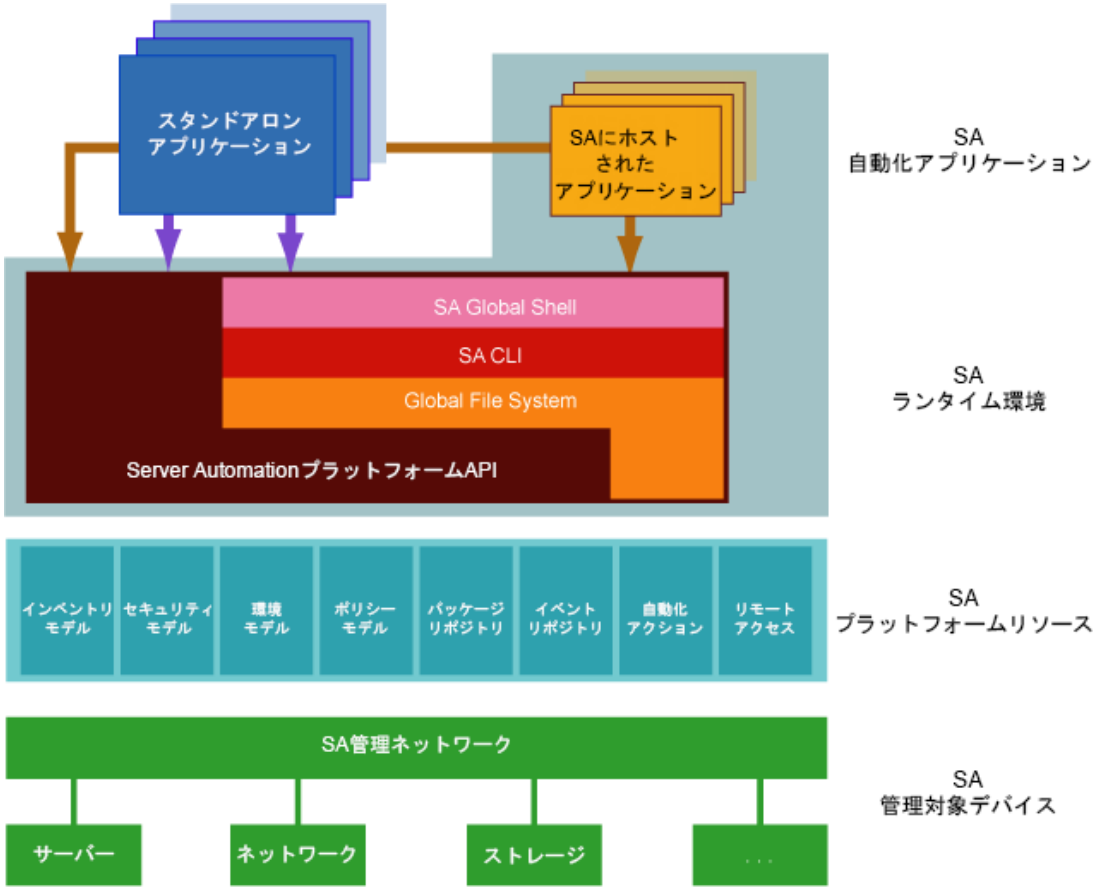
Server Automationプラットフォームを使用すると、次の作業を実行できます。

- 新しい自動化アプリケーションの構築やSAの拡張を通じて、ITの生産性向上やITポリシーへの適合を促進できます。
- 既存の監視、トラブルのチケット発行、課金、仮想化テクノロジーといった情報を他のITシステムと交換できます。
- SAモデルリポジトリを使用して、オペレーション、環境、資産といった重要なIT情報を記憶して組織化できます。
- さまざまなアプリケーションやオペレーティングシステムの管理を自動化できます。
- 既存のUnixおよびWindowsスクリプトをSAに組み込み、セキュアな監査対象の環境でスクリプトを実行することができます。

コンポーネント

次の図に、Server Automationプラットフォームの主な要素を示します。

Server Automationプラットフォームのコンポーネント



上の図に示すように、プラットフォームには次の5つの主要な要素があります。これらの要素については、この後の部分で詳しく説明します。

- **自動化アプリケーション:** ユーザーがプラットフォーム上に作成するアプリケーション。これらのアプリケーションは、SAにホストされたアプリケーション (実行中のSAのコンテキストで動作) またはスタンドアロンのアプリケーション (既存のビジネスおよび管理システムのコンテキストで動作) です。
- **ランタイム環境:** すぐに使用できる強力なランタイムサービスのセットと、それに対応する言語に依存しないプログラミングモデルを提供します。これらは、スクリプト作成者からWeb開発者、熟練した企業Javaプログラマーまで、さまざまなプログラマーから容易に利用できるように設計されています。
- **プラットフォームリソース:** プラットフォームのさまざまなデータオブジェクト、自動化アクション (パッチ適用、プロビジョニング、監査など)、機能 (各管理対象サーバーのランタイム環境へのリモートアクセスなど) を開発者が容易に利用できるようにします。
- **SA管理ネットワーク:** 強力な接続機能、セキュリティ機能、キャッシングテクノロジーのセットで、プラットフォームが場所、IPアドレス空間、利用可能帯域幅などの制約と無関係にデバイスに到達できるようにします。
- **SA管理対象デバイス:** SA管理ネットワークによってプラットフォームに接続される管理対象サーバーおよびネットワークデバイス。

自動化アプリケーション

上の図に示すように、自動化アプリケーションはスタックの最上部にあります。これらは、ユーザーがプラットフォーム上に作成するアプリケーションです。

自動化アプリケーションは、SAにホストされたアプリケーション (SAランタイム環境で動作) またはスタンドアロンのアプリケーション (完全に独立したコンテキストで動作) です。スタンドアロンのアプリケーションは、Webサービス呼び出しを通じてプラットフォームにリモートアクセスします。

単純なアプリケーションなら、Unixシェルスクリプトを使ってごく短時間で作成できます。もっと複雑なアプリケーション、たとえば既存のソース管理システムやチケット発行システムとの統合といったものは、もう少し時間がかかり、Python、Microsoft .NET、Javaなどによるコーディングが必要になる可能性があります。いずれの場合でも、プラットフォームは言語に依存しないシステムとして設計されているので、さまざまな開発者が容易に採用できます。

SAランタイム環境

プラットフォームスタックでその下にあるのがSAランタイム環境です。これは、すぐに使用できる強力なランタイムサービスのセットと、それに対応する言語に依存しないプログラミングモデルを提供します。SAにホストされるアプリケーションは、SAランタイム環境で動作します。

ランタイム環境のコアは、Global ShellとGlobal File Systemの2つのコンポーネントから構成されます。これら2つのコンポーネントの組み合わせにより、すべての管理対象デバイスが、なじみ深いLinux/Unixシェルのファイル/ディレクトリ構造で整理され、アクセスできるようになります。

Global Shell

Global Shellは、Global File System (OGFS) へのコマンドラインインタフェースです。コマンドラインインタフェースは、ターミナルウィンドウで動作するbashなどのLinuxシェルを通じて使用できます。OGFSは、SAデータモデルと管理対象サーバーの内容 (ファイルなど) を1つの仮想ファイルシステムに統合します。

Global File System

OGFSは、プラットフォームデータモデルのオブジェクト (ファシリティ、カスタマー、デバイスグループなど) と、プラットフォームの管理対象デバイスで使用可能な情報 (管理対象ネットワークデバイスの構成設定や管理対象サーバーのファイルシステムなど) を、ファイルディレクトリとテキストファイルの階層構造で表現します。たとえば、OGFSの/opsw/Customerディレクトリにはカスタマーオブジェクトの詳細が、/opsw/Serverディレクトリには管理対象サーバーに関する情報が記録されています。また、/opsw/Serverディレクトリに

は、管理対象サーバーの内容 (ファイルシステムやレジストリ) を反映するサブディレクトリも含まれています。

このファイル/ディレクトリ構造を使用することで、シェルスクリプトになじんだ管理者は、サーバーを表すディレクトリを反復処理することで同じ作業を複数のサーバーに対して実行するスクリプトを容易に作成できます。Global File Systemは、スクリプトのロジックを各管理対象サーバーに対してセキュアに配布して実行します。

デバイスの内容にアクセスするには、SAとNetwork Automation (NA) で管理されるすべてのデバイスを表す仮想ファイルシステムであるGlobal File Systemを使用します。エンドユーザーと自動化アプリケーションの両方が、必要なセキュリティ承認を経て、OGFSを通じてリモートサーバーのファイルシステムにアクセスできます。Windowsサーバーでは、管理者はレジストリ、IIメタベース、COM+オブジェクトにもアクセスできます。

SAコマンドラインインタフェース

SAコマンドラインインタフェース (CLI) は、システム管理者やプラットフォーム自動化アプリケーションが、ソフトウェアのプロビジョニング、デバイスへのパッチ適用、監査の実行といった自動化タスクをコマンドラインから起動するために使用できます。高機能の構文により、さまざまなオブジェクトタイプを、CLI呼び出しの入力に使用したり、出力として受け取ったりすることができます。

実際には、CLI自体は、次の項で説明するプラットフォームAPIの上に、プログラムによって生成されます。この方法の利点は、開発者によって新しいAPIがプラットフォームAPIに追加されたときに、それに対応するCLIメソッドが自動的に使用可能になることです。すなわち、製品で新機能が使用可能になってから、対応するCLIメソッドがプラットフォームで使用可能になるまでに、遅れが発生しないということです。

SAプラットフォームAPI

SAプラットフォームAPIは、SAのWin32 APIです。値の取得と設定や、アクションの実行のためのアプリケーションプログラミングインタフェースが定義されています。SAクライアントやSAコマンドラインインタフェース (CLI) といったSAのユーザーインタフェースは、すべてSAプラットフォームAPIの上に構築されています。APIには、Java RMIクライアント向けのライブラリと、SOAPベースのWebサービスクライアント向けのWSDLが含まれています。Webサービスのサポートにより、プログラマーはPerl、C#、Pythonといった一般的な言語でクライアントを作成できます。

SAプラットフォームリソース

SAプラットフォームリソースは、SAランタイム環境の下にあって、開発者がさまざまなオブジェクトやアクションのセットにアクセスし、自分のアプリケーションで再使用したり操作したりするために使用されます。

インベントリモデル

インベントリモデルは、各管理対象デバイスに対してSAが収集するすべての情報 (メーカー、製造者、CPU、オペレーティングシステム、インストール済みソフトウェアなど) を提供します。インベントリ情報は、SA APIを通じて入手できるとともに、Global File Systemのファイル (`attr`サブディレクトリの下) としても表示されます。インベントリモデルには、サーバーやネットワークデバイスなどのオブジェクトが含まれます。

管理者は、インベントリオブジェクトに関連付けられたデータを拡張できます。たとえば、ユーザーがデバイスの画像や、リース有効期限や、デバイスが接続されているUPSのIDなどを保存したい場合、このような属性を各デバイスレコードに容易に追加できます。ユーザーは、これらの属性の追加、削除、操作を、標準の属性の場合と同様に行うことができます。

セキュリティモデル

セキュリティモデルを使うことにより、開発者は、SAが備える認証と承認のセキュリティシステムを利用できます。

プラットフォームのすべてのクライアント (管理アプリケーション、スクリプト、SAのエンドユーザーインタフェース) は、同じセキュリティフレームワークで管理されます。

ユーザーの役割を作成し、アクセス権を付与するのは、開発者でなくセキュリティ管理者です。開発者は、自分のアプリケーションのコンテキスト内で、これらすべてのユーザーの役割とアクセス権を再使用できます。たとえば、ネットワーク管理者がシェルスクリプトを作成し、他のネットワーク管理者と共有する場合に、共有相手のネットワーク管理者は、自分が管理する権限を持つネットワークデバイス以外に対してそのスクリプトを実行できないことが保証されます。

承認メカニズムは、いくつかのレベルでアクセスを制御します。レベルとしては、ユーザーが実行できるタスク、タスクからアクセスされるサーバーとネットワークデバイス、SAオブジェクト (ソフトウェアポリシーなど) があります。

環境モデル

環境モデルは、デバイスが存在するビジネスコンテキスト全体を定義します。一般的に、デバイスは1つまたは複数のカスタマーに属し、特定のファシリティに存在し、1つまたは複数のグループに属します。プラットフォームは、これらのオブジェクト (カスタマー、ファシリティ、デバイスグループなど) をアプリケーション開発者から使用可能にします。

インベントリオブジェクトと同様、環境オブジェクトも容易に拡張できます。これにより、たとえば、特定のデータセンターで用いられるSNMPトラップレシーバー、特定のファシリティでのみ使用可能なプリンター、特定のビジネスユニットのみで使用されるApache構成といった属性の定義が容易になります。

ポリシーモデル

ポリシーモデルを使用することで、開発者はSAで定義されているすべてのベストプラクティスにアクセスできます。ポリシーは、サーバーまたはネットワークデバイスの適切な状態を記述します。たとえば、パッチポリシーはサーバー上に存在すべきパッチを、ソフトウェアポリシーはサーバー上に存在すべきソフトウェアを記述します。

これらのポリシーは当該分野の専門家が定義し、承認されたシステム管理者は、これらのポリシーを使用してデバイスを監査することによって、デバイス上に実際に存在するものが存在すべきものと異なっているかどうかを判定できます。プログラマーは、すべてのポリシーのライブラリを自分のアプリケーションから利用できます。

ソフトウェアポリシーはフォルダーによって整理され、これによってセキュリティ境界が定義されます。すなわち、アプリケーションは、ユーザーアクセス権に基づいてアクセスが許可されているソフトウェアポリシーだけにアクセスできます。

パッケージリポジトリ

パッケージリポジトリを使用することで、開発者はSAに記憶されているすべてのソフトウェアとパッチにアクセスできます。これには、オペレーティングシステムビルド、オペレーティングシステムパッチ、ミドルウェア、エージェント、およびユーザーがSAにアップロードしたすべてのソフトウェアが含まれます。

イベントリポジトリ

イベントリポジトリは、アクションが(ユーザーインタフェースを通じて、またはプログラムからプラットフォームによって)実行されたときにSAが生成するデジタル署名付き監査証跡を格納します。他のプラットフォームオブジェクトと同様、これらのイベントはプログラムから取得できます。

自動化アクション

自動化アクションを使用することで、開発者は、SAが管理対象デバイスに対して実行できるすべてのアクションをプログラムから起動できます。これには、監査の実行、ソフトウェアのプロビジョニング、最新のOSパッチの適用などが含まれます。

プラットフォームは、エンドユーザーがSAクライアントで使用できるのと同じ機能へのアクセスを提供します。これには、パッチのインストール、オペレーティングシステムのプロビジョニング、ソフトウェアポリシーのインストールと削除などの作業が含まれます。実際には、SAクライアントはSAランタイム環境を通じてプログラムから利用できるのと同じAPIを使用しています。

リモートアクセス

リモートアクセスを使用することで、開発者は、管理対象デバイスのファイルシステム(サーバーの場合)および実行環境(すべてのデバイスの場合)にプログラムからアクセスできます。開発者は、ファイルまたは特定のソフトウェアパッケージの存在をチェックしたり、オペレーティングシステムコマンドを実行してディスクの使用状況をチェックしたり、システムスクリプトを実行して日常のメンテナンス作業を実行したりするアプリケーションを容易に作成できます。

SA管理ネットワーク

管理ネットワークは、開発者が管理対象の任意のデバイスにセキュアにアクセスできるようにするための強力なテクノロジーの組み合わせです。管理ネットワークが提供する主なサービスは次のとおりです。

- **接続**: プラットフォーム (およびそれを使用する自動化アプリケーション) から任意の管理対象デバイスに到達できるようにします。
- **セキュリティ**: SSL/TLSベースの暗号化、認証、メッセージの完全性を含みます。
- **アドレス空間の仮想化**: 重なり合う複数のIPアドレス空間内でプラットフォームがサーバーを発見できるようにします。複雑な企業ネットワークのほとんどには、複数のプライベートIPアドレス空間が存在します。
- **可用性**: システムアーキテクチャーで、管理対象デバイスへの冗長経路を定義して、どれかのネットワーク経路に障害が起きてもデバイスに到達できるようにすることができます。
- **キャッシング**: サーバーが遠くのサーバーでなく近くのサーバーからソフトウェアやパッチをダウンロードできるようにすることで、時間とネットワーク接続のコストを節約します。
- **帯域幅スロットリング**: システムアーキテクチャーで、SAとSAアプリケーションがネットワーク経由で特定のデバイスと通信するときに消費できる帯域幅を決定できるようにします。
- **最小コストルーティング**: システムデザイナーが、特定のデバイスに到達するために使用する経路を決定するルールを設定することで、ネットワーク接続のコストを最小化できるようにします。

SA管理対象デバイス

プラットフォームスタックの最下部にあるのは、実際の管理対象デバイスです。プラットフォームは、65種類以上のサーバーOSバージョンと、35社以上のネットワークデバイスベンダーを管理し、標準でサポートされるデバイスのモデル/バージョン数は数千に及びます。

サポートされるデバイスのリストは常に更新されています。このリストは、プラットフォーム開発者やスクリプト作成者にとって直接的な利益があります。使い慣れた同じプラットフォームプログラミング環境を使いながら、自動化アプリケーションから到達できる管理対象デバイスを常に増やし続けることができるからです。

SAプラットフォームの利点

SAプラットフォームの主な利点を次に示します。

強力なセキュリティ

次のような包括的なセキュリティメカニズムがプラットフォームから提供されるので、アプリケーションで独自にセキュリティを実現する手間がかかりません。

- **セキュアな通信チャネル**: 自動化アプリケーションから管理対象デバイスへのエンドツーエンドの通信は暗号化され、認証されます。
- **役割ベースのアクセス制御**: プラットフォームはSAに備わっている役割ベースのアクセス制御に従うので、開発者がアプリケーションを共有する場合、管理者がアクセス権を付与されているデバイスに対してだけそのアプリケーションが実行可能であることが保証されます。
- **デジタル署名付き監査証跡**: 自動化アプリケーションが実行された後で、プラットフォームは、アプリケーションの実行者、実行時刻、対象デバイスを記録したデジタル署名付き監査証跡を生成します。
- **包括的な到達範囲**: プラットフォームはすべてのデバイスに対する包括的な到達範囲を実現するので、システム管理者や開発者は、デバイスへの到達方法について悩む必要はありません。
- **業界最大の対象プラットフォーム範囲**: サポートされるデバイスには、65種類以上のサーバーOSバージョンと、1,000種類以上のネットワークデバイスが含まれます。
- **任意の物理的場所**: デバイスは、大規模データセンター、小売店、サテライトオフィスなど、世界中のどこに存在していてもかまいません。
- **任意のIPアドレス空間**: プラットフォームは重なり合う複数のIPアドレス空間をサポートするので、デバイスはどのIPアドレス空間に属していてもかまいません。
- **DMZ**: デバイスがDMZなどのアクセス困難なネットワーク空間に存在する場合でも、開発者やシステム管理者は、デバイスへの到達方法の詳細 (要塞ホスト経由など) について悩む必要はありません。

多彩なサービス

プラットフォームからは、元になる自動化システムのほぼすべての関連データおよびアクションを利用できます。

- **多彩な標準データ**: 開発者は、プラットフォーム自体から生成されるデータ (デバイスインベントリデータやファシリティ情報など) と、プラットフォームを使用するユーザーによって生成されたデータ (デバイスグループカスタマー、ベストプラクティスポリシー、アップロードされたソフトウェア、パッチ、スクリプトなど) を含む多彩なデータに容易にアクセスできます。開発者は、このデータを読み書きするアプリケーションを容易に作成できます。
- **拡張可能なデータストア**: 開発者は、ネイティブプラットフォームオブジェクトを容易に拡張して、独自のデータを追加することができます。デバイスインベントリモデルを拡張することにより、プラットフォームでネイティブに検出されない属性を含めることができます。カスタマーオブジェクトやファシリティオブジェクトを拡張して、そのカスタマーに関連するデバイスのプロビジョニングや監査の参考となる属性を含めることができます。

- **自動化タスク**: プラットフォームは、元になる自動化システムのほぼすべての機能を開発者に公開しています。これには、パッチ適用、プロビジョニング、監査などが含まれます。これにより、複数のシステムにまたがる複雑なワークフローを作成する場合でも、これらのアクションを自動化アプリケーションのコンテキストから簡単に呼び出すことができます。

さまざまなプログラマーから容易に利用可能

プラットフォームは、Unixシェルスクリプトや Visual Basicスクリプトの作成者、PerlやPythonのプログラマー、企業の.NETまたはJavaプログラマーなど、広範囲の開発者に便利なように設計されています。プラットフォームのランタイムサービスレイヤーでは、ほとんどのプラットフォームオブジェクトがファイル/ディレクトリ構造で利用でき、ほとんどのプラットフォームサービスがコマンドラインインタフェース (SA CLI) を通じて使用できます。このため、シェルスクリプトの使用に慣れたシステム管理者は、新しいプログラミング言語やツールを習得しなくても、すぐにプラットフォームを利用できます。使い慣れたテキストエディターとUnixシェルを使って、すぐにスクリプトを開発できます。

もっと複雑なアプリケーションや既存のシステムとの統合が必要な場合、システムプログラマーは、Web サービスバインディングを備えた任意のプログラミングツールや言語を使用できます。

SAプラットフォームAPIの設計

プラットフォームAPIは、Javaインタフェースで定義され、Javaパッケージに組織化されています。さまざまなクライアント言語やリモートアクセスプロトコルをサポートするため、APIは関数方式の値呼び出しモデルを採用しています。

サービス

プラットフォームAPIでは、関連する関数のセットが1つのサービスにまとめられています。各サービスは、名前の末尾が`Service`であるJavaインタフェースによって指定されます。例としては、`ServerService`、`FolderService`、`JobService`などがあります。

サービスは、APIへのエントリポイントです。APIにアクセスするには、クライアントはサーバーインタフェースで定義されたメソッドを呼び出します。たとえば、管理対象サーバー上にインストールされているソフトウェアのリストを取得するには、クライアントは`ServerService`インタフェースの`getInstalledSoftware`メソッドを呼び出します。他の`ServerService`メソッドの例としては、`checkDuplex`、`setPrimaryInterface`、`changeCustomer`があります。

SAプラットフォームAPIには70以上のサービスが含まれているので、ここではすべてを記述できません。次の表に示すのは、最初に試してみることをお勧めするいくつかのサービスです。すべてのサービスの一覧については、「APIドキュメントとTwister」(37ページ)に示すURLにブラウザーでアクセスしてください。

SAAPIのサービスの一部

サービス名	サービスで提供される操作の一部
AuditTaskService	監査タスクの作成、取得、実行。
ConfigurationService	アプリケーション構成の作成、アプリケーション構成を使用するソフトウェアポリシーの取得。
DeviceGroupService	デバイスグループの作成、グループへのデバイスの割り当て、グループのメンバーの取得、動的ルールの設定。
EventCacheService	値オブジェクトのクライアント側キャッシュの更新などのアクションのトリガー。「イベントキャッシュ」(35ページ)を参照してください。
FolderService	フォルダーの作成、フォルダーの子の取得、フォルダーのカスタマーの設定、フォルダーの移動。
InstallProfileService	OSインストールプロファイルの作成、取得、更新。

SAAPIのサービスの一部 (続き)

サービス名	サービスで提供される操作の一部
JobService	ジョブの進行状況と結果の取得、ジョブのキャンセル、ジョブスケジュールの更新。
NasConnectionService	NAサーバーのホスト名の取得、NAサーバーに対するコマンドの実行。
NetworkDeviceService	指定した検索フィルターに基づくファミリー、名前、モデル、タイプなどの情報の取得。
SequenceService	サーバーにオペレーティングシステムをインストールするためのOSシーケンスの作成、取得、実行。
ServerService	サーバーに関する情報の取得、サーバー上でのポリシーの調整 (修復) (ソフトウェアのインストール)、カスタムフィールドおよび属性の取得と設定、OSシーケンスの実行 (OSのインストール)。
SoftwarePolicyService	ソフトウェアポリシーの作成、サーバーへのポリシーの割り当て、ポリシーの内容の取得、サーバーに対するポリシーの修復 (調整)。
SolPatchService	Solarisパッチのインストールとアンインストール、ポリシーオーバーライドの追加。
VirtualColumnService	カスタムフィールドおよびカスタム属性の管理。
WindowsPatchService	Windowsパッチのインストールとアンインストール、ポリシーオーバーライドの追加。

APIのオブジェクト

SAプラットフォームAPIは関数に基づいていますが、クライアントがオブジェクト指向のライブラリを作成できるように設計されています。SAのデータモデルには、サーバー、フォルダー、カスタマーなどのオブジェクトが含まれます。これらは永続的なオブジェクトです。すなわち、モデルリポジトリに格納されます。APIには、これらのオブジェクトに対応する次のアイテムがあります。

- オブジェクトの動作を定義するサービス。たとえば、`ServerService`のメソッドは、管理対象サーバーオブジェクトの動作を指定します。
- 永続的オブジェクトのインスタンスを表すオブジェクト (ID) 参照。たとえば、`ServerRef`は管理対象サーバーを一意に識別する参照です。`ServerService`のほとんどのメソッドの第1引数は、そのメソッドの操作対象の管理対象サーバーを識別する`ServerRef`です。`ServerRef`の`Id`属性は、モデルリポジトリに格納されているサーバーオブジェクトの主キーです。
- 永続的オブジェクトのデータメンバー (属性、フィールド) を表す1つ以上の値オブジェクト (VO)。たとえば、`ServerVO`には`agentVersion`や`loopbackIP`などの属性が含まれます。`ServerHardwareVO`の属

性には、manufacturer、model、assetTagが含まれます。ほとんどの属性は、クライアントアプリケーションからは変更できません。属性が変更可能な場合、setterメソッドのAPIドキュメントに「フィールドはクライアントから設定可能」と記されています。

パフォーマンス上の理由で、永続的オブジェクトの更新操作は大きい単位で行われます。たとえば、ServerServiceのupdateメソッドは、個々の属性でなくServerVO全体を引数に取ります。

例外

SA固有のAPI例外は、すべて次のいずれかの例外から派生します。

OpswareException - アプリケーションレベルのエラーが起きたときに発生します。たとえば、エンドユーザーが無効な値をメソッドに渡したときなどです。クライアントアプリケーションは通常、このタイプの例外からは回復できます。OpswareExceptionから派生する例外の例としては、NotFoundException、NotInFolderException、JobNotScheduledExceptionがあります。

OpswareSystemException - SA内部でエラーが起きたときに発生します。通常、クライアントアプリケーションが実行可能になるには、SA管理者が問題を解決する必要があります。

次の例外はセキュリティに関連しています。

AuthenticationException - 無効なSAユーザー名またはパスワードが指定されたときに発生します。

AuthorizationException - ユーザーに操作の実行またはオブジェクトへのアクセスの権限がないときに発生します。アクセス権の詳細については、『SA 10.50管理ガイド』を参照してください。

イベントキャッシュ

一部のクライアントアプリケーションは、SAオブジェクトのローカルコピーを保持する必要があります。クライアントからEventCacheServiceを通じてアクセスされるキャッシュには、SAオブジェクトへの最近の変更を記述するイベントが記録されています。クライアントは、定期的にキャッシュをポーリングすることで、オブジェクトの作成、更新、削除を検出できます。イベントは、設定された一定の時間だけキャッシュに保持されます。デフォルトでは、最近2時間分のイベントが保持されます。保持時間を変更するには、HPE SSOポータルの『Server Automation管理ガイド』の説明に従って、Webサービスデータアクセスエンジンの構成ファイルを編集します。

検索

SAプラットフォームAPIの検索メカニズムは、値オブジェクトの属性 (フィールド) に基づいてオブジェクト参照を取得します。たとえば、getServerRefsメソッドはServerVO値オブジェクトの属性によって検索を行います。getServerRefsメソッドのシグネチャは次のとおりです。

```
public ServerRef[] getServerRefs(Filter filter)...
```

get*Refsメソッドは、filterパラメーターで検索基準を指定するオブジェクトを受け取ります。単純な式によるfilterパラメーターの構文を次に示します。

値オブジェクト.属性 演算子 値

(この構文は単純化されています。完全な定義については、「フィルターの文法」(219ページ)を参照してください)。

次の例は、getServerRefsメソッドのfilterパラメーターを示します。

```
ServerVO.hostName = "d04.example.com"
ServerVO.model BEGINS_WITH "POWER"
ServerVO.use IN "UNKNOWN" "PRODUCTION"
```

次のような複合式も使用できます。

```
(ServerVO.model BEGINS_WITH "POWER") AND (ServerVO.use = "UNKNOWN")
```

値オブジェクトの属性の中には、filterパラメーターに指定できないものがあります。たとえば、ServerVO.stateはfilterパラメーターで使用できますが、ServerVO.OsFlavorは使用できません。使用可能な属性を知るには、APIドキュメントの値オブジェクトの項目に「フィールドはフィルタークエリに使用可能」というコメントがあるかどうかを確認します。

セキュリティ

SAプラットフォームのユーザーは、SA自動化プラットフォームAPIのメソッドを呼び出すために認証され、承認される必要があります。SAに接続する際に、クライアントはSAユーザー名とパスワードを送信します (認証)。メソッドを呼び出すためには、SAユーザーは必要なアクセス権を持つグループに属する必要があります (承認)。アクセス権は、ユーザーが実行できる操作のタイプを制限するだけでなく、操作で使用されるサーバーおよびネットワークデバイスへのアクセスも制限します。

アプリケーションクライアントがプラットフォーム上で動作するためには、SA管理者が必要なユーザーおよびアクセス権をコマンドセンターで指定する必要があります。詳細については、『SA 10.50管理ガイド』の

「ユーザーグループの設定」の項を参照してください。セキュリティ関連の例外の詳細については、「例外」(35ページ)を参照してください。

クライアントとSAの間の通信は暗号化されます。Webサービスクライアントの場合、要求と応答のSOAPメッセージ (操作呼び出しを実装するもの) はSSL over HTTP (HTTPS) で暗号化されます。

APIドキュメントとTwister

SAには、SAプラットフォームAPIに関して記述しているAPIドキュメント (Javadocs) が付属しています。APIドキュメントにアクセスするには、ブラウザーに次のURLを指定します。

```
https://<SAコアホスト>/twister
```

<SAコアホスト>は、コマンドセンターコンポーネントを実行しているSAコアサーバーのIPアドレスまたはホスト名です。

Twisterは、ブラウザー内部からAPIメソッドを一度に1つずつ呼び出すためのプログラムです。たとえば、`ServerService.getServerVO`メソッドを呼び出すには、次の手順を実行します。

1. ブラウザーでAPIドキュメントを開きます。
2. [すべてのクラス] ペインで`com.opsware.server`を選択します。
3. `com.opsware.server`ペインで`ServerService`を選択します。
4. メインペインで、下にスクロールして`getServerVO`メソッドを表示します。
5. `getServerVO`メソッドに対して [**試行**] をクリックします。
6. SAユーザー名とパスワードを入力します。
7. `ServerService.getServerVO`のTwisterペインで、[oid] フィールドに管理対象サーバーのIDを入力します。
8. [**実行**] をクリックします。Twisterペインに、返された`ServerVO`オブジェクトの属性が表示されます。

定数のフィールド値

APIのいくつかの値オブジェクト (VO) では、フィールドの値が定数で定義されています。たとえば、`JobInfoVO`の`status`フィールドは、`STATUS_ACTIVE`や`STATUS_PENDING`などの定数で定義される値を取ります。APIでは定数がJavaの`static final`フィールドとして指定されていますが、APIから生成されたWSDLでは定数が定義されていません。定数の定義を見るには、APIドキュメントで [定数のフィールド値] ページを開きます。

```
https://<SAコアホスト>/twister/docs/constant-values.html
```

たとえば、[定数のフィールド値] ページではSTATUS_ACTIVEが整数1と定義されています。

サポートされるクライアント

SAプラットフォームは、シェルスクリプトを作成するシステム管理者から、最新のツールやテクノロジーに精通した.NETやJavaのプログラマーまで、さまざまなスキルレベルのプログラマーをサポートします。サポートされるすべてのクライアントは同じメソッドのセットを呼び出します。メソッドはSAプラットフォームのサービスによって分類されています。開発者は、SAプラットフォームAPIのメソッドを呼び出す次のタイプのクライアントを作成できます。

- **SAコマンドラインインタフェース (CLI)**: Global Shellセッションから起動されるシェルスクリプトは、OGFSの実行可能プログラムであるCLIメソッドを呼び出すことで、SAプラットフォームAPIにアクセスできます。各CLIメソッドは、APIの1つのメソッドに対応します。
- **Webサービス**: これらのクライアントは、SOAP over HTTPSを使用して、SAに要求を送信し、応答を受信します。Webサービスの操作 (WSDLで定義) は、APIのメソッドに対応します。開発者は、PerlやC#といった一般的な言語でWebサービスクライアントを作成できます。
- **Java RMI**: これらのクライアントは、他のJava仮想マシンからリモートJavaオブジェクトを呼び出します。
- **Pytwist**: これらのPythonプログラムは、SAコアサーバーまたは管理対象サーバー上で動作します。

WebサービスクライアントとJava RMIクライアントは、SAコアサーバーまたは管理対象サーバーと異なるサーバー上で動作することができます。CLIメソッドは、OGFSがインストールされているコアサーバー上のGlobal Shellセッションで実行されます。

SA CLIメソッド

エンドユーザーは、SAクライアントを通じてSAにアクセスします。場合によっては、上級ユーザーが、複数のサーバーに対するバルク操作や繰り返し作業を実行するために、コマンドライン環境でSAにアクセスすることが必要になります。SAのコマンドライン環境は、Global Shell (OGSH)、Global File System (OGFS)、SAコマンドラインインタフェース (CLI) メソッドから構成されます。

SAの操作をコマンドラインから実行するには、OGSHセッション内部からSA CLIメソッドを呼び出します。SA CLIメソッドは、SA APIのメソッドに対応する、OGFS内の実行可能ファイルです。SA CLIメソッドを実行すると、対応するAPIメソッドが呼び出されます。

この項を理解するには、OGSHとOGFSに関する知識が必要です。詳細については、HPE SSOポータルで『Server Automationユーザーガイド』の「OGSH」を参照してください。

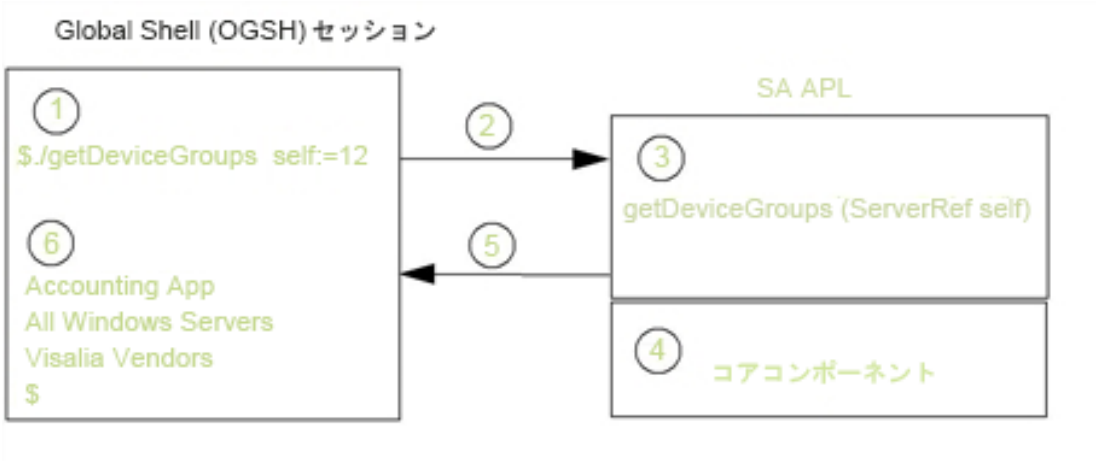
`ouploadおよびodownload`コマンドについては、HPE SSOポータルで『Server Automationユーザーガイド』の「OCLI 1.0」を参照してください。

メソッドの呼び出し

次の図に示すように、OGSHセッションでSA CLIメソッドを呼び出すと、次の処理が行われます。

1. 入力されたコマンドとパラメーターをOGSHが解析して、APIメソッドを判定します。
2. OGSHが対応するAPIメソッドを呼び出します。
3. この操作を実行するアクセス権がユーザーにあるかどうかが検証されます。その後、SAが操作を実行します。
4. APIメソッドが結果をSA CLIメソッドに返します。
5. SA CLIメソッドが戻り値をOGSHセッションの`stdout`に書き出します。例外が発生した場合、SA CLIメソッドは0以外のステータスを返します。

SA CLIメソッドの呼び出しの概要



セキュリティ

SA CLIメソッドは、SAクライアントと同じ認証と承認のメカニズムを使用します。OGSHセッションを開始すると、SAはSAユーザーを認証します。SA CLIメソッドを実行すると、承認が実行されます。SA CLIメソッドを正常に実行するには、必要なアクセス権を持つグループにSAユーザーが属する必要があります。セキュリティの詳細については、HPE SSOポータルで『Server Automation管理ガイド』を参照してください。

APIとSA CLIメソッドの間のマッピング

OGFSは、SAオブジェクトをディレクトリ構造で、オブジェクト属性をテキストファイルで、APIメソッドを実行可能ファイルで表現しています。これらの実行可能ファイルがSA CLIメソッドです。各SA CLIメソッドは、1つのAPIメソッドに対応します。メソッド名、パラメーター、戻り値は、どちらのタイプのメソッドでも同じです。

たとえば、`setCustomer` APIメソッドは次のJavaシグネチャを持ちます。

```
public void setCustomer(ServerRef self,
          CustomerRef customer)...
```

OGFSの対応するSA CLIメソッドは次の構文を取ります。

```
setCustomer self:i=サーバーID customer:i=カスタマーID
```

パラメーター名 (`self`と`customer`) がどちらの言語でも同じであることに注意してください(`:i`記法はフォーマット指定子と呼ばれます。これについてはこの項の後の方で説明します)。この例では、戻り値型はvoidなので、SA CLIメソッドは結果を`stdout`に書き出しません。SA CLIメソッドがオブジェクトを表す文字列を返す方法の詳細については、「戻り値」(60ページ)を参照してください。

SA CLIメソッドとUnixコマンドの違い

OGSHではUnixコマンドとSA CLIメソッドの両方を実行できますが、SA CLIメソッドはいくつかの点で異なっています。

- 多くのUnixコマンドと異なり、SA CLIメソッドは`stdin`からデータを読み取りません。したがって、パイプ(|)で接続されたコマンドのグループ内にSA CLIメソッドを挿入することはできません(ただし、SA CLIメソッドは`stdout`への出力は行います)。
- ほとんどのUnixコマンドはパラメーターをフラグと値で受け取ります(例、`ls -l /usr`)。SA CLIメソッドでは、コマンドラインパラメーターは名前と値のペアを等号で結んだものです。
- Unixコマンドはテキストベースです。これらのコマンドは、データを文字列で受け取って返します。これに対して、SA CLIメソッドは複合オブジェクトを受け取って返すことができます。
- SA CLIメソッドでは、パラメーターと戻り値の形式を指定できます。Unixコマンドにはこれに相当する機能はありません。

SA CLIメソッドのチュートリアル

ここでは、実際の環境で実行できる例を使用して、SA CLIメソッドの概要を説明します。このチュートリアルを終了すると、SA CLIメソッドを実行したり、SAオブジェクトの`self`ファイルを調べたり、複数のサーバーに対してSA CLIメソッドを呼び出すスクリプトを作成したりできるようになります。

チュートリアルを開始する前に、次のことが必要です。

- SAクライアントにログオンできること。
- SAユーザーが少なくとも1つの管理対象サーバーで読み取り/書き込みアクセス権を持つこと。アクセス権は通常はセキュリティ管理者によって割り当てられます。詳細については、HPE SSOポータルの『Server Automation管理ガイド』を参照してください。
- SAユーザーが同じ管理対象サーバーですべてのOGSHアクセス権を持つこと。これらのアクセス権の詳細については、HPE SSOポータルで『Server Automationユーザーガイド』の「aaaユーティリティ」を参照してください。
- OGSHとOGFSについての知識があること。これらの機能を初めて使用する場合は、このチュートリアルを実行する前に、HPE SSOポータルで『Server Automationユーザーガイド』の「Global Shell」を参照してください。

このチュートリアルのコマンド例は、abc.example.comという名前のWindowsサーバーを対象としています。このサーバーは、「All Windows Servers」という名前のサーバーグループに属しています。これらのコマンドを試すときには、abc.example.comを、自分がアクセス権を持つ管理対象サーバーのホスト名に置き換えてください。

1. OGSHセッションを開きます。

   Global ShellセッションはSAクライアントから開くことができます。[**アクション**] メニューで、[**Global Shell**] を選択します。OGSHセッションは、デスクトップ上で動作しているターミナルクライアントから開くこともできます。詳細については、HPE SSOポータルで『Server Automationユーザーガイド』の「Global Shellセッションを開く」を参照してください。

2. サーバーのSA CLIメソッドのリストを表示します。

   サーバーのmethodサブディレクトリに、そのサーバーで実行可能なメソッドに対応する実行可能ファイルが含まれます。次の例は、abc.example.comサーバーのSA CLIメソッドのリストを表示します。

   ```
   $ cd /opsw/Server/@/abc.example.com/method
   $ ls -1
   addDeviceGroups
   attachPolicies
   attachVirtualColumn
   checkDuplex
   clearCustAttrs
   ...
   ```

   これらのメソッドにはインスタンスコンテキストがあり、特定のサーバーインスタンス (この例では abc.example.com) で動作します。サーバーインスタンスは、メソッドのパスから知ることができます。静的コンテキストを持つメソッドについては手順5で説明します。

3. SA CLIメソッドをパラメーターなしで実行します。

   abc.example.comが属するパブリックサーバーグループを表示するには、getDeviceGroupsメソッドを呼び出します。

   ```
   $ cd /opsw/Server/@/abc.example.com/method
   $ ./getDeviceGroups
   Accounting App
   All Windows Servers
   Visalia Vendors
   ```

4. メソッドをパラメーター付きで呼び出します。

メソッドのコマンドラインパラメーターは、スペース文字で区切られた名前と値のペアで示されます。次のsetCustomerの呼び出しでは、パラメーター名はcustomerで、値は20039です。パラメーター名の末尾の:iは、後で説明するID形式指定子です。

次のメソッド呼び出しは、abc.example.comサーバーのカスタマーをOpswareからC39に変更します。カスタマーC39のIDは20039です。

```
$ cd /opsw/Server/@/abc.example.com
$ cat attr/customer ; echo
Opsware
$ method/setCustomer customer:i=20039
$ cat attr/customer ; echo
C39
```

5. 管理対象サーバーの静的コンテキストメソッドのリストを表示します。

   静的コンテキストメソッドは、/opsw/apiディレクトリにあります。これらのメソッドは、オブジェクトの特定のインスタンスに限定されません。

   サーバーの静的メソッドのリストを表示するには、次のコマンドを入力します。

   ```
   $ cd /opsw/api/com/opsware/server/ServerService/method

   $ ls
   ```

   リストに表示されるメソッドは、手順2で表示されるものと同じです。

6. メソッドをselfパラメーター付きで呼び出します。

   このステップでは、getDeviceGroupsを静的コンテキストメソッドとして呼び出します。手順3に示したインスタンスコンテキストメソッドと異なり、静的コンテキストメソッドにはサーバーインスタンスを識別するためのselfパラメーターが必要です。
   たとえば、abc.example.comサーバーのIDが530039だとします。このサーバーのグループのリストを表示するには、次のコマンドを入力します。

   ```
   $ cd /opsw/api/com/opsware/server/ServerService/method
   $ ./getDeviceGroups self:i=530039
   Accounting App
   All Windows Servers
   Visalia Vendors
   ```

このgetDeviceGroupsの呼び出しを、インスタンスコンテキストを示す手順3の呼び出しと比較してみてください。どちらの呼び出しも、APIの対応する同じメソッドを呼び出し、同じ結果を返します。

7. サーバーのselfファイルを調べます。

   SAでは、各管理対象サーバーは1つのオブジェクトです。ただし、OGFSはファイルシステムであり、オブジェクトモデルではありません。selfファイルは、SAオブジェクトのさまざまな表現にアクセスするために使用できます。表現には、ID、名前、構造があります。

   サーバーのデフォルトの表現は名前です。たとえば、サーバーの名前を表示するには、次のコマンドを入力します。

   ```
   $ cd /opsw/Server/@/abc.example.com
   $ cat self ; echo
   abc.example.com
   ```

   サーバーのIDがわかっている場合、次の例のように、selfファイルから名前を取得できます。

   ```
   $ cat /opsw/.Server.ID/530039/self ; echo
   abc.example.com
   ```

8. selfファイルのID形式指定子を指定します。

   selfファイルの特定の表現を選択するには、ピリオドの後にファイル名、その後に形式指定子を入力します。たとえば、次のcatコマンドには、サーバーIDを表示する形式指定子 (:i) が含まれています。

   ```
   $ cd /opsw/Server/@/abc.example.com
   $ cat .self:i ; echo
   com.opsware.server.ServerRef:530039
   ```

   この出力は、abc.example.comのIDが530039であることを示します。
   com.opsware.server.ServerRefは、SA APIの対応するオブジェクトであるサーバー参照のクラス名です。

   > **注:** 先頭のピリオドは、ファイルとメソッドの戻り値に形式指定子を付ける場合には必要ですが、メソッドのパラメーターには使用しません。

9. 構造形式指定子を示します。

   構造形式指定子 (:s) は、複合オブジェクトの属性を示します。属性は名前と値のペアとして表示

され、すべてのペアは中括弧に囲まれます。構造形式は、コマンドラインで複合オブジェクトをメソッドパラメーターに指定する場合に使用されます(メソッド呼び出しの例については、「パラメーターとしての複合オブジェクトと配列」(59ページ)を参照してください)。

次の例は、abc.example.comを構造形式で表示します。

```
$ cd /opsw/Server/@/abc.example.com
$ cat .self:s ; echo
{
managementIP="192.168.8.217"
modifiedBy="spujare"
manufacturer="DELL COMPUTER CORPORATION"
use="UNKNOWN"
discoveredDate=1149012848000
origin="ASSIMILATED"
osSPVersion="SP4"
locale="English_United States.1252"
reporting=false
netBIOSName=
previousSWReg=1150673874000
osFlavor="Windows 2000 Advanced Server"
. . .
```

サーバーの属性も、attrディレクトリのファイルによって次のように表されます。

```
$ pwd
/opsw/Server/@/abc.example.com
$ cat attr/osFlavor ; echo
Windows 2000 Advanced Server
```

10. SA CLIメソッドを呼び出すスクリプトを作成します。

このステップに示すスクリプトの例は、All Windows Serversという名前のパブリックサーバーグループのすべてのサーバーを反復処理します。各サーバーに対して、スクリプトはgetCommCheckTime SA CLIメソッドを実行します。

初めに、OGFSのホームディレクトリに戻ります。

```
$ cd
$ cd public/bin
```

次に、viエディターを実行します。

```
$ vi
```

viで、次の各行を挿入してbashスクリプトを作成します。

```
#!/bin/bash
# iterate_time.sh

METHOD_DIR="/opsw/api/com/opsware/server/ServerService/method"
GROUP_NAME="All Windows Servers"
cd "/opsw/Group/Public/$GROUP_NAME/@/Server"

for SERVER_NAME in *
do
 SERVER_ID=`cat $SERVER_NAME/.self:i`
 echo $SERVER_NAME
 $METHOD_DIR/getCommCheckTime self:i=$SERVER_ID
 echo
 echo
done
```

viでファイルをiterate_time.shという名前で保存します。viを終了します。

iterate_time.shのアクセス権をchmodで変更し、実行します。

```
$ chmod 755 iterate_time.sh
$ ./iterate_time.sh
abc.example.com
2006/06/20 16:46:56.000
. . .
```

形式指定子

形式指定子は、SA CLI環境の値の表示や解釈の方法を指定します。形式指定子は、メソッドのパラメーター、メソッドの戻り値型、selfファイル、オブジェクト属性に適用できます。フォーマット指定子を指定するには、コロンと次の表に示す文字の1つを追加します。

ファイルまたはメソッドの戻り値にフォーマット指定子を指定する場合、ファイルまたはメソッド名の前にピリオドを付ける必要があります。フォーマット指定子を持つメソッドパラメーターには、先頭のピリオドはありません。

形式指定子の一覧

形式指定子	説明	有効なオブジェクトタイプ	メソッドパラメーターとしての使用
:n	**名前**: オブジェクトを識別する文字列。一意の名前であることが推奨されますが、必須ではありません。名前がないオブジェクトの場合、この表現はID表現と同じです。	SAオブジェクト	可能。名前があいまいな場合、エラーが生じます。
:i	**ID**: オブジェクトタイプとそのSA IDを一意に識別する形式。オブジェクト参照とも呼ばれます。	SAオブジェクト、日付 (`java.util.Calendar`) オブジェクト	可能。タイプがコンテキストから明らかな場合、タイプは省略できます。
:s	**構造**: コマンドラインで複合値を指定するためのコンパクトな表現。属性は中括弧で囲まれます。	任意の複合オブジェクト	可能
:d	**ディレクトリ**: 属性をOGFSのディレクトリで表します。	属性である任意の複合オブジェクト。この表現は、メソッドパラメーターまたは戻り値には使用できません。	不可

形式指定子の位置

形式指定子は、対象の項目のすぐ後に指定します。ファイルの場合、形式指定子はファイル名の後に置きます。次の例では、先頭のピリオドに注意してください。

```
cat .self:s
```

メソッドの戻り値型に適用する場合、形式指定子はメソッド名の後に置きます。次の呼び出しは、返されるグループのIDを表示します。

```
./.getDeviceGroups:i
```

メソッドパラメーターの場合、形式指定子は、次の例のように、パラメーター名の後、等号の前に置きます。

```
./setCustomer self:i=9977 customer:i=239
```

形式指定子を持つメソッドパラメーターには先頭のピリオドはありません。

デフォルトの形式指定子

すべての値またはオブジェクトには、デフォルトの形式指定子があります。たとえば、osVersion属性のデフォルトの形式指定子は名前です。次の2つのcatコマンドは、同じ出力を生成します。

```
cd /opsw/Server/@/d04.example.com/attr
cat osVersion
cat .osVersion:n
```

名前形式指定子は、サーバーやカスタマーなど、モデルリポジトリに格納されているSAオブジェクトに対するデフォルトです。その他の複合オブジェクトに対しては、構造形式指定子がデフォルトです。

ID形式指定子の例

次の例は、d04.example.comサーバーが属するファシリティのIDを表示します。

```
cd /opsw/Server/@/d04.example.com/attr
cat .facility:i ; echo
```

(前のechoコマンドはオプションです。これは、出力を読みやすくするための改行文字を生成します。セミコロンは、同じ行に入力されたbashステートメントを区切るためのものです)。

ID形式指定子による値の出力は、前にJavaクラス名が付きます。たとえば、ファシリティ値のIDが39の場合、前のcatコマンドは次の出力を表示します。

```
com.opsware.locality.FacilityRef:39
```

次のgetDeviceGroupsメソッドの呼び出しは、d04.example.comが属するパブリックサーバーグループのIDをリストします。

```
cd /opsw/Server/@/d04.example.com/method
./.getDeviceGroups:i
```

ID形式のその他の例については、「selfファイル」(55ページ)を参照してください。

構造形式指定子の構文

構造形式は、さまざまな属性を持つ複合オブジェクトを表現します。この形式は、複合オブジェクトをメソッドパラメーターに指定するために使用できます。例については、「パラメーターとしての複合オブジェクトと配列」(59ページ)を参照してください。

構造形式は中括弧で囲んだ名前と値のペアの連続です。ペアの間は空白文字で区切られます。名前と値のペアは属性を表します。構造形式の構文は次のとおりです。

```
{ name-1=value-1 name-2=value-2 .. . }
```

単純な例を次に示します。

```
{ version=10.1.3 isCurrent=true }
```

区切り文字としては任意の空白文字が使用できます。

```
{
       version=10.1.3
       isCurrent=true
}
```

属性は構造で指定することもできます。これにより、ネストしたオブジェクトを表現できます。次の例では、versionDesc属性が構造で表現されています。

```
{
program=agent
versionDesc={
               version=10.1.3
               isCurrent=true
               comment="Latest version"
               }
}
```

構造内部で配列を指定するには、属性名を繰り返します。次の構造には、stepsという名前の配列があります。この配列は3要素で、値は33、14、28です。

```
{ moduleName="Some Initiator" steps=33 steps=14 steps=28 }
```

構造形式指定子の例

次の例は、facility属性の構造形式を指定します。

```
cd /opsw/Server/@/d04.example.com/attr
cat .facility:s
```

このcatコマンドは、次の出力を生成します。customersは配列で、このファシリティに関連付けられているすべてのカスタマーに対応する要素を持ちます。

```
{
       modifiedBy="192.168.9.246"
       customers="Customer Independent"
       customers="Not Assigned"
       customers="Opsware Inc."
       customers="Acme Inc."
       . . .
       ontogeny="PROD"
       createdBy=
       status="ACTIVE"
       createdDt=-1
       realms="Transitional"
       realms="C39"
       realms="C39-agents"
       modifiedDt=1146528752000
       name="C39"
       displayName="C39"
}
```

次のgetDeviceGroupsの呼び出しは、戻り値の構造形式指定子を示します。

```
cd /opsw/Server/@/d04.example.com/method
./.getDeviceGroups:s
```

このgetDeviceGroupsの呼び出しは、次の出力を表示します。d04.example.comは2つのサーバーグループに属するので、出力には2つの構造が含まれます。各構造のdevices配列には、そのグループに属するサーバーが要素として含まれます。

```
{
       dynamic=true
       devices="m302-w2k-vm1.dev.example.com"
       devices="d04.example.com"
       . . .
       status="ACTIVE"
       34 Chapter 2
       public=true
       fullName="Device Groups Public All Windows Servers"
       description="test"
       createdDt=-1
       modifiedDt=1142019861000
       parent="Public"
}
{
       dynamic=true
```

```
        devices="opsware-nibwp.build.example.com"
        devices="glengarriff.snv1.dev.example.com"
        devices="millstreet"
        . . .
        fullName="Device Groups Public z_testsrvgroup"
        . . .
}
```

構造形式指定子は、値オブジェクト (VO) を取得するメソッドに対するデフォルトです。たとえば、次の2つのgetServerVOの呼び出しは等価です。

```
cd /opsw/Server/@/d04.example.com/method
./.getServerVO:s./getServerVO
```

この例で、getServerVOは次の出力を表示します。

```
{
        managementIP="192.168.198.93"
        modifiedBy=
        manufacturer="DELL COMPUTER CORPORATION"
        use="UNKNOWN"
        discoveredDate=1145308867000
        origin="ASSIMILATED"
        osSPVersion="RTM"
        locale="English_United States.1252"
        reporting=false
        netBIOSName=
        previousSWReg=1147678609000
        osFlavor="Windows Server 2003, Standard Edition"
        peerIP="192.168.198.93"
        modifiedDt=1145308868000
        . . .
        serialNumber="HVKZS51"
}
```

この構造は、SA APIのServerVOクラスを表します。この構造のすべての属性は、それぞれattrディレクトリの1つのファイルに対応します。次の例では、getServerVOコマンドとcatコマンドはどちらもサーバーのserialNumber属性の値を表示します。

```
cd /opsw/Server/@/d04.example.com
./method/getServerVO | grep serialNumber
cat attr/serialNumber ; echo
```

ディレクトリ形式指定子の例

次のコマンドは、現在の作業ディレクトリを、サーバーd04.example.comに関連付けられているカスタマーに変更します。

```
cd /opsw/Server/@/d04.example.com/attr/.customer:d
```

次のコマンドは、このカスタマーの名前をリストします。

```
cat /opsw/Server/@/d04.example.com/attr/\
.customer:d/attr/name
```

ディレクトリ指定子は、ディレクトリ名を必要とするコマンド引数のみに使用できます。次のcatコマンドは、ディレクトリを表示しようとするため失敗します。

```
cat /opsw/Server/@/d04.example.com/attr/.customer:d # WRONG!
```

一方、次のコマンドは有効です。

```
ls /opsw/Server/@/d04.example.com/attr/.customer:d
```

値の表現

SA CLIメソッドは、シェル環境 (OGSH) で実行されるため、データを文字列で受け取って返します。一方、対応するAPIメソッドは、数値、ブール値、オブジェクトなどの他のデータ型を受け取って返します。ここでは、OGFSとSA CLIメソッドが文字列以外のデータ型を表現する方法を説明します。

OGFS内のSAオブジェクト

SAのデータモデルには、サーバー、サーバーグループ、カスタマー、ファシリティなどのオブジェクトが含まれます。OGFSでは、これらのオブジェクトはディレクトリ構造で表現されます。

```
/opsw/Customer
/opsw/Facility
/opsw/Group
/opsw/Library
/opsw/Realm
/opsw/Server
. . .
```

上のリストは一部だけを示しています。完全なリストを見るには、`ls /opsw`と入力してください。

オブジェクト属性

SAオブジェクトの属性は、`attr`サブディレクトリのテキストファイルによって表されます。各ファイルの名前は、属性の名前と一致します。ファイルの内容は、属性の値を示します。

たとえば、`/opsw/Server/@/buzz.example.com/attr`ディレクトリには次のファイルが含まれます。

```
agentVersion
codeset
createdBy
createdDt
customer
defaultGw
36 Chapter 2
description
discoveredDate
facility
hostName
locale
lockInfo
loopbackIP
managementIP
manufacturer
.. .
```

`buzz.example.com`サーバーの管理IPアドレスを表示するには、次のコマンドを入力します。

```
cd /opsw/Server/@/buzz.example.com/attr
cat managementIP ; echo
```

カスタム属性

カスタム属性は、サーバーなどのSAオブジェクトに割り当てることができる名前と値のペアです。OGFSでは、カスタム属性は`CustAttr`サブディレクトリのテキストファイルで表されます。カスタム属性を作成するには、OGSHセッションで`CustAttr`の下に新しいテキストファイルを作成します。次の例は、名前が`MyGreeting`で値が`hello there`のカスタム属性を`buzz.example.com`サーバーに作成します。

```
cd /opsw/Server/@/buzz.example.com/CustAttr
echo -n "hello there" > MyGreeting
```

その他の例については、『SA 10.50ユーザーガイド』の「カスタム属性の管理」を参照してください。

selfファイル

selfファイルは、サーバーやカスタマーなどのSAオブジェクトのディレクトリにあります。このファイルは、形式指定子に応じて、現在のオブジェクトのさまざまな表現へのアクセスを可能にします詳細については、「形式指定子」(47ページ)を参照してください。

buzz.example.comサーバーのIDをリストするには、次のコマンドを入力します。

```
cd /opsw/Server/@/buzz.example.com
cat .self:i ; echo
```

サーバーの場合、デフォルトの形式指定子は名前です。次のコマンドは同じ出力を表示します。

```
cat self ; echo
cat .self:n ; echo
```

次のコマンドは、サーバーの属性を構造形式でリストします。

```
cat .self:s
```

プリミティブ値

次の表は、APIとSA CLIメソッドでのその文字列表現の間でのプリミティブ値の変換方法を示します。日付を除いて、プリミティブ値は形式指定子をサポートしません。日付はID形式指定子をサポートします。

プリミティブ型とSA CLIメソッドの間の変換

プリミティブ型	Javaの相当する型	SA CLIメソッドの出力	SA CLIメソッドの入力
String	java.lang.String	現在のセッションのエンコードで表現された文字列。	現在のセッションのエンコードからUnicodeに変換された文字列。
数値	byte、short、int、long、float、double、およびそれらに相当するオブジェクト	10進形式。ローカライズなし。非常に大きいか小さい値の場合は科学的記数法。	例 - 10進: 101、512.34、-104 16進: 0x1F32, 0x2e40 8進: 0543 科学的記数法: 4.3E4、6.532e-9、1.945e+02
ブール値	boolean、Boolean	trueまたはfalse	文字列true (大文字と小文字は区別されません)

プリミティブ型とSA CLIメソッドの間の変換 (続き)

プリミティブ型	Javaの相当する型	SA CLIメソッドの出力	SA CLIメソッドの入力
			は、trueと評価されます。それ以外の値は、falseと評価されます。
バイナリデータ	byte[]、Byte[]	バイナリ文字列。セッションエンコードからの変換はありません。	バイナリ文字列。セッションエンコードへの変換はありません。
日付	java.util.Calendar	日付値。デフォルトでは次の形式で表現されます。YYYY/MM/DD HH:MM:SS.mmm 時刻はUTCで表現されます。ID形式指定子を指定した場合、値はエポックからのミリ秒数 (UTC) で表されます。	出力と同じ。

配列

配列オブジェクトの表現は、配列が単独で (配列属性ファイルまたはメソッドの戻り値で) 使用されるか、複合オブジェクトの構造内に含まれるかによって異なります。

単独の配列オブジェクトは、元になる型に基づいて、改行文字を区切りとして表現されます。配列要素内の改行文字は\n、バックスラッシュは\\のようにエスケープされます。

配列値は、元になる型でサポートされる任意の表現で出力または入力できます。たとえば、デフォルトでは、getDeviceGroupsメソッドはグループを名前でリストします。

```
All Windows Servers
Servers in Austin
Testing Pool
```

ID形式指定子を指定した場合 (.getDeviceGroups:i)、メソッドはグループのIDを表示します。

```
com.opsware.device.DeviceGroupRef:15960039
com.opsware.device.DeviceGroupRef:10390039
com.opsware.device.DeviceGroupRef:17380039
```

複合オブジェクトの構造内に含まれる配列は、属性を名前に使用して、名前と値のペアの集合で表現されます。属性は複数回、すなわち配列の各要素に対して1回ずつ現れます。属性が現れる順序によって、配列内の要素の順序が決まります。次の例に示す構造には、2つの要素が含まれています。1つはsubjectという文字列、もう1つはranksという3要素の数値配列です。

```
{ subject=my favorites ranks=17 ranks=44 ranks=24 }
```

配列はディレクトリで表現することもできます。配列ディレクトリ内の各配列要素には、対応するファイル(プリミティブ型の場合)またはサブディレクトリ(複合型の場合)があります。各エントリの名前は、配列要素のインデックス番号(先頭が0)です。

複合オブジェクトの属性である配列を変更するには、その属性ファイルを編集します。この操作を行うと、編集したファイルの内容によって配列全体が置き換えられます。

要素が複合オブジェクトである配列を変更するには、そのディレクトリ表現を変更します。要素の値を変更するには、要素ファイルを編集します。たとえば、5個の文字列要素からなる配列があるとします。`ls`コマンドを実行すると、次のように要素が表示されます。

```
0 1 2 3 4
```

次のコマンドは、3番目の要素の値を変更します。

```
echo -n "My new value" > 2
```

SA CLIメソッドのパラメーターと戻り値

ここでは、メソッドのコンテキスト(インスタンスまたは静的)、パラメーターの使用法、戻り値、終了ステータスについて説明します。

メソッドのコンテキストとselfパラメーター

OGFSでは、メソッドは複数の場所に存在します。メソッドの場所はそのコンテキスト(インスタンスまたは静的)に関連しています。

インスタンスコンテキストのメソッドは、特定のSAオブジェクトの`method`ディレクトリに存在します。メソッドの呼び出しに`self`パラメーターは必要ありません。メソッドによって影響されるオブジェクトのインスタンスは、メソッドの場所によって示されます。次の例は、`d04.example.com`サーバーのカスタマーを変更します。

```
cd /opsw/Server/@/d04.example.com/method
./setCustomer customer:i=9
```

静的コンテキストのメソッドは、`/opsw/api`の下の1つの場所にあります。メソッドの呼び出しには、影響されるインスタンスを指定するためのselfパラメーターが必要です。次の静的コンテキストの例で、`self:i`は管理対象サーバーのIDを指定します。

```
cd /opsw/api/com/opsware/server/ServerService/method
./setCustomer self:i=230054 customer:i=9
```

コマンドラインでの引数渡し

コマンドライン引数は、名前と値のペアを等号 (=) で結んだもので指定されます。名前と値のペアとペアの間は、1つ以上の空白文字 (通常はスペース) で区切ります。コマンドラインに指定する名前は、SA APIの対応するJavaメソッドのパラメーター名と同じです。

たとえば、SA APIの`setCustomField`メソッドは次のように定義されています。

```
public void setCustomField(CustomFieldReference self,
        java.lang.String fieldName, java.lang.String strValue)...
```

次のSA CLIメソッドの例は、IDが3670039のサーバーのカスタムフィールドに値を割り当てます。

```
cd /opsw/api/com/opsware/server/ServerService/method
./setCustomField self:i=3670039 \
fieldName="Service Agreement" strValue="Gold"
```

執筆者メモ: 上記コマンドを実行して確認 (未定)

前に説明したように、インスタンスコンテキストのメソッドには`self`パラメーターは不要です。次の`setCustomField`の例は前の例と等価です。

```
cd /opsw/.Server.ID/3670039
./setCustomField \
fieldName="Service Agreement" strValue="Gold"
```

コマンドライン引数は任意の順序で指定できます。次の2つのSA CLIメソッド呼び出しは等価です。

```
./setCustomField fieldName="My Stuff" strValue="abc"
./setCustomField strValue="abc" fieldName="My Stuff"
```

パラメーターにNull値を指定するには、パラメーターを省略するか、等号の後ろに空白を入れます。次の例で、`myParam`の値はNullです。

```
./someMethod myField="more info" myParam= anotherParam=9834
./someMethod myField="more info"         anotherParam=9834
```

パラメーターの型の指定

メソッドのパラメーターに抽象型が指定されている場合、値とともに具象型を指定する必要があります。次の例では、`com.opsware.folder.FolderRef`型が必要です。

```
cd /opsw/api/com/opsware/folder/FolderService/method
./remove self:i="com.opsware.folder.FolderRef:730555"
```

具象型を指定しない場合、次のエラーメッセージが表示されます。

オブジェクト型型名は抽象型です。具象型を指定してください。

パラメーターとしての複合オブジェクトと配列

複合オブジェクトを引数として渡すには、「構造形式指定子の構文」(50ページ)に示すように、オブジェクトの属性を中括弧で囲みます。

次の例は、AllMineというパブリックサーバーグループを作成します。createメソッドにはpatternという1つのパラメーターがあり、属性parentとshortNameが中括弧に囲まれています。この例で、getPublicRootはトップパブリックグループのIDである2340555を返します。

```
cd /opsw/api/com/opsware/device/DeviceGroupService/method
./.getPublicRoot:i ; echo
./create “pattern={ parent:i=2340555 shortName=’AllMine’ }”
```

配列パラメーターを指定するには、各配列要素に対して1回ずつパラメーター名を繰り返し指定します。たとえば、次のassignメソッドの呼び出しは、policiesという名前の配列パラメーターの最初の2つの要素を指定しています。

```
cd /opsw/api/com/opsware/swmgmt
cd SoftwarePolicyService/method
./attachPolicies self:i=4220039 \
policies:i=4400335 policies:i=4400942
```

オーバーロードされたメソッド

Javaメソッド名がオーバーロードされるのは、同じクラスに同じ名前の複数のメソッドがあり、それらのパラメーターリストが異なっている場合です。SA CLIメソッドがオーバーロードされている場合、コマンドラインの引数名によって、どのメソッドが呼び出されるかが決まります。たとえば、setCustomFieldメソッドは、異なるデータ型の設定をサポートするためにオーバーロードされています。次の2つのコマンドは、それぞれメソッドの異なるバージョンを呼び出します。

```
./setCustomField \
fieldName="Service Agreement" strValue="Gold"
./setCustomField \
fieldName=hmp longValue=2245
```

戻り値

SA CLIメソッドに対応するAPIメソッドが値を返す場合、SA CLIメソッドは値をstdoutに出力します。Unixコマンドの場合と同様、メソッドのstdoutをファイルにリダイレクトしたり、環境変数に代入したりできます。

戻り値の表現を変更するには、メソッド名の前にピリオドを付け、後ろに形式指定子を追加します。次の例は、サーバー参照をデフォルトの名前でなくIDで返します。

```
cd /opsw/api/com/opsware/server/ServerService/method
./.findServerRefs:i
```

メソッドの戻り値型と互換性のない形式指定子を指定した場合、ファイルシステムはエラーを生成します。

終了ステータス

Unixシェルコマンドと同様、SA CLIメソッドは終了ステータス($?)で呼び出しの結果を示します。終了ステータスが0の場合は成功を示し、0以外の場合はエラーを示します。SA CLIメソッドはエラーメッセージをstderrに出力します。

SA CLIメソッドの終了ステータスコード

終了ステータス	カテゴリ	説明
0	成功	メソッドは正常に完了しました。
1	コマンドライン解析エラー	メソッド呼び出しのコマンドラインの形式が正しくなく、オプション (--option[=value]) とパラメーター値 (param=value) のセットに解析できません。
2	パラメーター解析エラー	パラメーター値をAPIが必要とするオブジェクトタイプに解析できません。
3	API使用法エラー	無効なパラメーター値などの使用法エラーによって呼び出しが失敗しました。
4	アクセスエラー	ユーザーにはこの操作を実行するアクセス権がありません。
5	その他のエラー	終了ステータス1～4で示されるもの以外のエラーが発生しました。

たとえば、次のbashスクリプトは、getDeviceGroupsメソッドの終了ステータスを確認します。

```
#!/bin/bash

cd /opsw/Server/@/toro.snv1.corp.example.com/method
./getDeviceGroups
cmnd_exit_status=$?
if [ $cmnd_exit_status -eq 0 ]
then
       echo "The command was successful."
else
       echo "The command failed."
       echo "Exit status = " $cmnd_exit_status
fi
```

各SA CLIメソッドは、対応する1つのAPIメソッドを呼び出します。APIメソッドで例外が発生した場合、SA CLIメソッドは0以外の終了ステータスを返します。メソッド呼び出しをデバッグする場合、発生した例外に関する情報を表示できると役に立つことがあります。OGFSの/sys/last-exceptionファイルには、最近のAPI呼び出しから発生した例外のスタックトレースが記録されています。このファイルを読み取ると、ファイルの内容は破棄されます。

検索フィルターとSA CLIメソッド

SA APIには、オブジェクト参照をパラメーターとして受け取るメソッドが多数あります。検索基準に基づいてオブジェクト参照を取得するには、findServerRefsやfindJobRefsなどのメソッドを呼び出します。たとえば、findServerRefsを呼び出すと、hostname属性にexample.comを持つすべてのサーバーを検索することができます。

検索の構文

findServerRefsなどのメソッドは、次の構文を取ります。

findオブジェクトRefs filter=[オブジェクトタイプ:]式

filterパラメーターには、検索条件を指定する式が含まれます。式は括弧または中括弧で囲みます。単純な式の構文を次に示します。

value-object.attribute operator value

この構文は単純化されています。完全な定義については、「フィルターの文法」(219ページ)を参照してください。

検索の例

SAオブジェクトタイプのほとんどには、対応する検索メソッドがあります。ここでは、そのうちいくつかの使用法を示します。他のSA CLIメソッドでの検索の使用法については、「サンプルスクリプト」(65ページ)を参照してください。

サーバーの検索

ホスト名にexample.comを含むサーバーを検索します。

```
cd /opsw/api/com/opsware/server/ServerService/method
./.findServerRefs:i \
filter='device:{ ServerVO.hostname CONTAINS example.com }'
```

use属性の値がUNKNOWNまたはPRODUCTIONのサーバーを検索します。

```
cd /opsw/api/com/opsware/server/ServerService/method
./.findServerRefs:i \
filter='{ ServerVO.use IN "UNKNOWN" "PRODUCTION" }'
```

次のbashスクリプトは、サーバーを検索し、それらのIDを一時ファイルに保存し、各IDを別のメソッド呼び出しのパラメーターとして指定する方法を示します。このスクリプトは、各Linuxサーバーが属するパブリックグループを表示します。

```
#!/bin/bash


TMPFILE=/tmp/server-list.txt

rm -f $TMPFILE


cd /opsw/api/com/opsware/server/ServerService/method


./.findServerRefs:i \

filter='{ ServerVO.osVersion CONTAINS Linux }' > $TMPFILE


for ID in `cat "$TMPFILE"`

do

  echo Server ID: $ID
```

```
  ./getDeviceGroups self:i=$ID
  echo
done
```

ジョブの検索

ここに示す例は、サーバー監査やポリシー修復などのジョブのIDを返します。

正常に完了したジョブを検索します。

```
cd /opsw/api/com/opsware/job/JobService/method
./.findJobRefs:i filter='job:{ job_status = "SUCCESS" }'
```

(job_statusに使用できる値の一覧については、『SA 10.50統合ガイド』の「ジョブ承認の統合」を参照してください)。

正常に完了したか、警告が発生して完了したジョブを検索します。

```
cd /opsw/api/com/opsware/job/JobService/method
./.findJobRefs:i \
filter='job:{ job_status IN "SUCCESS" "WARNING" }'
```

今日開始されたジョブを検索します。

```
cd /opsw/api/com/opsware/job/JobService/method
./.findJobRefs:i \
filter='job:{ JobInfoVO.startDate IS_TODAY "" }'
```

すべてのサーバー監査ジョブを検索します。

```
cd /opsw/api/com/opsware/job/JobService/method
./findJobRefs \
filter='job:{ JobInfoVO.description = "Server Audit" }'
```

IDが280039のサーバー上で実行されたジョブを検索します。

```
cd /opsw/api/com/opsware/job/JobService/method
./.findJobRefs:i filter='job:{ job_device_id = "280039" }'
```

今日のジョブの中で失敗したものを検索します。

```
cd /opsw/api/com/opsware/job/JobService/method
./.findJobRefs:i \
```

```
filter='job:{ (( JobInfoVO.startDate IS_TODAY "" ) \
& ( job_status = "FAILURE" )) }'
```

その他のオブジェクトの検索

ここでは、ソフトウェアポリシーおよびパッケージを検索する例を示します。

jdoeというSAユーザーによって作成されたソフトウェアポリシーを検索します。

```
cd /opsw/api/com/opsware/swmgmt/SoftwarePolicyService/method
./.findSoftwarePolicyRefs:i \
filter='{ SoftwarePolicyVO.createdBy CONTAINS jdoe }'
```

名前にismtoolが含まれるWindows 2003プラットフォーム用のMSIを検索します。

```
cd /opsw/api/com/opsware/pkg/UnitService/method
./.findUnitRefs:i \
filter='software_unit:{ ((UnitVO.unitType = "MSI") \
& ( UnitVO.name contains "ismtool" ) \
& ( software_platform_name = "Windows 2003" )) }'
```

名前が117170-01のSolarisパッチを検索します。

```
cd /opsw/api/com/opsware/pkg/solaris/SolPatchService/method
./.findSolPatchRefs:i filter='{name = 117170-01}'
```

名前にTestという文字列を含み、親フォルダー名がMy Stuffのフォルダーを検索します。

```
cd /opsw/api/com/opsware/folder/FolderService/method
./.findFolders:s \
filter='( ( FolderVO.name CONTAINS "Test" ) \
& (folder_parent_name = "My Stuff" ) )'
```

検索可能な属性と有効な演算子

値オブジェクトの属性の中には、検索フィルターで使用できないものがあります。たとえば、ServerVO.useで検索することはできますが、ServerVO.OsFlavorで検索することはできません。

オブジェクトタイプのどの属性が検索可能かを知るには、getSearchableAttributesメソッドを呼び出します。次の例は、検索式に指定できるServerVOの属性のリストを表示します。

```
cd /opsw/api/com/opsware/search/SearchService/method

./getSearchableAttributes searchableType=device
```

searchableTypeパラメーターは、オブジェクトタイプを示します。searchableTypeで使用可能な値を知るには、次のコマンドを入力します。

```
cd /opsw/api/com/opsware/search/SearchService/method

./getSearchableTypes
```

属性に対してどの演算子が有効かを知るには、getSearchableAttributeOperatorsメソッドを呼び出します。次の例は、属性ServerVO.hostnameに対して有効な演算子 (CONTAINS、INなど) のリストを表示します。

```
cd /opsw/api/com/opsware/search/SearchService/method

./getSearchableAttributeOperators searchableType=device \

searchableAttribute=ServerVO.hostname
```

サンプルスクリプト

ここでは、さまざまなSA CLIメソッドを呼び出す単純なbashスクリプトのコードリストを示します。これらのスクリプトは、コマンドラインでメソッドパラメーターを渡す方法を示しており、複合オブジェクトやselfパラメーターも使用されています。これらのスクリプトの例をコピーして再使用する場合は、コード内に書かれているオブジェクト名 (d04.example.com) を適切に変更してください。OGFSでスクリプトを作成して実行する手順のチュートリアルについては、「SA CLIメソッドのチュートリアル」(42ページ)を参照してください。

スクリプト「remediate_policy.sh」(70ページ)は、ソフトウェアポリシーを作成し、ポリシーにパッケージを追加し、最後の行で、startFullRemediateNowメソッドを呼び出してパッケージを管理対象サーバーにインストールします。

create_custom_field.sh

このスクリプトは、TestFieldAという名前のカスタムフィールド (仮想列) を作成し、フィールドをすべてのサーバーにアタッチし、1つのサーバーに対してこのフィールドの値を設定します。カスタムフィールドはアタッチされるまでSAクライアントには表示されません。カスタムフィールドは、サーバー、デバイスグループ、ソフトウェアポリシーに対して作成できます。カスタムフィールドを作成するには、SAユーザーが属するユーザーグループに仮想列の管理アクセス権が必要です。

カスタム属性と異なり、カスタムフィールドはそのタイプのすべてのインスタンスに適用されます。OGFSでカスタム属性を作成する例については、HPE SSOポータルで『Server Automationユーザーガイド』の「カスタム属性の管理」を参照してください。

create_custom_field.shスクリプトには、次のコードがあります。

```
#!/bin/bash
# create_custom_field.sh

cd /opsw/api/com/opsware/custattr/VirtualColumnService/method

# Create a virtual column.
# Remember the name because you cannot search for the
# displayName.
./create vo='{ name=TestFieldA type=SHORT_STRING \
displayName="Test Field A" }'

column_id=`./.findVirtualColumn:i name=TestFieldA`

echo --- column_id = $column_id

cd /opsw/api/com/opsware/server/ServerService/method

# Attach the column to all servers.
# All servers will have this custom field.
./attachVirtualColumn virtualColumn:i=$column_id

# Get the ID of the server named d04.example.com
devices_id=`./.findServerRefs:i \
filter=\
'device:{ ServerVO.hostname CONTAINS "d04.example.com" }'`

echo --- devices_id = $devices_id
```

```
# Set the value of the custom field (virtual column) for
# a specific server.
./setCustomField self:i=$devices_id fieldName=TestFieldA \
strValue="This is something."
```

create_device_group.sh

このスクリプトは、静的デバイスグループを作成し、そのグループにサーバーを追加します。次に、動的グループを作成し、そのグループにルールを設定し、グループのメンバーを更新します。最後のステートメントは、動的グループに属するデバイスのリストを表示します。

スクリプトのコードを次に示します。

```
#!/bin/bash
# create_device_group.sh

cd /opsw/api/com/opsware/device/DeviceGroupService/method

# Get the ID of the public root group (top of hierarchy).
public_root=‘./.getPublicRoot:i‘

# Create a public static group.
./create "vo={ parent:i=$public_root shortName=’Test Group A’ }"

# Get the ID of the group just created.
group_id=‘./.findDeviceGroupRefs:i \
filter=’{ DeviceGroupVO.shortName = "Test Group A" }’ ‘

echo --- group_id = $group_id

cd /opsw/api/com/opsware/server/ServerService/method

# Get the ID of the server named d04.example.com
```

```
devices_id=‘./.findServerRefs:i \
filter=\
’device:{ ServerVO.hostname CONTAINS "d04.example.com" }’‘

echo --- devices_id = $devices_id

cd /opsw/api/com/opsware/device/DeviceGroupService/method

# Add a server to the device group.
./addDevices \
self:i=$group_id devices:i=$devices_id

# Create a dynamic device group.
./create \
"vo={ parent:i=$public_root \
shortName=’Test Dyn B’ dynamic=true }"

# Get the ID of the device group.
dynamic_group_id=‘./.findDeviceGroupRefs:i \
filter=’{ DeviceGroupVO.shortName = "Test Dyn B" }’ ‘

echo --- dynamic_group_id = $dynamic_group_id

# Set the rule so that this group contains servers with
# hostnames containing the string example.com.
# The rule parameter has the same syntax as the filter
# parameter of the find methods.
./setDynamicRule self:i=$dynamic_group_id \
rule=’device:{ ServerVO.hostname CONTAINS example.com }’

# By default, membership in dynamic device groups is refreshed
```

```
# once
# an hour, so force the refresh now.
./refreshMembership selves:i=$dynamic_group_id now=true

# Display the names of the devices that belong to the group.
echo --- Devices in group:
./getDevices selves:i=$dynamic_group_id
```

create_folder.sh

このスクリプトは、/Test 1というフォルダーを作成し、ルート (/) フォルダーの下のフォルダーのリストを表示し、/Test 1/Test 2というサブフォルダーを作成します。作成したフォルダーは、SAクライアントのナビゲーションペインの「ライブラリ」の下に表示されます。

このスクリプトのコードを次に示します。

```
#!/bin/bash
# create_folder.sh

cd /opsw/api/com/opsware/folder/FolderService/method

# Get the ID of the root (top) folder.
root_id=`./.getRoot:i`

# Create a new folder under the root folder.
./create vo="{ name='Test 1' folder:i=$root_id }"

# Display the names of the folders under the root folder.
./getChildren self:i=$root_id

# Get the ID of the folder "/Test 1"
folder_id=`./.getFolderRef:i path="Test 1"`
```

```
# Create a subfolder.
./create vo="{ name='Test 2' folder:i=$folder_id }"

# Get the ID of the folder "/Test 1/Test 2"
folder_id=`./.getFolderRef:i path="Test 1" path="Test 2"`
echo folder_id = $folder_id
```

remediate_policy.sh

このスクリプトは、Test 2という名前の既存のフォルダーにTestPolicyAという名前のソフトウェアポリシーを作成し、ismtoolを含むパッケージをこのポリシーに追加し、ポリシーを1つのサーバー(グループでなく)にアタッチし、そのサーバーを修復します。修復操作は、パッケージをサーバーにインストールするジョブを起動します。ジョブの進行状況と結果はSAクライアントで確認できます。SA CLIメソッドでジョブを検索する例については、「ジョブの検索」(63ページ)を参照してください。

このスクリプトでは、SoftwarePolicyServiceのcreateメソッドで、platformsパラメーターの値がコードに直接書き込まれています。ここにあるスクリプト例では、できる限りオブジェクトを名前で検索し、値を直接書き込まないようにしています。プラットフォームの場合は、name属性での検索が困難です。この属性はSAクライアントに表示されるdisplayName属性(検索不可)とは異なっているからです。プラットフォームIDを知る最も簡単な方法は、twisterでPlatformService.findPlatformRefsメソッドをパラメーターなしで実行することです。

このスクリプトのupdateメソッドには、softwarePolicyItemsのIDが直接書き込まれています。ソフトウェアリポジトリに似た名前のパッケージが多数含まれている場合、このオブジェクトは名前で検索するのが困難だからです。IDを知る1つの方法は、SAクライアントを実行し、ファイル名やオペレーティングシステムなどのフィールドでソフトウェアを検索し、検索で見つかったパッケージを開き、パッケージのプロパティ表示にあるSA IDを記録することです。

次のリストでは、updateメソッドの中に不要な改行が入っています。このコードをコピーする場合、voパラメーターが1行になるようにスクリプトを編集してください。

remediate_policy.shスクリプトのソースコードを次に示します。

```
#!/bin/bash
# remediate_policy.sh

# Get the ID of the folder where the policy will reside.
cd /opsw/api/com/opsware/folder/FolderService/method
```

```
folder_id=`./.findFolders:i filter='{ FolderVO.name = "Test 2" }'`

cd /opsw/api/com/opsware/swmgmt/SoftwarePolicyService/method

# Create a software policy named TestPolicyA.
# This policy resides in the folder located in the preceding findFolders
# call.
# The platform for this policy is Windows 2008 (ID 160076)
./create vo="{ platforms:i=160076 name="TestPolicyA" \
folder:i=$folder_id lifecycle=AVAILABLE }"

policy_id=`./.findSoftwarePolicyRefs:i \
filter='{ SoftwarePolicyVO.name = "TestPolicyA" }'`

echo --- policy_id = $policy_id

# Call the update method to add a package to the software policy.
# The package ID for the "ismtool" msi installer is 4010001.

# Note that "force = true" is required.

./update self:i=$policy_id force=true \
vo='{ softwarePolicyItems:i=com.opsware.pkg.windows.MSIRef:4010001 }'

cd /opsw/api/com/opsware/server/ServerService/method

# Get the ID of the server named d04.opsware.com
devices_id=`./.findServerRefs:i \
filter='device:{ ServerVO.hostname CONTAINS "d04.opsware.com" }'`

echo --- devices_id = $devices_id
```

```
# Attach the policy to a single server (not a group).
./attachPolicies self:i=$devices_id \
policies:i=$policy_id

# Remediate the server to install the package in the policy.
job_id=`./.startFullRemediateNow:i self:i=$devices_id`

echo --- job_id = $job_id
```

remove_custom_field.sh

運用環境ではほとんどありませんが、テスト環境ではカスタムフィールドの削除が必要になることがあります。カスタムフィールドを削除するには、先にアタッチを解除しておく必要があります。

remove_custom_field.shのコードを次に示します。

```
#!/bin/bash
# remove_custom_field.sh

if [ !-n "$1" ]
   then
   echo "Usage: ‘basename $0‘ <name>"
   echo "Example: ‘basename $0‘ hmp"
   exit
fi

cd /opsw/api/com/opsware/custattr/VirtualColumnService/method

column_id=‘./.findVirtualColumn:i name=$1‘

echo --- column_id = $column_id
```

```
cd /opsw/api/com/opsware/server/ServerService/method

# Column must be detached before it can be removed.
./detachVirtualColumn virtualColumn:i=$column_id

cd /opsw/api/com/opsware/custattr/VirtualColumnService/method

# Remove the virtual column.
./remove self:i=$column_id
```

schedule_audit_task.sh

このスクリプトは、監査タスクを開始し、未来の日付にスケジュールします。SA CLIメソッドでは、日付パラメーターは次の構文で指定されます。

```
YYYY/MM/DD HH:MM:SS.sss
```

タスクを起動するメソッド`startAudit`は、監査を実行するジョブのIDを返します。SA CLIメソッドでジョブを検索する例については、「ジョブの検索」(63ページ)を参照してください。

`schedule_audit_task.sh`のコードを次に示します。

```
#!/bin/bash
# schedule_audit_task.sh

cd /opsw/api/com/opsware/compliance/sco/AuditTaskService/method

# Get the ID of the audit task to schedule.

audit_task_id=`./.findAuditTask:i \
filter='audit_task:{ (( AuditTaskVO.name BEGINS_WITH "HW check" ) \
& ( AuditTaskVO.createdBy = "gsmith" )) }'`

echo --- audit_task_id = $audit_task_id
```

```
# Schedule the audit task for Oct. 16, 2013.
# In the startDate parameter, note that the last delimiter for the time
# is a period, not a colon.

job_id=`./.startAudit:i self:i=$audit_task_id
schedule:s='{ startDate="2013/10/16 00:00:00.000" }' \
notification:s='{ onFailureOwner="sjones@opsware.com" \
onFailureRecipients="jdoe@opsware.com" \
onSuccessOwner="sjones@opsware.com" \
onSuccessRecipients="jdoe@opsware.com" }'`

echo --- job_id = $job_id
```

SA CLIメソッドの使用法情報の取得

将来のリリースでは、SA CLIメソッドで使用法情報を表示できるようになる予定です。それまでは、次に示す方法でAPIドキュメントまたはOGFSから必要な情報を得ることができます。

サービスのリストの表示

SA APIメソッドは、サービスによって分類されています。SA CLIメソッドに対して使用可能なサービスを知るには、OGSHセッションに次のコマンドを入力します。

```
cd /opsw/api/com/opsware
find .-name "*Service"
```

APIドキュメントでサービスのリストを表示するには、ブラウザーに次のURLを指定します。

```
https://OCCホスト:1032
```

OCCホストは、コマンドセンターコンポーネントを実行しているコアサーバーのIPアドレスまたはホスト名です。

APIドキュメントでのサービスの検索

OGFSのサービスのパスは、APIドキュメントのJavaパッケージ名に対応します。たとえば、OGFSのServerServiceメソッドは次のディレクトリにあります。

```
/opsw/api/com/opsware/server
```

APIドキュメントでは、これらのメソッドは次のインタフェースで定義されます。

```
com.opsware.server.ServerService
```

サービスのメソッドのリストの表示

OGFSでは、サービスのmethodディレクトリの内容のリストを表示できます。たとえば、ServerServiceのメソッド名を表示するには、次のコマンドを入力します。

```
ls /opsw/api/com/opsware/server/ServerService/method
```

APIドキュメントでは、次の手順でServerServiceのメソッドを表示できます。

左上のペインでcom.opsware.serverを選択します。

左下のペインでServerServiceを選択します。

メインペインで、下にスクロールしてメソッドを表示します。

メソッドのパラメーターのリストの表示

APIドキュメントで、前の項に示した手順を実行します。サービスインタフェースのページの「メソッドの詳細」の項で、パラメーターと戻り値の型を確認します(メソッドのパラメーターの詳細については、「コマンドラインでの引数渡し」(58ページ)を参照してください)。

値オブジェクトに関する情報の取得

APIドキュメントには、いくつかのサービスメソッドが値オブジェクト(VO)を引数に取るか値として返すことが記述されています。値オブジェクトにはデータメンバー(属性)が含まれます。たとえば、

ServerService.getServerVOメソッドはServerVOオブジェクトを返します。ServerVOに含まれる属性を知るには、次の手順を実行します。

APIドキュメントで、ServerVOリンクを選択します。このリンクはいくつかの場所にあります。

- getServerVOのメソッドシグネチャ
- com.opsware.serverのクラスのリスト (左下のペイン)
- 索引ページ。索引ページへのリンクは、APIドキュメントのメインペインの上部にあります。
- ServerVOページで、getterメソッドとsetterメソッドを見つけます。各getter-setterペアは、値オブジェクトの1つの属性に対応します。たとえば、getCustomerとsetCustomerは、ServerVOにcustomerという属性があることを示します。

属性が変更可能かどうかの判定

クライアントアプリケーションから変更できるオブジェクト属性は一部だけです。属性が変更可能かどうかを知るには、次の手順を実行します。

1. 前の項に示した手順で、APIドキュメントの値オブジェクトのページを開きます。
2. setterメソッドの「メソッドの詳細」の項で、「フィールドはクライアントから設定可能」と記されているかどうかを確認します。

サーバーやカスタマーなど、OGFSで表現されるSAオブジェクトの場合、attrディレクトリ内のファイルのアクセスタイプを調べることで、どの属性が変更可能かを知ることができます。読み取り/書き込み (rw) アクセスタイプを持つファイルは、変更可能な属性に対応します。たとえば、サーバーの変更可能な属性のリストを表示するには、次のコマンドを入力します。

```
cd /opsw/Server/@/server-name/attr
ls -l | grep rw
```

属性がフィルタークエリで使用可能かどうかの判定

値オブジェクトの属性がフィルタークエリ (検索) で使用可能かどうかを知るには、次の手順を実行します。

1. APIドキュメントで、値オブジェクトのページを開きます。
2. 属性に対応するgetterメソッドの「メソッドの詳細」の項で、「フィールドはフィルタークエリに使用可能」と記されているかどうかを確認します。

属性が検索可能かどうかをOGSHセッション内で知るには、「検索可能な属性と有効な演算子」(64ページ)に記されている方法を使用します。

PytwistによるPython APIアクセス

Pytwistの概要

Pytwistは、管理対象サーバーおよびカスタム拡張からSA APIにアクセスするためのPythonライブラリのセットです(twistはWebサービスデータアクセスエンジンの内部名です)。管理対象サーバーに対して、Pytwistを使用してSA APIを呼び出すPythonスクリプトを用意すれば、エンドユーザーはスクリプトをDSEまたはISMコントロールとして呼び出すことができます。カスタム拡張は、HPE SAプロフェッショナルサービスによって作成され、コマンドエンジン(way)で動作するPythonスクリプトです。Pytwistを使うと、SAデータモデルに最近追加されたフォルダーやソフトウェアポリシーなどにアクセスできます。これらはコマンドエンジンスクリプトからはアクセスできません。

このトピックの内容は、SAのデータモデル、カスタム拡張、エージェント、Pythonプログラミング言語に関する知識がある開発者やコンサルタントを対象としています。

Pytwistのセットアップ

この項の例を試す前に、環境が以下に詳述するセットアップ要件を満たすことを確認してください。

Pytwistでサポートされるプラットフォーム

Pytwistは、管理対象サーバーとコアサーバーでサポートされます。これらのサーバーでサポートされるオペレーティングシステムのリストについては、

PytwistはPythonバージョン2.7.10に依存します。これはSAのエージェントとカスタム拡張が使用するバージョンと同じです。

WebサービスやJava RMIクライアントと異なり、Pytwistクライアントは内部SAライブラリに依存します。クライアントプログラムが管理対象サーバーでもコアサーバーでもないサーバーからSA APIにアクセスする必要がある場合は、PytwistでなくWebサービスまたはJava RMIクライアントを使用してください。

Pytwistのアクセス要件

Pytwistは、Webサービスデータアクセスエンジンを実行しているコアサーバーのポート1032にアクセスする必要があります。エンジンはデフォルトでポート1032をリッスンします。

Pytwistライブラリのインストール

Pytwistライブラリはエージェントライブラリに含まれているため、インストールする必要はありません。

Pytwistの例

ここに示すPythonコードの例は、管理対象サーバーからの情報の取得、フォルダーの作成、ソフトウェアポリシーの修復の方法を示します。それぞれのPytwistの例は、次の操作を実行します。

1. パッケージのインポート。

   SA API名前空間のオブジェクト (`Filter`など) をインポートする場合、パスではJavaパッケージ名の前に`pytwist`が付きます。`get_server_info.py`の例の`import`ステートメントを次に示します。

   ```
   import sys
   from pytwist import *
   from pytwist.com.opsware.search import Filter
   ```

2. `TwistServer`オブジェクトを作成します。

   ```
   ts = twistserver.TwistServer()
   ```

   メソッドの引数に関する情報については、TwistServerメソッドの構文を参照してください。

3. サービスへの参照を取得します。

   サービスのPythonパッケージ名はJavaパッケージ名と同じですが、先頭の`opsware.com`がありません。たとえば、Javaの`com.opsware.server.ServerService`パッケージはPytwistの`server.ServerService`に対応します。

   serverservice = ts.server.ServerService

4. サービスのSA APIメソッドを呼び出します。

   ```
   filter = Filter()
   . . .
   servers = serverservice.findServerRefs(filter)
   . . .
   for server in servers: vo = serverservice.getServerVO(server)
   . . .
   ```

get_server_info.py

このスクリプトは、ホスト名がコマンドライン引数を含むすべての管理対象サーバーを検索します。検索メソッド findServerRefsは、サーバー永続的オブジェクトへの参照の配列を返します。各参照に対して、getServerVOメソッドは値オブジェクト (VO) を返します。これはサーバーの属性を保持するデータ表現です。get_server_info.pyスクリプトのコードを次に示します。

```
#!/opt/opsware/agent/bin/python
# get_server_info.py

# Search for servers by partial hostname.

import sys
from pytwist import *
from pytwist.com.opsware.search import Filter

# Check for the command-line argument.
if len(sys.argv) < 2:
       print "You must specify part of the hostname as the search target."
       print "Example: " + sys.argv[0] + " " + "opsware.com"
       sys.exit(2)

# Construct a search filter.
filter = Filter()
filter.expression = 'device_hostname *=* "%s" ' % (sys.argv[1])

# Create a TwistServer object.
ts = twistserver.TwistServer()

# Get a reference to ServerService.
serverservice = ts.server.ServerService

# Perform the search, returning a tuple of references.
servers = serverservice.findServerRefs(filter)

if len(servers) < 1:
       print "No matching servers found"
       sys.exit(3)

# For each server found, get the server's value object (VO)
# and print some of the VO's attributes.
for server in servers:
       vo = serverservice.getServerVO(server)
       print "Name: " + vo.name
       print "Management IP: " + vo.managementIP
       print "OS Version: " + vo.osVersion
```

create_folder.py

このスクリプトは、createPathメソッドを呼び出して、/TestA/TestBという名前のフォルダーを作成します。createPathのpathパラメーターにはスラッシュは含まれません。pathの各文字列要素がフォルダー内のレベルを示します。次に、スクリプトはルートフォルダー直下のすべてのフォルダーの名前を取得して出力します。create_folder.pyスクリプトのリストを次に示します。

```
#!/opt/opsware/agent/bin/python
# create_folder.py

# Create a folder in SA.

import sys
from pytwist import *

# Create a TwistServer object.
ts = twistserver.TwistServer()

# Get a reference to FolderService.
folderservice = ts.folder.FolderService

# Get a reference to the root folder.
rootfolder = folderservice.getRoot()
# Construct the path of the new folder.
path = 'TestA', 'TestB'

# Create the folder /TestA/TestB relative to the root.
folderservice.createPath(rootfolder, path)

# Get the child folders of the root folder.
rootchildren = folderservice.getChildren(rootfolder,
'com.opsware.folder.FolderRef')

# Print the names of the child folders.
for child in rootchildren:
       vo = folderservice.getFolderVO(child)
       print vo.name
```

remediate_policy.py

このスクリプトは、ソフトウェアポリシーを作成し、サーバーにアタッチし、ポリシーを修復します。いくつかの名前はスクリプトに直接書き込まれています。プラットフォーム、サーバー、親フォルダーがそうです。オプショ

ンで、ポリシー名をコマンドラインに指定することもできます。これはスクリプトを何回も実行する場合に便利です。ソフトウェアポリシーのプラットフォームは、ポリシーに含まれるパッケージのOSに一致する必要があります。したがって、スクリプトに直接書き込まれているプラットフォーム名を変更する場合は、unitfilter.expression内の名前も変更する必要があります。

次のリストにはいくつか不要な改行があります。このコードをコピーした場合は、必ず不要な改行を削除してから実行してください。不要な改行がある個所には "NOTE" で始まるコメントが付いています。

```
#!/opt/opsware/agent/bin/python
# remediate_policy.py

# Create, attach, and remediate a software policy.

import sys
from pytwist import *
from pytwist.com.opsware.search import Filter
from pytwist.com.opsware.swmgmt import SoftwarePolicyVO

# Initialize the names used by this script.
foldername = 'TestB'
platformname = 'Windows 2003'
servername = 'd04.example.com'
# If a command-line argument is specified,
# use it as the policy name
if len(sys.argv) == 2:
       policyname = sys.argv[1]
else:
       policyname = 'TestPolicyA'

# Create a TwistServer object.
ts = twistserver.TwistServer()
ts.authenticate("SAUser", "SAPassword")

# Get the references to the services used by this script.
folderservice = ts.folder.FolderService
swpolicyservice = ts.swmgmt.SoftwarePolicyService
serverservice = ts.server.ServerService
unitservice = ts.pkg.UnitService
platformservice = ts.device.PlatformService

# Search for the folder that will contain the policy.
folderfilter = Filter()
folderfilter.expression = 'FolderVO.name = %s' % foldername
folderrefs = folderservice.findFolderRefs(folderfilter)
if len(folderrefs) == 1:
       parent = folderrefs[0]
elif len(folderrefs) < 1:
       print "No matching folders found."
       sys.exit(2)
```

```
else:
       print "Non-unique folder name: " + foldername
       sys.exit(3)
# Search for the reference to the platform "Windows Server 2003."
platformfilter = Filter()
platformfilter.objectType = 'platform'
# Because the platform name contains spaces,
# it’s enclosed in double quotes
# NOTE: The following code line has a bad line break.
# The assignment statement should be on a single line.
platformfilter.expression = 'platform_name = "%s"' % platformname
platformrefs = platformservice.findPlatformRefs(platformfilter)

if len(platformrefs) == 0:
       print "No matching platforms found."
       sys.exit(4)

# Search for the references to some software packages.
unitfilter = Filter()
unitfilter.objectType = 'software_unit'
# NOTE: The following code line has a bad line break.
# The assignment statement should be on a single line.
unitfilter.expression = '((UnitVO.unitType = "MSI") & ( UnitVO.name contains
"ismtool" ) & ( software_platform_name = "Windows 2003" ))'
unitrefs = unitservice.findUnitRefs(unitfilter)

# Create a value object for the new software policy.
vo = SoftwarePolicyVO()
vo.name = policyname
vo.folder = parent
vo.platforms = platformrefs
vo.softwarePolicyItems = unitrefs

# Create the software policy.
swpolicyvo = swpolicyservice.create(vo)

# Search by hostname for the reference to a managed server.
serverfilter = Filter()
serverfilter.objectType = 'server'
# NOTE: The following code line has a bad line break.
# The assignment statement should be on a single line.
serverfilter.expression = 'ServerVO.hostname = %s' % servername
serverrefs = serverservice.findServerRefs(serverfilter)

if len(serverrefs) == 0:
       print "No matching servers found."
       sys.exit(5)

# Create an array that has a reference to the
```

```
# newly created policy.
swpolicyrefs = [1]
swpolicyrefs[0] = swpolicyvo.ref

# Attach the software policy to the server.
swpolicyservice.attachToPolicies(swpolicyrefs, serverrefs)

# Remediate the policy and the server.
# NOTE: The following code line has a bad line break.
# The assignment statement should be on a single line.
jobref = swpolicyservice.startRemediateNow(swpolicyrefs, serverrefs)
print "The remediation job ID is %d" % jobref.id
```

Pytwistの例の仮想化

この項では、SA APIを使用した仮想マシン (VM) の作成とデプロイ例について説明します。仮想化のその他の例については、HPE SSOポータルで『Server Automationユーザーガイド』を参照してください。

createVM_WithOSBP.py

この基本例は、CDブートを使用して静的IPの構成で、VMware vCenter上にVMを作成します。

以下の例では、すべてのプロパティは設定されていません。自分のユースケースのためのプロパティの理解と設定方法については、APIドキュメント (javadocs) を参照してください。

```
#!/opt/opsware/agent/bin/python
from pytwist import twistserver
from pytwist.com.opsware.locality import CustomerRef, RealmRef
from pytwist.com.opsware.osprov import OSBuildPlanRef
from pytwist.com.opsware.pkg import UnknownPkgRef
from pytwist.com.opsware.v12n import AdapterIPSettings, V12nHypervisorRef, \
       V12nHypervisorService, V12nInventoryFolderRef, V12nResourcePoolRef, \
       V12nResourcePoolRef, V12nVIManagerService, VirtualCpuConfig,
VirtualDevice, \
       VirtualDeviceChangeConfig, VirtualDeviceTypeConstant,
VirtualHardwareConfigSpec, \
       VirtualMemoryConfig, VirtualServerCDProvisioningSpec,
VirtualServerComputeSpec, \
       VirtualServerConfigSpec, VirtualServerCreateSpec,
VirtualStorageDeviceConstant, \
       VirtualStorageDeviceHWConfig
from pytwist.com.opsware.v12n.vmware import V12nDatastoreRef, \
       VmwareVirtualInterfaceBacking, VmwareVirtualNicHWConfig, \
       VmwareVirtualServerDetails, VmwareVirtualServerStorageSpec, \
       VmwareVirtualStorageFileBacking
import time


# This is a bare bones example of creating a Virtual Machine on a VMware
# vCenter while booting from CD with Static IP configuration.It also
# provisions the Virtual Machine with the give OS Build Plan.For more
# detailed information please refer to the java doc. All the properties have
# not been set in the example below, please review the java doc to understand
# and set the properties for your use case.
```

```
#This method constructs the create specification to create the Virtual
# Machine and provision it.
def constructCreateSpec():

        #Construct VmwareVirtualServerDetails
        detail = VmwareVirtualServerDetails()
        # Virtual Machine Name
        detail.name = "Test VM"
        # Description for the Virtual Machine
        detail.description = "Sample test create VM"
        # This is the key for the guest operating system that will installed on
        # the Virtual Machine.
        #V12nVIManagerService.getGuestOSList() provides the supported list for
        # the given V12n Manager and hypervisor.
        detail.guestId = "rhel6Guest"
        # This is folder where the VM will reside in you can see the list of
        # folders at V12nInventoryFolderService.findV12nInventoryFolderRefs() it
        # is the inventory location of the Virtual Machine
        folder = V12nInventoryFolderRef(2020001)
        detail.inventoryFolderRef = folder


        # Configure the number of Virtual processors on the Virtual Machine
        cpuConfig = VirtualCpuConfig()
        cpuConfig.virtualCpuCount = 1

        # Configure the Memory for the Virtual Machine
        memoryConfig = VirtualMemoryConfig()
        memoryConfig.size = 1024*1024*1024

        # Configure NICs
        # Construct the virtual device of type network i.e a NIC
        virtualNetworkDevice = VirtualDevice()
        virtualNetworkDevice.type = VirtualDeviceTypeConstant.NETWORK
        # A unique identifier for the virtual device
        virtualNetworkDevice.key = "4001"
        backingNetwork = VmwareVirtualInterfaceBacking()
        # This is the port group that the nic will be assigned to
        backingNetwork.portGroup = "VLAN 625"

        hwConfigNetwork = VmwareVirtualNicHWConfig()
        # The kind of network adapter to use, other options are listed in
        # VmwareVirtualNicHWConfig
        hwConfigNetwork.adapterType = VmwareVirtualNicHWConfig.E1000
        hwConfigNetwork.macAddressIsDynamic = True

        virtualNetworkDevice.hwConfig = hwConfigNetwork
        virtualNetworkDevice.backingInfo = backingNetwork
```

```
        virtualNetworkDevice.connected = True
        virtualNetworkDevice.startConnected = True

        # Configure Hard Disk
        virtualDiskDevice = VirtualDevice()
        virtualDiskDevice.type = VirtualDeviceTypeConstant.STORAGE

        backingStorage = VmwareVirtualStorageFileBacking()

        # This is Ref for the data store on the hypervisor where the VM will be
        # hosted.The list of datastores associated with the Hypervisors are
        # listed at V12nHypervisorService.getV12nHypervisorVO() under storage
        # config
        dataStoreRef = V12nDatastoreRef(90001)
        backingStorage.datastore = dataStoreRef
        backingStorage.lazyAllocation = True

        hwConfigStorage = VirtualStorageDeviceHWConfig()
        hwConfigStorage.capacity = 10*1024*1024*1024
        hwConfigStorage.usageType =
VirtualStorageDeviceConstant.USAGE_TYPE_DISK_DRIVE

        virtualDiskDevice.hwConfig = hwConfigStorage
        virtualDiskDevice.backingInfo = backingStorage

        # Add both the virtual devices to be created, i.e. the hard disk and the
        # nic
        virtualDvcs_toAdd = []
        virtualDvcs_toAdd.append(virtualNetworkDevice)
        virtualDvcs_toAdd.append(virtualDiskDevice)
        deviceChange = VirtualDeviceChangeConfig()
        deviceChange.addList = virtualDvcs_toAdd

        # Finalize the Config Spec
        configSpec = VirtualServerConfigSpec()
        configSpec.detail = detail
        configSpec.virtualHardware = VirtualHardwareConfigSpec()
        configSpec.virtualHardware.cpuConfig = cpuConfig
        configSpec.virtualHardware.memoryConfig = memoryConfig
        configSpec.virtualHardware.deviceChange = deviceChange

        # Constructing the Compute Spec
        computeSpec = VirtualServerComputeSpec()
        # This is the hypervisor hosting the VM
        hypervisorRef = V12nHypervisorRef(2030001)
        computeSpec.computeProviderRef = hypervisorRef
        # This is resource pool on the hypervisor/cluster that the VM belongs to
        # It can be retrieved by using hypervisorVO.children or the Cluster
        # children
```

```
resourcePool = V12nResourcePoolRef(2040001)
computeSpec.resourcePoolRef = resourcePool

storageSpec = VmwareVirtualServerStorageSpec()
storageSpec.datastore = dataStoreRef

# This example deals with provisioning a VM through CD boot and with
# static IP configuration.The example deals setting the boot ISO and
# network information to be used.
#All the information for this is contained in the
# VirtualServerCDProvisioningSpec

# Set all the network information
gateways =[]
gw ="192.168.135.33"
gateways.append(gw)
dnsServers =[]
dnsServer = "192.168.2.13"
dnsServers.append(dnsServer)


interfaces =[]
interface = AdapterIPSettings()

# Construct the network interface
interface.useDHCP=False
# Note this is the virtual device we have created above, we use the same
# device key to indicate to provisioning which virtual device is to be
# used for provisioning
interface.virtualDeviceKey="4001"
interface.gateways=gateways
interface.ipAddress="192.168.135.45"
interface.netmask="255.255.255.224"
interface.dnsServerList=dnsServers

interfaces.append(interface)

# This is the boot ISO Ref that will be used to get the server into
# maintenance mode
# The name and the id need to match the packages on the core.
# Use the UnitService.findUnitRefs() to find the boot ISO's
bootISORef = UnknownPkgRef(5340001)
bootISORef.name="HPSA_linux_boot_cd.iso"
# The realm assigned to the Virtual Machine will be the realm of the
# Virtualization Service
realmRef = RealmRef(30001)
# The OS Build Plan that needs be run on the Virtual Machine after the VM
# has been created.
osbpRef = OSBuildPlanRef(580001)
```

```
        provisioningSpec = VirtualServerCDProvisioningSpec()

        provisioningSpec.bootISORef = bootISORef
        provisioningSpec.interfaces = interfaces
        provisioningSpec.realmRef = realmRef
        provisioningSpec.oSBuildPlanRef = osbpRef

        # Finally put together all the information to be set on the Create
        # Specification
        createSpec = VirtualServerCreateSpec()
        createSpec.configSpec = configSpec
        createSpec.computeSpec = computeSpec
        createSpec.storageSpec = storageSpec

        createSpec.provisioningSpec = provisioningSpec
        #Set the customer to be associated with the Virtual Machine
        customer = CustomerRef(9)
        createSpec.setCustomerRef(customer)
        return createSpec

def createVirtualMachine():
        twist = twistserver.TwistServer()
        twist.authenticate("hp", "opsware")
        vmService = twist.v12n.V12nVirtualServerService
        createSpec = constructCreateSpec()
        jobRef = vmService.startCreate(createSpec,4*60*60,"Sample create
VM",None, None)


createVirtualMachine()
```

deployVM.py

この基本例は、VMware vCenter上のVMテンプレートからVMをデプロイし、デプロイしたVMのゲストOSをカスタマイズする方法を示します。

以下の例では、すべてのプロパティは設定されていません。自分のユースケースのためのプロパティの理解と設定方法については、APIドキュメント (javadocs) を参照してください。

```
#!/opt/opsware/agent/bin/python
from pytwist import twistserver
from pytwist.com.opsware.locality import CustomerRef, RealmRef
from pytwist.com.opsware.osprov import OSBuildPlanRef
from pytwist.com.opsware.pkg import UnknownPkgRef
from pytwist.com.opsware.v12n import AdapterIPSettings, V12nHypervisorRef, \
```

```
        V12nHypervisorService, V12nInventoryFolderRef, V12nResourcePoolRef, \
        V12nResourcePoolRef, V12nVIManagerService, VirtualCpuConfig,
VirtualDevice, \
        VirtualDeviceChangeConfig, VirtualDeviceTypeConstant,
VirtualHardwareConfigSpec, \
        VirtualMemoryConfig, VirtualServerCDProvisioningSpec,
VirtualServerComputeSpec, \
        VirtualServerConfigSpec, VirtualServerCreateSpec,
VirtualStorageDeviceConstant, \
        VirtualStorageDeviceHWConfig, V12nVirtualServerTemplateRef, \
        VirtualServerCloneSpec, VirtualServerGuestCustomizationSpec
from pytwist.com.opsware.v12n.vmware import V12nDatastoreRef, \
        VmwareVirtualInterfaceBacking, VmwareVirtualNicHWConfig, \
        VmwareVirtualServerDetails, VmwareVirtualServerStorageSpec, \
        VmwareVirtualStorageFileBacking
import time

# This is a bare bones example of deploying a Template VMware vCenter.It
# deploys the template and then guest customizes the deployed Virtual Machine.
# For more detailed information please refer to the java doc. All the
# properties have not been set in the example below, please review the java
# doc to understand and set the properties for your use case.

#This method constructs the deploy specification to deploy the Template and
# customizes it.
def constructDeploySpec(sourceTemplateVO):

        #Construct the Deploy Spec
        clonespec = VirtualServerCloneSpec()

        clonespec.computeSpec = VirtualServerComputeSpec()
        # This is the hypervisor hosting the VM
        targetHypervisorRef = V12nHypervisorRef(2030001)
        clonespec.computeSpec.computeProviderRef = targetHypervisorRef

        computeSpec = VirtualServerComputeSpec()
        # This is the resource pool on the hypervisor/cluster that the VM belongs to
        # It can be retrieved by using hypervisorVO.children or the Cluster
        # children
        targetResourcePoolRef = V12nResourcePoolRef(2040001)
        computeSpec.resourcePoolRef = targetResourcePoolRef
        clonespec.computeSpec.resourcePoolRef = targetResourcePoolRef

        storageSpec = VmwareVirtualServerStorageSpec()
        dataStoreRef = V12nDatastoreRef(90001)
        storageSpec.datastore = dataStoreRef
        clonespec.storageSpec = storageSpec
        # Construct VmwareVirtualServerDetails
        detail = VmwareVirtualServerDetails()
```

```
        # Virtual Machine Name
        detail.name = "Test Deploy VM"
        # Description for the Virtual Machine
        detail.description = "Sample Deploy create VM"

        # This is the folder where the VM will reside in. You can see the list of
        # folders at V12nInventoryFolderService.findV12nInventoryFolderRefs().It
        # is the inventory location of the Virtual Machine
        targetFolderRef = V12nInventoryFolderRef(2020001)
        detail.inventoryFolderRef = targetFolderRef
        configSpec = VirtualServerConfigSpec()
        configSpec.detail = detail
        clonespec.configSpec=configSpec

        # Create the Guest Customization Spec, this is needed to customized the
        # deployed VM so that it does not use the network settings and host name
        # of the source template
        # In this example all the interfaces are set to DHCP but you can
        # customize each of the interfaces by either providing static or DHCP
        # configuration details
        interfaces = createInterfaces(sourceTemplateVO)
        # The realm assigned to the Virtual Machine will be the realm of the
        # Virtualization Service
        realmRef = RealmRef(30001)
        clonespec.guestCustomizationSpec =
createGuestCustomizationSpec("testDeployVM",realmRef,interfaces)
        clonespec.setPowerOn(True)
        # Set the customer to be associated with the Virtual Machine
        customerRef = CustomerRef(9)
        clonespec.customerRef = customerRef
        return clonespec

def createGuestCustomizationSpec(newVmNameVal,realmRef,interfaces):
        gcSpec = VirtualServerGuestCustomizationSpec()
        gcSpec.computerName = newVmNameVal
        gcSpec.interfaces = interfaces
        gcSpec.realmRef = realmRef
        return gcSpec

def createInterfaces(virtualServerVO):
        interfaces = []
        virtualDevices = virtualServerVO.virtualHardware.deviceList
        vNICs = [vd for vd in virtualDevices if vd.type ==
VirtualDeviceTypeConstant.NETWORK]
        for vNIC in vNICs:
                intf = AdapterIPSettings()
                intf.useDHCP = True
                intf.hardwareAddress = vNIC.hwConfig.macAddress
                intf.virtualDeviceKey = vNIC.key
```

```
                interfaces.append(intf)
        return interfaces


def deployVirtualMachine():
        twist = twistserver.TwistServer()
        twist.authenticate("hp", "opsware")
        vmTemplateService = twist.v12n.V12nVirtualServerTemplateService
        vmService = twist.v12n.V12nVirtualServerBaseService
        sourceTemplateRef = V12nVirtualServerTemplateRef(1520001)
        sourceTemplateVO =
vmService.getV12nVirtualServerBaseVO(sourceTemplateRef)
        deploySpec = constructDeploySpec(sourceTemplateVO)
        jobRef =
vmTemplateService.startDeploy(sourceTemplateRef,deploySpec,30*60,"Sample
Deploy VM",None, None);

deployVirtualMachine()
```

Pytwistの詳細

ここでは、Pytwistに固有の動作と構文について説明します。

認証モード

Pytwistクライアントの認証モードは、クライアントからアクセスできるSAの機能とリソースに影響するため重要です。Pytwistクライアントは次のいずれかのモードで動作します。

- **認証済み**: クライアントは`TwistServer`オブジェクトに対して`authenticate(username, password)`メソッドを呼び出しています。`authenticate`メソッドを呼び出した後、クライアントは、SAクライアントにログオンするエンドユーザーと同様に、`username`パラメーターで指定されたSAユーザーとして認証されます。
- **非認証**: クライアントは`TwistServer.authenticate`メソッドを呼び出していません。管理対象サーバー上で、クライアントはエージェント証明書を制御するデバイスと同様に認証されます。カスタム拡張で使用した場合、非認証Pytwistクライアントはコマンドエンジン証明書にアクセスできる必要があります。カスタム拡張と証明書に関する詳細については、テクニカルサポート担当者にお問い合わせください。

TwistServerメソッドの構文

`TwistServer`メソッドは、クライアントからWebサービスデータアクセスエンジンへの接続を構成します(呼び出しの例については、Pytwistの例を参照してください)。`TwistServer`のすべての引数はオプションです。次の表に、引数のデフォルト値を示します。

TwistServerメソッドの引数

引数	説明	デフォルト
host	接続先のホスト名。	twist
port	接続先のポート番号。	1032
secure	接続にhttpsを使用するかどうか。次の値を指定できます。1 (true) または0 (false)。	1
ctx	接続のSSLコンテキスト。	なし(「認証モード」(94ページ)も参照)。

TwistServerオブジェクトの作成時には、クライアントはサーバーとの接続を確立しません。したがって、接続の問題が存在する場合、クライアントがauthenticateまたはSA APIメソッドを呼び出すまで問題は明らかになりません。

エラー処理

TwistServer.authenticateメソッドまたはSA APIメソッドで問題が発生した場合、Python例外が発生します。これらの例外は、次の例に示すようにexcept句で捕捉できます。

```
# Create the TwistServer object.
ts = twistserver.TwistServer(localhost)
# Authenticate by passing an SA user name and password.
try:
       ts.authenticate('jdoe', 'secretpass')
except:
       print "Authentication failed."
       sys.exit(2)
```

Javaパッケージ名およびデータ型のPytwistへのマッピング

Pytwistインタフェースは Python用ですが、SA APIはJavaで作成されています。2つのプログラミング言語の間の違いのために、Pytwistクライアントはここに記すマッピングルールに従う必要があります。

SA APIドキュメントでは、Javaパッケージ名はcom.opswareで始まります。Pytwistでパッケージ名を指定する場合、次の例のように先頭にpytwistを挿入します。

```
from pytwist.com.opsware.compliance.sco import *
```

SA APIドキュメントでは、メソッドのパラメーターと戻り値がJavaデータ型で指定されています。次の表に、PytwistでAPIメソッドを呼び出す場合のJavaデータ型からPythonへのマッピング方法を示します。

JavaからPythonへのデータ型のマッピング

SA APIでのJavaデータ型	pytwistでのPythonデータ型
Boolean	真の場合は整数1、偽の場合は整数0。
Object[] (オブジェクト配列)	APIメソッド呼び出しの入力パラメーターでは、オブジェクト配列はPythonのタプルまたはリストです。APIメソッド呼び出しからの出力では、オブジェクト配列はPythonタプルで返されます。

JavaからPythonへのデータ型のマッピング (続き)

SA APIでのJavaデータ型	pytwistでのPythonデータ型
マップ	辞書
`Date`	エポック (1970年1月1日 午前0時) からのミリ秒数を表す`long`データ型。

自動化プラットフォーム拡張 (APX)

ここでは、自動化プラットフォーム拡張 (APX) の作成と管理の方法を説明します。これは一般には単に「拡張」と呼ばれます。APXは、シェルスクリプト、Python、Perl、PHPなどスクリプトベースのプログラミングツールの知識を持つユーザー向けに、SAの機能を拡張し、SAとの緊密な統合を可能にするアプリケーションを作成するフレームワークを提供します。SAでは、次の2つのタイプのAPXを使用できます。

- **Program APX** (**Script APX**とも呼ばれる) は、Global File System (OGFS) で実行され、すべてのOGFS機能を使用できます。一般的なプログラミング方法でSA APIを使用し、コアの管理対象サーバーにアクセスして新しいカスタム機能を実装できます。たとえば、管理対象サーバーからBIOS情報を収集し、シェルコマンドを使用してカスタムフィールドの値を設定するAPXを作成することができます。「Program APX」(99ページ)を参照してください。
- **Web APX**では、Webベースのアプリケーションを作成し、GET URLまたはPOST URLを使用してApache 2.xプロセスまたはは CGI/PHPスクリプトを呼び出すことができます。Web APXには、画像などの静的なWebリソースを含めることも、CGIまたはPHPを使用して動的なコンテンツを生成することもできます。「Web APX」(100ページ)を参照してください。

APXを使えば、管理対象環境に関するデータにアクセスし、そのデータをWebアプリケーション、スクリプト、プログラム、およびその他のアプリケーションと共有して処理することができます。APXの利点のいくつかを次に示します。

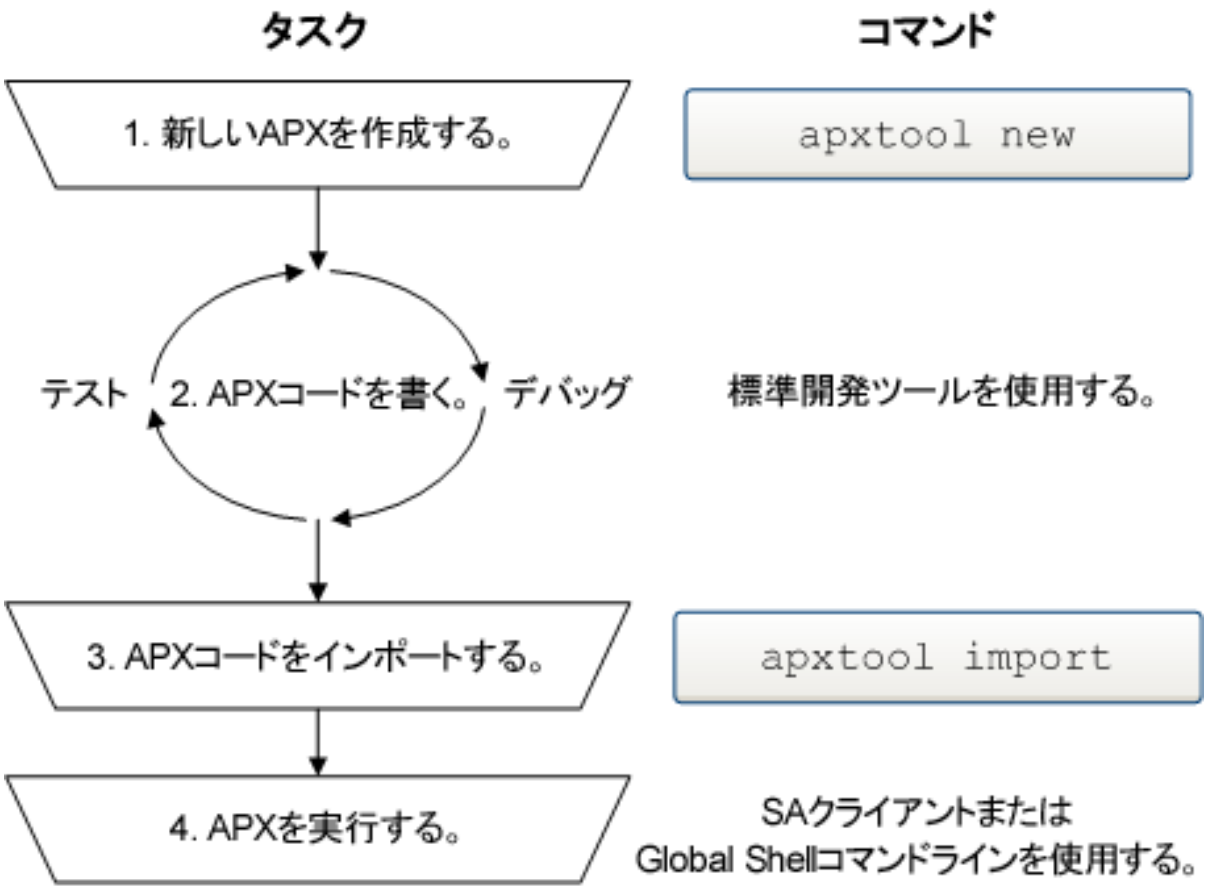
- SAライブラリにリストされ、SAクライアントから使用できます。
- バージョン管理によって一意に識別され、管理されます。
- SAのセキュリティモデルをフルに利用できるため、セキュリティを確保できます。APXでは、必要な場合、APXセッション中にセキュリティを確保しながら一時的にユーザーのアクセス権を通常のデフォルトからエスカレートすることができます。
- SAコア内部およびコア間に拡張可能です。
- サーバーに自動的にプッシュするようにスケジュールできます。
- 監査可能です。
- SAプラットフォームのアップグレードの影響を受けません。アップグレード後もAPXを作成し直す必要はありません。

APX拡張の使用に関する詳細については、『SA 10.50ユーザーガイド』の「SAの拡張の実行」を参照してください。.また、APX拡張はSA Global Shellからも実行できるので、『SA 10.50ユーザーガイド』の「SA Global Shell」も参照してください。

APXの作成

次の図は、APXの作成の基本的な手順と、そこで使用するコマンドを示します。Web APXの作成方法のチュートリアルについては、「チュートリアル: WebアプリケーションAPXの作成」(123ページ)を参照してください。Program APXの作成方法のチュートリアルについては、「チュートリアル: Program APXの作成」(130ページ)を参照してください。

APXの作成



1. 新しいAPXを作成するには、`apxtool new`コマンドを使用します。このコマンドはテンプレートファイルのセットを作成します。これらのファイルを編集して、独自のAPXを作成できます。

   `apxtool new`コマンドでは、オプションで新しいAPXを登録することもできます。APXを登録すると、そのAPXの名前がSAで予約されます。このステップでAPXを登録しない場合、この後のステップ3で`apxtool import`コマンドを使用して登録できます。

「apxtoolコマンド」(108ページ)を参照してください。

2. APXテンプレートファイルを作成したら、`apxtool new`コマンドで作成されたテンプレートファイルを変更し、場合によっては独自のファイルを追加して、APXコードを開発します。APXコードをテストして、動作を確認することもできます。
3. APXコードをテストしたら、`apxtool import`コマンドを使用してSAにインポートします。
4. APXは、SAクライアントまたはGlobal Shellコマンドラインから実行します。

   - SAクライアントから: [**ライブラリ**] > [**タイプ別**] タブ > [**拡張子**] > [**プログラム**] を選択します。APXを選択します。[**アクション**] > [**実行**] メニューを選択します。
   - Global Shellコマンドラインから: SAクライアントから [ツール] > [Global Shell] メニューを選択して、Global Shellを開きます。次のコマンドを入力してAPXを実行します。
     `/opsw/apx/bin/<APX名>`

     詳細については、HPE SSOポータルで『Server Automationユーザーガイド』の「SAの拡張の実行」および「SA Global Shell」を参照してください。

VMware ESXiサーバーで動作するAPX拡張を作成する場合、APX拡張はWebサービスインタフェースを通じてリモートでESXiサーバーと通信する必要があります。VMware ESXiサーバーの詳細については、HPE SSOポータルで『Server Automationユーザーガイド』の「仮想サーバーの管理」を参照してください。

Program APX

Program APXはScript APXとも呼ばれ、シェルコマンドに似ていて、OGFSサーバースクリプトとして実装されます。OGFSコマンドラインから呼び出し、入力引数はSTDINまたはコマンドライン引数を通じて渡します。出力はSTDOUTおよびSTDERRに行われます。

Program APXはGlobal Shell (OGSH) セッション内部で実行され、APXを呼び出したユーザーがアクセス権を持つすべてのOGSH機能にアクセスできます。これには、rosh、CLI、OGFSなどが含まれます。Program APXの作成には、シェルスクリプト、Python、Perlなど、任意のスクリプトベースのツールが使用できます。

Program APXはOGSHコマンドプロンプトから呼び出すことができます。Program APXは通常は同期的に実行されます。すなわち、Program APXが終了するまでシェルプロンプトは戻りません。APXは、twisterまたはOGFSの定期的ジョブとしてスケジュールすることはできません。

Program APXが存在する場所は、OGFSディレクトリ`/opsw/apx/bin`です。

対話型OGSHセッション中には、/opsw/apx/binに存在するProgram APXのうち、ユーザーが実行アクセス権を持つものだけがユーザーに見えます。ユーザーが実行アクセス権を持たないProgram APXを呼び出そうとすると、シェルから`File Not Found`エラーが返されます。

Program APXは、他のWeb APXまたはProgram APXから呼び出すこともできます。たとえば、Web APXのCGIプログラムまたはPHPスクリプトからProgram APXを呼び出すことができます。

Web APX

Web APXは、CGIプログラムまたはPHPスクリプトで実装されます。これらのCGIプログラムやPHPスクリプトは、ユーザー固有のOGSHセッション内部で実行されます。rosh、SA API、CLIなどのSA機能と、その他OGSHセッションから使用可能な任意のコマンドにアクセスできます。Web APXは、PHPモジュールを備えた付属のApache Webサーバーで実行されます。

Web APXにアクセスするには2つの方法があります。1つはInternet ExplorerやFirefoxなどのスタンドアロンのWebブラウザーからアクセスする方法、もう1つはSAクライアントからアクセスする方法です。Microsoft ActiveXはサポートされません。

スタンドアロンのWebブラウザーから初めてWeb APXを呼び出した場合、ログインダイアログが表示され、SAのユーザー資格情報を認証する必要があります。SAクライアントからWeb APXを呼び出した場合は、改めてログインする必要はありません。Web APXを使用すれば、カスタムカスタマーアプリケーションのユーザーインタフェースを構築できます。

Windows Server 2003、2008、2012でセキュリティ強化の構成がオンになっているときに、Microsoft Internet Explorerバージョン6および7からAPXを呼び出す場合は、Internet Explorerの信頼済みサイトのリストにSAクライアント URLをあらかじめ追加しておく必要があります。

APXユーザーの役割

APXユーザーには、次の表に示す3つの一般的な役割があります。

APXユーザーの役割

ユーザーの役割	説明
エンドユーザー	APXを実行します。このユーザーは通常、APXを変更したりその内容を見たりするアクセス権はありません。
APX開発者	APXを作成し、公開します。このクラスのユーザーは、APXをインポート/エクスポートしたり、APXの内容を変更したりできます。
APX管理者	ユーザーがどのAPXを実行できるかを決定します。これらのユーザーは、フォル

APXユーザーの役割 (続き)

ユーザーの役割	説明
	ダーのアクセス権を管理することにより、APXの実行アクセス権を割り当てます。APX管理者はAPX自体を変更するアクセス権は持たない場合がありますが、どのAPXを実行可能にするかを判断するためにAPX内容を表示するアクセス権を持つ可能性があります。

APXのアクセス権

APXを使用するには、SAクライアントの機能のアクセス権である**拡張の管理**が必要です。ユーザーグループには、次のいずれかのアクセス権を与えることができます。

- 拡張の管理: 読み取り
- 拡張の管理: 読み取り/書き込み
- 拡張の管理: なし

APX機能のアクセス権

これらの機能のアクセス権は、APX開発者および管理者のみに当てはまります。APXを実行するだけのユーザーには当てはまりません。

- **読み取り**: このアクセス権は、APXのソースの内容を表示したり、APXソースアーカイブをエクスポート(ダウンロード)したりする権限を付与します。
- **読み取り/書き込み**: このアクセス権は、APXの内容の読み取りに加えて変更の権限を付与します。
- **なし**: このアクセス権は、APXソースへのすべてのアクセスを禁止します。

SAクライアント機能の**拡張の管理**アクセス権に加えて、フォルダーのアクセス権 (リスト、読み取り、書き込み、実行)を使用して、ユーザーがどのAPXにアクセスできるかを決定する必要があります。

APXのアクセス権

アクセス権	説明
リスト	システムのAPXをリストするアクセス権。

APXのアクセス権 (続き)

アクセス権	説明
読み取り	APXの内容を表示するアクセス権。
書き込み	APXの内容の変更と、APXのインポート/エクスポートのアクセス権。
実行	APXの実行と、APXのプロパティの表示のアクセス権。

次の表のマトリクスは、拡張の管理機能のアクセス権とフォルダーのアクセス権の組み合わせに基づいて、アクセス権がどのように決定されるかを示します。

APXアクセス権のマトリクス

フォルダーのアクセス権:	拡張の管理機能のアクセス権:		
	読み取り	**読み取り/書き込み**	**なし**
リスト	APXのリスト	APXのリスト	APXのリスト
読み取り	APXのエクスポート	APXのエクスポート	APXのリスト
書き込み	APXのエクスポート	APXのインポート/エクスポート	APXのリスト
実行	APXの実行	APXの実行	APXの実行

他のSA機能と同様に、APXへのユーザーのアクセスを許可して、どの管理対象サーバーまたはポリシーにユーザーがAPXを適用できるかを指定できます。

ユーザーが実行アクセス権のないWeb APXにアクセスしようとすると、WebブラウザーはHTTP 403 Forbiddenリターンコードを受け取ります。

SAのアクセス権の詳細については、HPE SSOポータルで『Server Automation管理ガイド』を参照してください。

アクセス権のエスカレーション

APXを実行する際に、ユーザーはSAで許可されているリソースと操作だけにアクセスする権限があります。しかし、場合によっては、APXの実行中に、SAの権限を超えるエスカレートされたアクセス権をユーザーに一時的に付与することが必要になります。APXの実行中に、デフォルトのSAの権限を超える特定の権限を一時的にユーザーに付与することができます。アクセス権のエスカレーションは、APXを実行するユーザーに対しては透過的です。

たとえば、BIOS情報を収集するアプリケーションを管理対象サーバーに対して実行する場合に、ユーザーがそのためのアクセス権を持っていない可能性があります。BIOS情報収集アプリケーションの実行に

必要な権限を持たないユーザーのために、必要な権限をそのユーザーに一時的に付与するAPXを作成することができます。ユーザーの権限は、APXの実行が終了するとデフォルトに戻ります。

権限のエスカレーションは、apx.permファイルで指定されます。詳細については、「APXアクセス権エスカレーション構成ファイル - apx.perm」(120ページ)を参照してください。

APXの構造

APXは次の属性を持ちます。

- APXタイプ: Program APX (Script APXとも呼ばれる) またはWeb APX。
- APXの一意の名前: これはAPXの完全な名前で、一意である必要があります。たとえば、com.hpe.sa.RestartMyAppのようになります。
- APXの表示名: これは通常、APXの一意の名前よりも短い名前です。たとえば、RestartMyAppのようになります。
- APXバージョン: APXの複数のバージョンを維持するには、バージョン文字列を設定するか、SAに自動的にバージョンを管理させます。APXバージョンは、バージョン1、2、3などの単純な数字でも、任意の英数字文字列でもかまいません。

詳細については、「APXのSAへのインポート - apxtool import」(113ページ)および「APXの現在のバージョンの設定 - apxtool setcurrent」(116ページ)を参照してください。

ファイル構造

SAから見ると、APXは単にAPXタイプ (Program APXまたはWeb APX) の規約に従うファイルとディレクトリの集合です。これらの規約は、APXランタイムがAPXを正しく実行するために必要です。たとえば、Web APXにはindex.htmlファイルまたはindex.phpファイルが必要です。Program APXには、APXと同じ名前のシェルコマンドが必要です。

APXのファイルに関する詳細については、「APXファイル」(118ページ)を参照してください。

OGFS統合

APXインフラストラクチャーは、OGFSを利用して、ユーザーセッションを管理し、SAファイルシステム内のAPXのさまざまな部分を公開しています。ここでは、APXがOGFSとその各種アプリケーションにどのように統合されるかを説明します。

APX実行可能ディレクトリ

Program APXは、Global Shell (OGSH) の実行可能プログラムとして扱われます。これらのAPXは、OGSHの実行可能コマンドとして公開されます。これにより、シェルのユーザーはシェルコマンドと同じ方法でAPXを呼び出すことができます。

APXの実行可能ディレクトリは次の形式を取ります。

```
/opsw/apx/bin/{APX名 }
```

ここで、APX名 はAPXの名前です。/opsw/apx/bin/{APX名 }でAPX名 を実行すると、APX名 の現在のバージョンが呼び出されます。

APXランタイムディレクトリ

APXランタイムディレクトリは、APXの実行をサポートするためにAPXランタイムによって使用されます。APXランタイムディレクトリは、APXソースへのアクセス権を持つ必要があります。また、開発者権限を持ち、APXへの読み取りアクセス権を持つユーザーも、APXにアクセスすることができます。APXランタイムディレクトリは、Global ShellのAPX開発者以外からは使用できません。

APXランタイムディレクトリは、APXの現在のバージョンのソースを参照します。これは次の形式を取ります。

```
/opsw/apx/runtime/{APXタイプ}/{APX名 }
```

ここで、APXタイプはscriptまたはwebです。

APXインタフェース - APX拡張のカテゴリの定義

APXインタフェースを使用すると、APXの名前付きのカテゴリを作成し、特定のカテゴリに属するすべてのAPXを知ることができます。インタフェースとは、カテゴリの名前です。たとえば、特定の入力パラメーターの組を取り、特定のタイプの出力データを生成するAPXのカテゴリを作成することができます。あるいは、特定の操作の組を実行するAPXのカテゴリを作成することもできます。

さらに、特定のカテゴリのすべてのAPXの名前を取得して、それらを実行するAPXまたは外部アプリケーションを作成することもできます。あるいは、APXまたはアプリケーションは単に特定のカテゴリのAPXのリストを表示して、どれを実行するかをユーザーに選択させることもできます。

APXインタフェースとは、APXの呼び出し元とAPXの間の非公式の規約を定義する名前です。

- **インタフェース名を定義する**APXは、その名前のAPXカテゴリを作成します。
- **インタフェースを実装する**APXは、自分自身をそのカテゴリのAPXと宣言します。

インタフェースの例

SAには、RightClickToRunという名前のインタフェースが用意されています。このインタフェースが定義するカテゴリのAPXは、1つまたは複数のデバイスを入力パラメーターに取り、これらのデバイスに対して実行されます。さらに、SAクライアントはこのインタフェースを実装しているすべてのAPXを [**アクション**] > [**拡張の実行**] メニューに表示します。これにより、ユーザーは1つまたは複数のデバイスを選択して、これらのAPXを選択したデバイスに対して実行することができます。このインタフェースの詳細については、「RightClickToRunインタフェース」(107ページ)を参照してください。

インタフェースの定義

APXインタフェースは、APXのカテゴリの名前を定義します。インタフェースを実装するすべてのAPXはそのカテゴリに属し、そのインタフェースの規則に従う必要があります。新しいカテゴリを作成するには、APXにインタフェースを「定義」させます。

APXにインタフェースを定義させるには、次の手順を実行します。

1. `apxtool new`コマンドでAPXを作成します。このコマンドの詳細については、「新しいAPXの作成 - apxtool new」(110ページ)を参照してください。
2. 新しいAPXのファイルを探し、`interfaces`という名前のファイルをテキストエディターで開きます。interfacesファイルは、APXディレクトリ下のAPX-INFディレクトリにあります。
3. `interfaces`ファイルの末尾に、次の3行を追加します。
   - ファイル内のインタフェースセクションの名前。これはインタフェースの一意の名前です。
   - インタフェースの表示名。
   - インタフェースの説明。

   次の例は、com.hpe.sa.MyNewInterfaceという名前のインタフェースのインタフェースセクション名、表示名、説明を示します。

   ```
   [com.hpe.sa.MyNewInterface]
   name=MyNewInterface
   description=”This is a simple interface for testing purposes.”
   ```
4. 変更を保存し、ファイルを閉じます。
5. 変更したAPXを`apxtool import`コマンドでSAにインポートします。このコマンドの詳細については、「APXのSAへのインポート - apxtool import」(113ページ)を参照してください。

既存のAPXをインタフェースを定義するようにアップグレードする場合は、`interfaces`ファイルを作成し、上記の手順でインタフェースを追加する必要があります。

インタフェースの実装

APXインタフェースは、インタフェースの規則に従うAPXのカテゴリを指定します。APXがカテゴリに属することを指定するには、APXにインタフェースを「実装」させます。APXにインタフェースを実装させるには、次の手順を実行します。

1. `apxtool new`コマンドでAPXを作成します。このコマンドの詳細については、「新しいAPXの作成 - apxtool new」(110ページ)を参照してください。
2. 新しいAPXのファイルを探し、`apx.cfg`という名前のファイルをテキストエディターで開きます。
3. `apx.cfg`ファイルでImplementingセクションを探します。このセクションには、APXが実装するインタフェースを指定する方法が簡単に説明されています。
4. `apx.cfg`ファイルで次の行を見つけます。

   ```
   [Implementing]
   interfaces=
   ```

5. `interfaces=`行を変更して、インタフェースの名前を行の末尾に追加します。たとえば、APXがcom.hpe.sa.MyNewInterfaceという名前のインタフェースを実装する場合は、`apx.cfg`ファイルに次の行が含まれるようにします。

   ```
   [Implementing]
   interfaces=com.hpe.sa.MyNewInterface
   ```

   複数のインタフェースを実装する場合、次のようにコロンで区切ってinterfaces行に追加します。

   ```
   [Implementing]
   interfaces=com.hpe.sa.MyNewInterface:com.hpe.sa.AnotherInterface
   ```

6. 変更を保存し、`apx.cfg`ファイルを閉じます。
7. 変更したAPXを`apxtool import`コマンドでSAにインポートします。このコマンドの詳細については、「APXのSAへのインポート - apxtool import」(113ページ)を参照してください。

SAクライアントまたは`apxtool query`コマンドでAPXを表示する際に、実装されたインタフェースを見るには、APXの現在のバージョンを設定する必要があります。詳細については、「APXの現在のバージョンの設定 - apxtool setcurrent」(116ページ)を参照してください。

既存のAPXをインタフェースを使用するようにアップグレードする場合は、既存の`apx.cfg`ファイルに上記の手順でインタフェースを追加する必要があります。

RightClickToRunインタフェース

SAには、APXで使用できる`com.hpe.client.server.RightClickToRun`という名前のインタフェースが用意されています。このインタフェースが使用できるのはProgram APXだけで、Web APXでは使用できません。このインタフェースは、APXに次のことを実行させたい場合に使用します。

- APXへの入力パラメーターとして1つまたは複数のデバイスを取ります。このインタフェースを実装するAPXは、入力引数として`-d <デバイスID>`を取る必要があります。
- **[アクション] > [拡張の実行] >[拡張の選択...]** ウィンドウに表示されます。
- SAクライアントの **[アクション] > [拡張の実行]** メニューに表示されます。APXがこのメニューに表示されるのは、**[アクション] > [拡張の実行] > [拡張の選択...]** メニューを使用して1回実行された後です。

[アクション] > [拡張の実行] メニューからAPXを実行するには、ユーザーはAPXに対する実行アクセス権を持つ必要があります。ユーザーが実行アクセス権を持たないAPXは、このメニュー項目に表示されません。アクセス権については、HPE SSOポータルで『Server Automation管理ガイド』を参照してください。

RightClickToRunインタフェースにより、ユーザーはSAクライアントで1つまたは複数のデバイスを選択して、これらのデバイスに対してAPXを実行できます。

[アクション] > [拡張の実行] メニュー項目を選択すると、SAクライアントはインタフェースcom.hpe.client.server.RightClickToRunを実装しているすべてのProgram APXを表示します。ユーザーがAPXを選択すると、選択したすべてのサーバーに対してAPXが実行されます。APXは選択したそれぞれのサーバーに対して1回ずつ呼び出されます。

APXにこのインタフェースを実装させる方法の詳細については、「インタフェースの実装」(106ページ)を参照してください。このインタフェースを実装しているAPXの使用に関する詳細については、HPE SSOポータルで『Server Automationユーザーガイド』の「SAの拡張の実行」を参照してください。

CoreAffinityインタフェース

SAには、APXで使用できるcom.hpe.client.server.CoreAffinityという名前のインタフェースが用意されています。このインタフェースは、CoreAffinityモードでAPXを実行する場合に使用できます。

CoreAffinityモードが適用されるのは、少なくとも2つのSAコアを含むメッシュがある場合のみです。各ターゲットサーバーでこのモードが有効になっている場合、実際のジョブがどこで開始されたかに関係なく、APXはこのターゲットサーバーが登録されているSAコアで実行されます。

例:

- 2つのコア (コアAおよびコアB) を含むメッシュがある
- 2つのターゲットサーバーMA (コアAに登録) およびMB (コアBに登録) 上でコアAからAPXジョブを開始する

CoreAffinityモードの場合、このジョブはコアAでMAに対するAPXを実行し、コアBでMBに対するAPXを実行します。CoreAffinityが無効な場合は、どちらもコアAで実行されます (コアAでジョブが開始されたため)。

APXにCoreAffinityインタフェースを実装させる方法の詳細については、「インタフェースの実装」(106ページ)を参照してください。

インタフェースAPIの使用

SA APIを使用して、自作のアプリケーションをSAおよびAPXと統合できます。特定のインタフェースを実装しているすべてのAPXをアプリケーションから知るには、SA APIのcom.opsware.apxというパッケージにあるAPXInterfaceServiceというインタフェースを使用します。SA APIの使用法の詳細については、「APIドキュメントとTwister」(37ページ)を参照してください。

apxtoolコマンド

apxtoolコマンドをOGFSセッションで使用すると、APXの作成と管理を行うことができます。apxtoolコマンドは、Global Shellのディレクトリ`/opsw/bin/apxtool`で使用できます。

apxtoolを使用してWeb APXを作成する方法のチュートリアルについては、「チュートリアル: WebアプリケーションAPXの作成」(123ページ)を参照してください。

apxtoolの構文

APXツールを呼び出すには、OGFSコマンドラインに次のように入力します。

```
apxtool [-h | --help] {機能} arguments
```

APXツールでサポートされるコマンドと引数の一覧を見るには、OGSHコマンドラインからapxtoolを引数なしで実行します。

APXツールがサポートする主な機能を次に示します。

APXツールの機能

機能	使用法
new	新しいAPXソースディレクトリと、新しいテンプレートファイルのセットをOGFSに作成します。オプションで、APXをSAに登録します。登録すると、APX IDが割り当てられ、SAを使用している(適切なアクセス権を持つ)他のユーザーからAPXの名前が使用できるようになります。詳細については、「新しいAPXの作成 - apxtool new」(110ページ)を参照してください。
import	APXファイルをSAライブラリにインポートし、APXの新しいバージョンを作成します。オプションで、APXをSAに登録します。登録すると、APX IDが割り当てられ、SAを使用している(適切なアクセス権を持つ)他のユーザーからAPXの名前が使用できるようになります。詳細については、「APXのSAへのインポート - apxtool import」(113ページ)を参照してください。
setcurrent	SAライブラリ内のAPXの現在のバージョンを設定します。SAではAPXの複数のバージョンを持つことができますが、実行できるのは現在のバージョンだけです。詳細については、「APXの現在のバージョンの設定 - apxtool setcurrent」(116ページ)を参照してください。
query	APXに関する情報を表示します。詳細については、「APX情報のクエリ - apxtool query」(114ページ)を参照してください。
export	APXのすべてのファイルをSAライブラリから別のファイルセットにコピーします。
delete	APXをSAライブラリから削除します。

短いコマンドオプションと長いコマンドオプションの使用

apxtoolコマンドのオプションのほとんどには、短い形式と長い形式があります。

- 短い形式は1個のハイフンと1文字からなります。例としては、“-t“ や“-v” があります。
- 長い形式は、2個のハイフンと1語からなります。例としては、“--type“ や“--view“ があります。

一部のオプションでは、オプションの後に引数が必要です。例としては、“-t webapp“ や“-t details“ があります。引数の指定には4つの形式があり、どれも効果は同じです。たとえば、次のコマンドはすべて等価であり、同じ結果になります。

```
apxtool query -t webapp
apxtool query -twebapp
apxtool query -tw
apxtool query --type webapp
apxtool query --type=webapp
```

一部のオプションでは、オプション引数の識別に必要な最小限の文字を入力するだけで済みます。たとえば、query機能の場合、--viewオプションには引数 “list“、“details“、“versions“ が必要です。次のコマンドは同じ結果になります。

```
apxtool query --view=details
apxtool query --view=d
apxtool query -vdetails
apxtool query -vd
```

新しいAPXの作成 - apxtool new

APXツールを使用して、新しいAPXを作成し、オプションでAPXの名前をSAに登録することができます。このコマンドはAPXテンプレートファイルのセットを作成します。これらのファイルは変更できます。APXを構成するファイルに関する詳細については、「APXファイル」(118ページ)を参照してください。

使用法

```
apxtool new [オプション] {ソースディレクトリ}
```

ここで、ソースディレクトリ引数は、新しいAPXのテンプレートファイルが作成されるディレクトリを指定します。この引数を省略した場合、テンプレートファイルは現在のディレクトリに作成されます。

次の表に、新しいAPXを作成するためのオプションの一覧を示します。

apxtool newのオプション

オプション	使用法
-h、--help	このヘルプメッセージを表示して終了します。
-t <タイプ> --type=<タイプ>	(**必須**) APXタイプ。有効な値はscriptまたはwebappです。たとえば、-tsはScript APX、-twはWeb APXを表します (Script APXはProgram APXとも呼ばれます)。
-u <一意の名前> --uniquename=<一意の名前>	(**必須**) APXの一意の名前。一意の名前は、ファイルシステムの形式に従ったドット区切りの名前です。ドットが少なくとも1つ含まれる必要があります。使用可能な文字は[a-zA-Z0-9_.] です。 **例**: com.hpe.sa.security.scan_ports
-n <名前>	(**オプション**) フォルダー内のAPXの表示名。名前を指定せず、一意の名前

apxtool newのオプション (続き)

オプション	使用法
`--name=<名前>`	を指定した場合は、APXの一意の名前の最後の部分が表示名として使用されます。この名前は指定したフォルダー内で一意である必要があります。 たとえば、一意の名前がcom.hpe.sa.MyWebExtの場合、デフォルトの表示名はMyWebExtとなります。
`-d <説明>` `--description=<説明>`	(**必須**) APXの簡単な説明。説明が拡張子.txtのファイル名の場合、そのファイルはテキストファイルと見なされ、その内容がAPXの説明として用いられます。
`-r` `--register`	(**オプション**) APXの名前をシステムに登録します。このオプションを指定した場合、`-f`または`--folder`も指定する必要があります。 `apxtool new`に`-r`と`-f`を指定しなかった場合は、`apxtool import`で`-f`を使用する必要があります。
`-f <パス>` `--folder=<パス>`	(**オプション**) APXが登録されるSAフォルダーのパス。これは、特定のフォルダーを一意に識別するものであれば、完全パス、部分パス、絶対パス、相対パスのどれでもかまいません。このオプションが必要なのは、`-r`または`--register`を使用した場合だけです。 `apxtool new`に`-r`と`-f`を指定しなかった場合は、`apxtool import`で`-f`を使用する必要があります。
`-Q、--quiet`	(**オプション**) 出力メッセージを抑制します。
`-F、--force`	(**オプション**) 確認プロンプトを抑制します。

APXの削除 - apxtool delete

APXツールを使用して、既存のAPXをSAライブラリから削除できます。

使用法

`apxtool delete [オプション]`

次の表に、APXを削除するためのオプションの一覧を示します。

apxtool deleteのオプション

オプション	使用法
-h --help	このヘルプメッセージを表示して終了します。
-t <タイプ> --type=<タイプ>	(**必須**) APXタイプ。有効な値はscriptまたはwebappです。たとえば、-tsはスクリプトを表します。
--id=<APX ID>	(**オプション**) 目的のAPXのオブジェクトID。
-u <一意の名前> --uniquename=<一意の名前>	(**オプション**) APXの一意の名前。一意の名前は、ファイルシステムの形式に従ったドット区切りの名前です。ドットが少なくとも1つ含まれる必要があります。使用可能な文字は[a-zA-Z0-9_.]です。 **例**: com.hpe.sa.security.scan_ports
-n <名前>、--name=<名前>	(**オプション**) フォルダー内のAPXの表示名。
-f <パス>、--folder=<パス>	(**オプション**) SAフォルダーのパス。これは、特定のフォルダーを一意に識別するものであれば、完全パス、部分パス、絶対パス、相対パスのどれでもかまいません。
-Q、--quiet	(**オプション**) 出力メッセージを抑制します。
-F、--force	(**オプション**) 確認プロンプトを抑制します。

SAからのAPXのエクスポート - apxtool export

APXツールを使用して、APXをエクスポートできます。エクスポートとは、APXソースアーカイブファイルの特定のバージョンをダウンロードして、ファイルをディレクトリまたは.zipアーカイブファイルに格納することです。

使用法

apxtool export [**オプション**] {**ターゲットディレクトリ**}

ここで、引数ターゲットディレクトリはディレクトリであり、--archiveオプションが指定されているかどうかに応じて、APXソースアーカイブファイルがコピーされるか、APXソースアーカイブの内容が展開されるディレクトリを表します。省略した場合、現在のディレクトリが使用されます。

次の表に、APXをエクスポートするためのオプションの一覧を示します。

apxtool exportのオプション

オプション	使用法
-h、--help	このヘルプメッセージを表示して終了します。
-t <タイプ>、--type=<タイプ>	(**必須**) APXタイプ。有効な値はscriptまたはwebappです。たとえば、-tsはスクリプトを表します。
--id=<APX ID>	(**オプション**) 目的のAPXのオブジェクトID。
-u <一意の名前> --uniquename=<一意の名前>	(**オプション**) APXの一意の名前。一意の名前は、ファイルシステムの形式に従ったドット区切りの名前です。ドットが少なくとも1つ含まれる必要があります。使用可能な文字は [a-zA-Z0-9_.] です。 **例**: com.hpe.sa.security.scan_ports
-n <名前>、--name=<名前>	(**オプション**) フォルダー内のAPXの表示名。
-f <パス>、--folder=<パス>	(**オプション**) SAフォルダーのパス。これは、特定のフォルダーを一意に識別するものであれば、完全パス、部分パス、絶対パス、相対パスのどれでもかまいません。
-v <バージョン文字列>、--version=<バージョン文字列>	(**オプション**) このオプションは、ダウンロードするAPXバージョンを指定します。省略した場合、現在のバージョンがダウンロードされます。
-a、--archive	指定した場合、元のソースアーカイブのAPXソースをZIPまたはJARファイルでエクスポートします。
-Q、--quiet	(**オプション**) 出力メッセージを抑制します。
-F、--force	(**オプション**) 確認プロンプトを抑制します。

APXのSAへのインポート - apxtool import

APXツールを使用して、APXをインポートできます。インポートとは、APXの新しいバージョンを公開し、オプションでそのバージョンを現在のバージョンに設定することです。APXがまだ登録されていない場合は、このコマンドはAPXの登録も行います。

実行できるのは現在のバージョンのAPXだけです。現在のバージョンを選択していない場合、APXは実行できません。現在のバージョンを設定するには、apxtool importまたはapxtool setcurrentを使用

します。詳細については、「APXの現在のバージョンの設定 - apxtool setcurrent」(116ページ)を参照してください。

使用法

```
apxtool import [オプション] {APXソース}
```

ここで、APXソースは拡張子.zipまたは.jarのAPXソースアーカイブか、公開するAPXファイルを含むディレクトリです。APXソースは相対パスでも絶対パスでもかまいません。省略した場合、現在のディレクトリが使用されます。指定したディレクトリまたはアーカイブファイルには、ディレクトリAPX-INFが含まれる必要があります。

次の表に、APXをインポートする際に使用できるオプションの一覧を示します。

apxtool importのオプション

オプション	使用法
-h、--help	このヘルプメッセージを表示して終了します。
-c、--setcurrent	指定した場合、新しく公開したバージョンをAPXの現在のバージョンに設定します。
--version=<バージョン文字列>	このAPXの新しいバージョン。バージョン文字列がapx.cfgにすでに指定されている場合は、このオプションは使用できません。バージョンを指定しない場合、自動的にバージョンが割り当てられます。
-f <パス>、--folder=<パス>	(**オプション**) SAフォルダーのパス。これは、特定のフォルダーを一意に識別するものであれば、完全パス、部分パス、絶対パス、相対パスのどれでもかまいません。 apxtool newに-rと-fを指定しなかった場合は、apxtool importで-rを使用する必要があります。
-Q、--quiet	(**オプション**) 出力メッセージを抑制します。
-F、--force	(**オプション**) 確認プロンプトを抑制します。

APX情報のクエリ - apxtool query

APXツールを使用して、APX情報を取得して表示できます。追加のオプションを指定することで、結果のAPXを制限できます。同じオプションを複数回指定した場合、論理OR式で結合されます。一致する結果が見つからない場合、このコマンドは終了コード100を返します。

使用法

apxtool query [オプション]

次の表に、APX情報のクエリの際に使用できるオプションの一覧を示します。

apxtool queryのオプション

オプション	使用法
-h、--help	このヘルプメッセージを表示して終了します。
-v <表示>, --view=<表示>	(**オプション**) クエリ結果の事前定義された表示の1つを選択します。選択肢としては、list (デフォルト)、details、versionsがあります。 -v listは、APXの基本情報を1行で表したものを表形式で表示します。 -v detailsは、APXの基本情報を複数行で表したものです。 -v versionsは、APXのすべてのバージョンをリストします。表示タイプの指定には、十分な数の文字だけを指定すれば済みます。たとえば、-vdは-v detailsと同じです。レイアウトとして versionsを選択した場合、クエリの結果は1個のAPXオブジェクトでなければなりません。
-t <タイプ>、--type=<タイプ>	(**オプション**) 表示するAPXのタイプを指定します。有効な値はscriptまたはwebappまたはinterfaceです。デフォルトでは、すべてのタイプが表示されます。 -t scriptはすべてのScript APXを表示します。 -t webappはすべてのWeb APXを表示します。 -t interfaceは1つまたは複数のインタフェースを定義するすべてのAPXを表示します。 たとえば、apxtool query -tsはすべてのScript APXを表示します。
--id=<APX ID>	(**オプション**) 目的のAPXのオブジェクトID。
-u <一意の名前> --uniquename=<一意の名前>	(**オプション**) APXの一意の名前。一意の名前は、ファイルシステムの形式に従ったドット区切りの名前です。ドットが少なくとも1つ含まれる必要があります。使用可能な文字は[a-zA-Z0-9_.] です。 **例**: com.hpe.sa.security.scan_ports
-n <名前>、--name=<名前>	(**オプション**) フォルダー内のAPXの表示名。
-f <パス>、--folder=<パス>	(**オプション**) SAフォルダーのパス。これは、特定のフォルダーを一意に識別するものであれば、完全パス、部分パス、絶対パス、相対パスのどれでもかまいま

apxtool queryのオプション (続き)

オプション	使用法
	せん。
`--current`	(**オプション**) 指定した場合、現在のバージョンが設定されているAPXオブジェクトだけをクエリの対象とします。
`--format=<フォーマット文字列>`	(**オプション**) これは高度なオプションであり、APXリストのカスタム表示形式を指定します。 フォーマット文字列は文字列で、中に埋め込まれたタグ名が表示時に値に置き換えられます。タグ名の形式は`%`(タグ名)です。 サポートされるタグ名の一覧を表示するには、フォーマット文字列として "`__show_tags__`" を使用します。
`--csv`	(**オプション**) 出力をカンマ区切り値形式で表示します。`--format`オプションを指定した場合は無視されます。
`-Q、--quiet`	(**オプション**) 重要でない出力メッセージを抑制します。

APXの現在のバージョンの設定 - apxtool setcurrent

APXツールを使用して、APXバージョンを現在のバージョンとして設定できます。

実行できるのは現在のバージョンのAPXだけです。現在のバージョンを選択していない場合、APXは実行できません。現在のバージョンを設定するには、`apxtool import`または`apxtool setcurrent`を使用します。詳細については、「APXのSAへのインポート - apxtool import」(113ページ)を参照してください。

使用法

`apxtool setcurrent [オプション] {バージョン文字列}`

ここで、引数バージョン文字列は、APXの既存のバージョンを一意に識別するために必要です。

次の表に、APXバージョンを設定する際に使用できるオプションの一覧を示します。

apxtool setcurrentのオプション

オプション	使用法
-h、--help	このヘルプメッセージを表示して終了します。
-t <タイプ>、--type=<タイプ>	(**必須**) APXタイプ。有効な値はscript、webappです。たとえば、-tsはスクリプトを表します。
--id=<APX ID>	(**オプション**) 目的のAPXのオブジェクトID。
-u <一意の名前> --uniquename=<一意の名前>	(**オプション**) APXの一意の名前。一意の名前は、ファイルシステムの形式に従ったドット区切りの名前です。ドットが少なくとも1つ含まれる必要があります。使用可能な文字は [a-zA-Z0-9_.] です。 **例**: com.hpe.sa.security.scan_ports
-n <名前>、--name=<名前>	(**オプション**) フォルダー内のAPXの表示名。
-f <パス>、--folder=<パス>	(**オプション**) SAフォルダーのパス。これは、特定のフォルダーを一意に識別するものであれば、完全パス、部分パス、絶対パス、相対パスのどれでもかまいません。
-Q、--quiet	(**オプション**) 出力メッセージを抑制します。
-F、--force	(**オプション**) 確認プロンプトを抑制します。

エラー処理

APXツールコマンドは標準のPOSIX規則に準拠しており、成功の場合は0、他のエラーの場合は0以外の値を返します。APXツールは、通常の出力をSTDOUTに、エラーと警告をSTDERRに送信します。エラーが発生した場合、APXツールは通常STDERRに説明のメッセージを出力します。

エラー条件は通常、次の表に示すように分類されます。

APXツールのエラー条件

リターンコード	説明
0	成功
1	構文または使用法のエラー
2	アクセス権関連のエラー
3	ユーザーが操作をキャンセル
4	実行時エラー

文書化されていない他の終了コードが存在する可能性もあります。保証されるのは、終了コードが0の場合、コマンドは操作を正常に完了したということだけです。

APXファイル

ここでは、apxtool newコマンドを実行したときに作成されるテンプレートファイルについて説明します。次の表にファイルの一覧を示します。この後で、いくつかのファイルについて詳細に説明します。

APXファイル

ファイル名	説明
apx.cfg	APX構成ファイル。APXを完全に記述するメタデータを含みます。詳細については、「APX構成ファイル - apx.cfg」(118ページ)を参照してください。
apx.perm	APXアクセス権ファイル。アクセス権のエスカレーションルールを指定します。詳細については、「APXアクセス権エスカレーション構成ファイル - apx.perm」(120ページ)を参照してください。
description.txt	APXの説明テキスト。apxtool new -dオプションで指定されます。詳細については、「新しいAPXの作成 - apxtool new」(110ページ)を参照してください。
interfaces	APXインタフェース定義ファイル。APXが定義または実装するインタフェースを指定します。詳細については、「APXインタフェース - APX拡張のカテゴリの定義」(104ページ)を参照してください。
usage.txt	APXの使用法に関する説明テキスト。
run.sh	Program APXの場合のみ、このファイルにはAPXの実行可能コードが含まれます。このファイルには、Program APXの機能が記述されています。例については、「チュートリアル: Program APXの作成」(130ページ)を参照してください。
index.php	Web APXの場合のみ、このファイルにはWeb APXのPHPソースコードが含まれます。このファイルには、Web APXの機能が記述されています。例については、「チュートリアル: WebアプリケーションAPXの作成」(123ページ)を参照してください。

APX構成ファイル - apx.cfg

構成ファイルapx.cfgは、タイプに関わらずすべてのAPXに存在します。apxtool newコマンドがこのファイルのテンプレートを作成し、ユーザーはそれを変更することができます。このファイルには、APXを完全に記述するメタデータが含まれます。apx.cfgは “キー=値“ 形式を使用してAPXのプロパティを定義します。複数の行は、行継続文字 "\" によって結合されます。

「APX構成ファイルの属性」(119ページ)の表に、すべてのAPXに共通の属性が記されています。APXタイプに固有の属性については、対応するAPXタイプの機能仕様に記述されています。いくつかの属性は、apx.cfg構成ファイルから抽出して、SAで管理することができます。説明などの変更可能な属性については、apx.cfgファイルを更新すると、SAで管理されているデータもそれに基づいて更新されます。

apx.cfgファイルの例を見るには、apxtool newコマンドを実行して、作成されたファイルを開いてください。

APX構成ファイルの属性

属性	変更	説明
タイプ	不可	APXのタイプ。webappまたはscriptのどちらか(Script APXはProgram APXとも呼ばれます)。作成後にAPXタイプを変更することはできません。
name	可能	これはAPXの表示名であり、マルチバイト文字が含まれることがあります。この名前はいつでも変更できます。この名前はSAクライアントのAPXフォルダーにリストされます。
unique_name	不可	APXの一意の名前。この名前は、OGFSでAPXのファイル名として使用されます。この名前とタイプの組み合わせは、APXを一意に識別するキーの役割を果たします。作成後に名前を変更することはできません。この名前はファイルシステムで用いられるので、ファイルシステムの名前付け仕様に適合する必要があります。一般的には名前にはASCII文字だけを使用します。
version	可能	APXの現在のバージョンを表すバージョン文字列。値が文字列"auto":で始まる場合、SAがバージョンを自動的に管理し、新しいバージョンごとに1ずつ増加する整数を使用します。
description	可能	APXの機能の説明テキスト。この属性の代わりにdescription.txtファイルを使用することもできます。
usage	可能	APXの使用法に関する説明テキスト。この属性の代わりにusage.txtファイルを使用することもできます。
interfaces	可能	APXが実装する1つまたは複数のインタフェース。複数のインタフェースはコロン (:) 文字で区切ります。
command	可能	APXを呼び出したときに実行される実行可能ファイル。

APXアクセス権エスカレーション構成ファイル - apx.perm

apx.permファイルは、アクセス権のエスカレーションルールを指定するために使用します。このファイルが存在しないか、エスカレーションアクセス権が記述されていない場合、APXはユーザーのデフォルトのアクセス権で動作します。

APX Toolの**New**コマンドで新しいAPXを作成すると、デフォルトのファイル群が生成され、その中にデフォルトのapx.permファイルがありますが、これにはエスカレーションアクセス権は定義されていません。デフォルトのファイルにはいくつかの例がコメントアウトされた形で記述されているので、APX開発者はそれをテンプレートとして利用できます。

エスカレーションを指定する3つの方法を次に示します。

- 「エスカレーションなし」(120ページ)
- 「すべてのアクセス権」(120ページ)
- 「エスカレーションあり」(121ページ)

エスカレーションなし

エスカレーション属性は指定されません。APXランタイムは、現在のユーザー権限を使用してAPXを実行します。ユーザーが持たない権限を必要とする操作をAPXが呼び出した場合、APXの実行はエラーによって終了します。

すべてのアクセス権

これは特殊な権限で、一時的にすべての操作へのアクセス権をユーザーに付与します。これは、開発またはデモ用にのみ用意されています。アクセス権を細かく調整する手間をかけずに簡単に概念実証やデモを行うために便利です。ただし、セキュリティが無効になるため、本番環境には適しません。

すべての権限を付与するには、apx.permファイルを編集して、すべての機能に一致するワイルドカード文字を含むマクロを記述します。以下に例を挙げます。

```
use_feature(name="*")
```

エスカレーションあり

事前定義された共通操作のリストをapx.permファイルに指定します。APXを実行する際に、APXランタイムは一時的にこれらのアクセス権をAPXに付与します。SAには、機能とリソースへのアクセス権の包括的なリストが用意されています。関連する機能のエスカレーションを容易にするため、ワイルドカード文字を使用して関連機能のグループに一致させることができます。以下に例を挙げます。

```
@use_feature(name="Application.*")
```

APXの進行状況の表示

apxprogressコマンドをProgram APXで使用すると、APXの進行状況に関する情報を提供することができます。これは、実行に時間がかかるProgram APXで、APXの進行状況をユーザーに通知したい場合に使用できます。

Web APXをProgram APXのフロントエンドとして使用して、Web APXの進行状況を表示することもできます。

apxprogressコマンド

apxprogressコマンドを使用すると、Program APXの実行ステップ数を定義し、各ステップの完了を記録することができます。これにより、APXのユーザーは、APXの進行状況と、残りがどれくらいかを知ることができます。

apxprogressの構文

```
apxprogress {オプション}..
```

apxprogressコマンドのオプション

オプション	説明
-i <総ステップ数>	APXの実行にかかる総ステップ数を指定します。このオプションは、APXの最初で1回使用して、APXの実行にかかる総ステップ数を指定します。 このオプションをAPX内で複数回使用すると、ステップ数を増やすことができま

apxprogressコマンドのオプション (続き)

オプション	説明
	す。使用するたびに、総ステップ数が指定した値だけ増加します。
-c <現在のステップ>	現在のステップ番号を指定します。APXコードの各ステップの終了時に、このオプションを指定してapxprogressを呼び出します。
-m <メッセージ>	APXのステータスを示すテキストメッセージを指定します。
-a <データ>	APXが自分自身に関して提供する追加情報を指定します。
-d	デバッグモードを示します。コマンドの出力をデバッグ用にstdoutに表示します。
-h	apxprogressコマンドに関するヘルプ情報を表示します。

apxprogressを使用するサンプルシェルスクリプト

次のシェルスクリプトは、apxprogressコマンドを使用するProgram APXの一部です。このAPXは、総ステップ数として100を定義し、現在の進行状況を100回報告します。また、報告のたびにステップ番号を示すメッセージも出力します。

```
#!/bin/sh
#########################################################################
# A simple shell script for a program APX that displays progress
# about itself.
#Author: <name>
#########################################################################
echo "This is a simple APX that uses apxprogress."
totalsteps=100
apxprogress -i $totalsteps -c 1
for i in `seq $totalsteps`; do
apxprogress -c $i -m "APX is running, working on step $i" -d
sleep 10
done
```

APXの進行状況の表示

SAのAPIメソッドJobService.getProgress()を使用して、apxprogressコマンドを呼び出す実行中のAPXの進行状況の情報にアクセスできます。このメソッドの使用例については、「チュートリアル: Program APXの作成」(130ページ)の中の「TwisterインタフェースへのAPXの進行状況の表示」(137ページ)を参照してください。

チュートリアル: WebアプリケーションAPXの作成

このチュートリアルは、mywebappという名前の単純なWebアプリケーションAPXを作成し、公開し、実行する方法を示します。

このチュートリアルで作成するAPXのデフォルトバージョンを実行すると、PHPコマンドphpinfoの出力が表示されます。チュートリアルの後の方で、PHPコードを変更して、管理対象サーバーのリストを表示する方法を示します。チュートリアルにはソースコードが付属しているので、PHPに関する事前の知識は必要ありません。

次の作業を順番に実行してください。

1. 「アクセス権の設定とチュートリアルフォルダーの作成」(124ページ)
2. 「新規Webアプリケーションの作成」(125ページ)
3. 「新規WebアプリケーションのSAへのインポート」(127ページ)
4. 「新規Webアプリケーションの作成」(127ページ)
5. 「Webアプリケーションの変更」(128ページ)
6. 「変更したWebアプリケーションの実行」(130ページ)

チュートリアルの前提条件

このチュートリアルを実行するには、次の機能と環境が必要です。

- adminまたは**スーパー管理者**グループの他のメンバーとしてSAにログオンできること。adminでログオンすることで、アクセス権を設定できるようになります。
- **上級ユーザー**グループに属するユーザーとしてSAにログオンできること。
- 上級ユーザーには、Webアプリケーションを作成して実行するアクセス権があります。このチュートリアルに示すコマンドの例では、このユーザーの名前はjdoeになっています。
- SAクライアントでクライアント機能のアクセス権を設定する方法に関する理解。
- アクセス権の詳細については、『SA 10.50管理ガイド』の「ユーザーおよびユーザーグループの設定とセキュリティ」の項を参照してください。
- SAクライアントでフォルダーを作成する方法に関する理解

- フォルダーの詳細については、『SA 10.50ユーザーガイド』を参照してください。
- Global Shellセッションを開く方法に関する理解。
- `ls`や`cd`などの基本的なUnixコマンドに関する理解。
- HTTPサーバーで動作するWebアプリケーションの開発経験。

アクセス権の設定とチュートリアルフォルダーの作成

1. SAクライアントに**上級ユーザー**グループのメンバーでログオンし、SAライブラリに次のフォルダーを作成します。

   ```
   /Dev/MyApp
   ```

   チュートリアルの後の方で、MyAppフォルダーにWebアプリケーションをアップロードします。チュートリアル以外の環境では、このフォルダーの名前は任意です。Webアプリケーションを置くフォルダーは任意に作成または選択してかまいません。
2. SAクライアントを終了します。
3. SAクライアントに`admin`でログオンし、`MyApp`フォルダーの [**フォルダーのプロパティ**] を開きます。
4. [**フォルダーのプロパティ**] の [**アクセス権**] タブで、**上級ユーザー**グループに次のアクセス権があることを確認します。
   - フォルダーの内容のリスト表示
   - フォルダー内のオブジェクトの読み取り
   - フォルダー内のオブジェクトの書き込み
   - フォルダー内のオブジェクトの実行
5. SAクライアントを終了します。

新規Webアプリケーションの作成

1. **上級ユーザー**グループに属するSAユーザーとしてGlobal Shellセッションを開きます。
2. コアのOGFSホームディレクトリに`mywebapp`という名前のディレクトリを作成し、そのディレクトリに移動します。

   ```
   $ mkdir mywebapp
   $ cd mywebapp
   ```

   Webアプリケーションのファイルは、`mywebapp`ディレクトリに格納されます。
3. 次に示すように、`apxtool new`コマンドを使用して、Webアプリケーションのディレクトリ構造とデフォルトのファイルを作成します。

   ```
   $ pwd
   /home/jdoe/mywebapp
   $ ls
   $
   $ apxtool new -tw -d "This is my first app."\
   -u com.hpe.sa.jdoe.mywebapp
   Create source directory /home/jdoe/mywebapp/com.hpe.sa.jdoe.mywebapp? Y/N y
   Info: Successfully created APX 'mywebapp' source directory:
   /home/jdoe/mywebapp.
   ```

   `-tw`オプションはAPXタイプがWebアプリケーションであることを示し、`-d`は説明を指定し、`-u`はアプリケーションの一意の名前を指定します。

   `apxtool new`コマンドのオプションの詳細については、次のコマンドでオンラインヘルプを参照してください。

   ```
   $ apxtool new -h
   ```
4. `apxtool new`コマンドで作成した新しいディレクトリに移動し、そこにあるファイルのリストを表示します。

   ```
   $ pwd
   /home/jdoe/mywebapp
   $ cd com.hpe.sa.jdoe.mywebapp
   $ ls
   ```

```
APX-INF cgi-bin css images index.php
$ ls -R
.:
APX-INF cgi-bin css images index.php
./APX-INF:
apx.cfg apx.perm description.txt interfaces usage.txt
./cgi-bin:
./css:
hp_sa.css
./images:
```

5. デフォルトのindex.phpファイルの内容を表示します。

```
$ cat index.php
<?php
// Show information about PHP
phpinfo();
?>
```

他のWebアプリケーションと同様、index.phpファイルはindex.htmlファイルに置き換えることができます。ただし、このチュートリアルではindex.phpファイルを使用し、後でこのファイルを変更します。

6. APX-INFディレクトリにあるファイルをいくつか調べてみます。詳細については、「APXファイル」(118ページ)を参照してください。

APX-INFディレクトリには、APX Webアプリケーションに固有の情報が含まれています。次のcatコマンドで表示されるように、description.txtファイルにはapxtool newの-dオプションで指定したテキストが記録されています。

```
$ ls APX-INF/
description.txt apx.cfg apx.perm usage.txt
$ cat APX-INF/description.txt
This is my first app $
```

次のgrepコマンドは、APX構成ファイルapx.cfgにあるプロパティの一部を表示します。typeとuniquenameの値は、apxtool newコマンドの-tオプションと-uオプションで決まります。APX構成ファイルの詳細については、「APX構成ファイル - apx.cfg」(118ページ)を参照してください。

```
$ grep "=" APX-INF/apx.cfg
type=webapp
```

```
name=mywebapp
unique_name=com.hpe.sa.jdoe.mywebapp
```

新規WebアプリケーションのSAへのインポート

Webアプリケーションをインポートすると、次の処理が実行されます。

- WebアプリケーションがSA内部のHTTPサーバーにインストールされます。
- WebアプリケーションがSAライブラリとGlobal Shellに表示されるフォルダーにコピーされます。
- Webアプリケーションにバージョン番号が割り当てられます。

次に示すように、`apxtool import`コマンドを入力し、プロンプトにyと答えます。`-f`オプションは、Webアプリケーションが格納されるSAライブラリ内のフォルダーを指定します。`-c`オプションは、Webアプリケーションの現在のバージョンを設定します。

```
$ pwd
/home/jdoe/mywebapp/com.hpe.sa.jdoe.mywebapp
$
$ apxtool import -f "/Dev/MyApp" -c
APX source is not specified.
Do you want to publish current directory: /home/jdoe/mywebapp/
com.hpe.sa.jdoe.mywebapp? Y/N y
APX with unique name 'com.hpe.sa.jdoe.mywebapp' does not exist.
Register it into the system? Y/N y
Info: Successfully registered APX 'mywebapp' (310001) in folder ‘/Dev/
MyApp’.
Info: Successfully published a new version '1' for APX 'mywebapp'.
Info: Successfully set APX 'mywebapp'(310001) current version as '1'.
```

新規Webアプリケーションの作成

これでWebアプリケーションが公開されたので、エンドユーザーが行うのと同様に、SAクライアントからWebアプリケーションを実行できます。

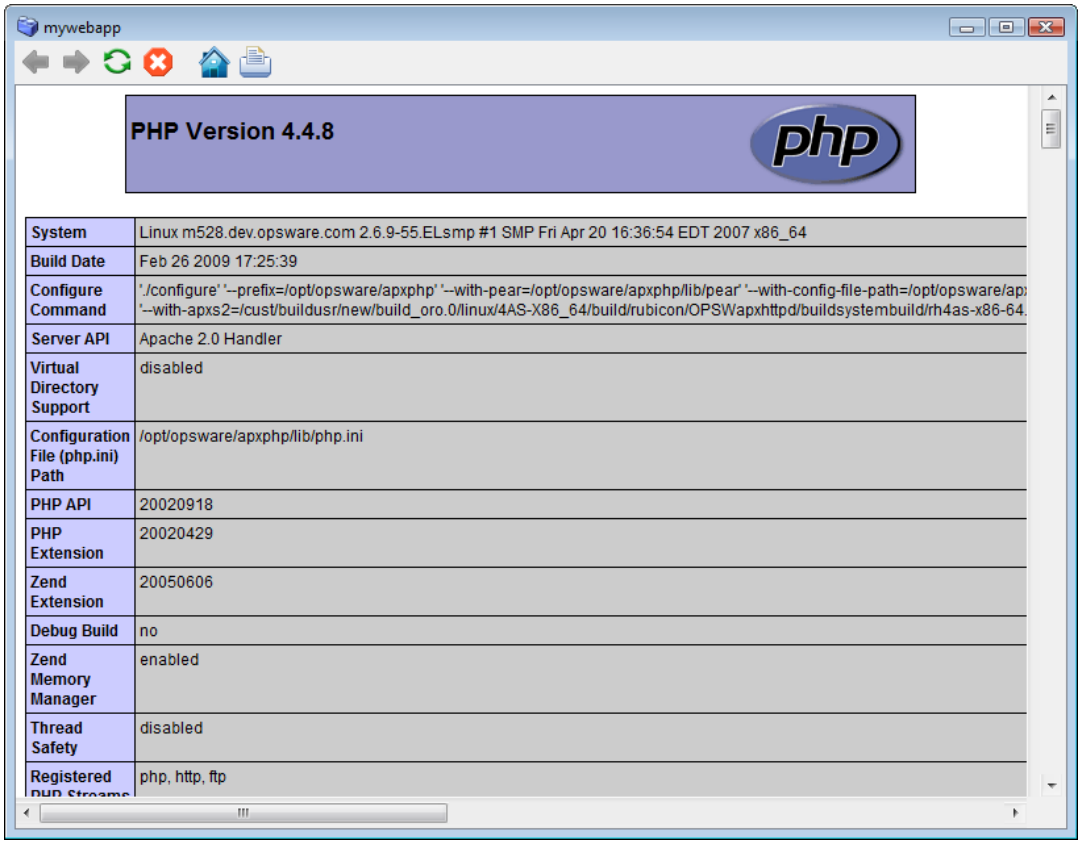
1. **上級ユーザー**グループに属するユーザーとしてSAクライアントにログオンします。
2. [ライブラリ] タブと[タイプ別] タブを選択します。
3. [**拡張**] > [**Web**] ノードに移動すると、`mywebapp`拡張が表示されます。

mywebappが表示されない場合は、「アクセス権の設定とチュートリアルフォルダーの作成」(124ページ)に示す必要なアクセス権があることを確認してください。

4. Webアプリケーションを実行するには、mywebappを選択し、[**アクション**] > [**実行**] メニューを選択します。

   次の図が表示されます。Webアプリケーションには、index.phpファイルのphpinfoステートメントによって生成された情報が表示されます。

   Webアプリケーションバージョン1



Webアプリケーションの変更

デフォルトのindex.phpファイルを実行するのは開発環境を確認するにはよい方法ですが、SAの機能は利用していません。ここでは、index.phpファイルを変更して、SAによって管理されるサーバーの名前のリストを表示します。

1. Global Shellセッションで、Webアプリケーションのindex.phpファイルを見つけます。

   ```
   $ cd /home/jdoe/mywebapp/com.hpe.sa.jdoe.mywebapp
   $ ls
   APX-INF cgi-bin css images index.php
   ```

2. index.phpファイルをviなどのテキストエディターで開きます。
3. index.phpの内容を次の各行に置き換えます。

   ```
   <html>
   <head>
   <title>Servers</title>
   </head>
   <body>
   <p>List of servers:</p>
   <?php
   passthru("ls /opsw/Server/@");
   ?>
   </body>
   </html>
   ```

   上記のpassthruステートメントは、lsコマンドを実行し、stdoutを(再インフレートなしで)Webページに返します。lsコマンドは、OGFSにある管理対象サーバーの名前をリストします。

4. index.phpファイルを保存し、テキストエディターを終了します。
5. 変更したWebアプリケーションを公開します。

   次のapxtool importコマンドは、現在のバージョンを2に設定します。-Fオプションは確認プロンプトを抑制します。

   ```
   $ apxtool import -f "/home/jdoe/mywebapp/com.hpe.sa.jdoe.mywebapp" \
   -c --version=2 -F
   Info: Successfully published a new version '2' for APX 'mywebapp'
   Info: Successfully set APX 'mywebapp'(310001) current version as '2'.
   ```

変更したWebアプリケーションの実行

1. SAクライアントで、**[表示] > [更新]** メニューを使用して、Web拡張の表示を更新します。mywebappのバージョン2が表示されるはずです。
2. mywebappを選択し、[**アクション**] > [**実行**] メニューを選択します。出力はのようになります。実際には、PHPのpassthruステートメントとOGSHのlsステートメントの出力が表示され、管理対象サーバーがリストされます。passthruステートメントは、lsコマンドによって返されるサーバー名の区切りの改行を削除します。

チュートリアル: Program APXの作成

このチュートリアルは、単純なシェルスクリプトを実行するmyshellappという名前の単純なProgram APXを作成し、公開し、実行する方法を示します。チュートリアルの後の方で、シェルスクリプトを変更して、apxprogressコマンドを実行し、進行状況の情報を提供する方法を示します。チュートリアルにはソースコードが付属しているので、シェルプログラミングに関する事前の知識は必要ありません。

次の作業を順番に実行してください。

- 「アクセス権の設定とチュートリアルフォルダーの作成」(131ページ)
- 「新規Program APXの作成」(132ページ)
- 「新規APXのSAへのインポート」(134ページ)
- 「新規APXの実行」(134ページ)
- 「APXの変更」(135ページ)
- 「変更したAPXの実行」(136ページ)
- 「TwisterインタフェースへのAPXの進行状況の表示」(137ページ)

チュートリアルの前提条件

このチュートリアルを実行するには、次の機能と環境が必要です。

- adminまたは**スーパー管理者**グループの他のメンバーとしてSAにログオンできること。adminでログオンすることで、アクセス権を設定できるようになります。

- **上級ユーザー**グループに属するユーザーとしてSAにログオンできること。
- 上級ユーザーには、Webアプリケーションを作成して実行するアクセス権があります。このチュートリアルに示すコマンドの例では、このユーザーの名前は`jdoe`になっています。
- SAクライアントでクライアント機能のアクセス権を設定する方法に関する理解。
- アクセス権の詳細については、『SA 10.50管理ガイド』の「ユーザーおよびユーザーグループの設定とセキュリティ」の項を参照してください。
- SAクライアントでフォルダーを作成する方法に関する理解
- フォルダーの詳細については、『SA 10.50ユーザーガイド』を参照してください。
- Global Shell (OGSH) セッションを開いてGlobal Shellを使用する方法に関する理解。
- `ls`や`cd`などの基本的なUnixコマンドに関する理解。

アクセス権の設定とチュートリアルフォルダーの作成

1. SAクライアントに`admin`でログオンし、`MyApp`フォルダーの [**フォルダーのプロパティ**] を開きます。
2. [**フォルダーのプロパティ**] の [**アクセス権**] タブで、**上級ユーザー**グループに次のアクセス権があることを確認します。
   - フォルダーの内容のリスト表示
   - フォルダー内のオブジェクトの読み取り
   - フォルダー内のオブジェクトの書き込み
   - フォルダー内のオブジェクトの実行
3. SAクライアントを終了します。

新規Program APXの作成

1. **上級ユーザー**グループに属するSAユーザーとしてGlobal Shellセッションを開きます。
2. コアのOGFSホームディレクトリに`myshellapp`という名前のディレクトリを作成し、そのディレクトリに移動します。

   ```
   $ mkdir myshellapp
   $ cd myshellapp
   ```

   Program APXのファイルは、`myshellapp`ディレクトリに格納されます。
3. 次に示すように、`apxtool new`コマンドを使用して、Program APXのディレクトリ構造とデフォルトのファイルを作成します。

   ```
   $ pwd
   /home/jdoe/myshellapp
   $ ls
   $
   $ apxtool new -ts -d "This is my first program APX."\
   -u com.hpe.sa.jdoe.myshellapp

   Create source directory under
   ‘/home/jdoe/myshellapp/com.hpe.sa.jdoe.myshellapp’ for APX ‘myshellapp’? Y/N y
   Info: Successfully created source directory
   ‘/home/jdoe/myshellapp/com.hpe.sa.jdoe.myshellapp for APX ‘myshellapp’.
   ```

   `-ts`オプションはAPXタイプがProgram APX (Script APXとも呼ばれる) であることを示し、`-d`は説明を指定し、`-u`はアプリケーションの一意の名前を指定します。

   `apxtool new`コマンドのオプションの詳細については、次のコマンドでオンラインヘルプを参照してください。

   ```
   $ apxtool new -h
   ```
4. `apxtool new`コマンドで作成されたファイルのリストを表示します。

   ```
   $ pwd
   /home/jdoe/mywebapp
   $ ls
   ```

```
com.hpe.sa.jdoe.myshellapp
$ cd com.hpe.sa.jdoe.myshellapp
$ pwd
/home/jdoe/myshellapp/com.hpe.sa.jdoe.myshellapp
$ ls -R
.:
APX-INF run.sh
./APX-INF:
apx.cfg apx.perm description.txt interfaces usage.txt
```

5. デフォルトのrun.shファイルの内容を表示します。

```
$ cat run.sh
#!/bin/sh

################################################################################
# APX myshellapp
#
# Created by: jdoe
#
################################################################################
echo "This is APX myshellapp"
```

6. APX-INFディレクトリにあるファイルをいくつか調べてみます。これらのファイルの詳細については、「APXファイル」(118ページ)を参照してください。

APX-INFディレクトリには、APXに固有の情報が含まれています。次のcatコマンドで表示されるように、description.txtファイルにはapxtool newの-dオプションで指定したテキストが記録されています。

```
$ ls APX-INF/
apx.cfg apx.perm description.txt interfaces usage.txt
$ cat APX-INF/description.txt
This is my first program APX.$
```

次のgrepコマンドは、APX構成ファイルapx.cfgにあるプロパティの一部を表示します。typeとuniquenameの値は、apxtool newコマンドの-tオプションと-uオプションで決まります。APX構成ファイルの詳細については、「APX構成ファイル - apx.cfg」(118ページ)を参照してください。

```
$ grep "=" APX-INF/apx.cfg
type=script
name=myshellapp
```

```
unique_name=com.hpe.sa.jdoe.myshellapp
command=run.sh
```

新規APXのSAへのインポート

APXをインポートすると、次の処理が実行されます。

- SAライブラリに表示されるフォルダーにAPXがコピーされます。
- APXにバージョン番号が割り当てられます。

次に示すように、`apxtool import`コマンドを入力し、プロンプトに`y`と答えます。`-f`オプションは、Webアプリケーションが格納されるSAライブラリ内のフォルダーを指定します。`-c`オプションは、Webアプリケーションの現在のバージョンを設定します。

```
$ pwd
/home/jdoe/myshellapp/com.hpe.sa.jdoe.myshellapp
$
$ apxtool import -f "/Dev/MyApp" -c
APX source is not specified.
Do you want to publish current directory: /home/jdoe/myshellapp/
com.hpe.sa.jdoe.myshellapp? Y/N y
APX with unique name 'com.hpe.sa.jdoe.myshellapp' does not exist.
Register it into the system? Y/N y
Info: Successfully registered APX 'myshellapp' (20001).
Info: Successfully published a new version '1' for APX 'myshellapp'
Info: Successfully set APX 'myshellapp'(20001) current version as '1'.
```

これでAPXが公開されたので、他のSAユーザーが行うのと同様に、SAクライアントからAPXを実行できます。

新規APXの実行

これでAPXが公開されたので、SAクライアントからAPXを実行できます。

1. **上級ユーザー**グループに属するユーザーとしてSAクライアントにログオンします。
2. [ナビゲーション] ペインで、[ライブラリ] タブを選択し [タイプ別] タブを選択します。
3. [拡張] ノードを開き、[プログラム] ノードを選択します。SAライブラリ内のすべてのProgram APXが表示されます。作成したAPXがここに表示されているはずです。`myshellapp`が表示されない場合は、

「アクセス権の設定とチュートリアルフォルダーの作成」(131ページ)に示す必要なアクセス権があることを確認してください。

4. APXを選択します。
5. [**アクション**] > [**実行**] メニュー項目を選択します。[プログラム拡張の実行] ウィザードが表示されます。
6. [次へ] ボタンを選択します。
7. [ジョブの開始] ボタンを選択します。
8. APXが終了したら、ステータスインジケーターを選択して詳細を表示します。
9. [閉じる] ボタンをクリックします。

APXの変更

このセクションでは、`run.sh`ファイルを変更して、進行状況の情報を提供するための`apxprogress`コマンドの呼び出しを追加します。

1. Global Shellセッションで、APXの`run.sh`ファイルを見つけます。

   ```
   $ cd /home/jdoe/myshellapp/com.hpe.sa.jdoe.myshellapp
   $ ls
   APX-INF run.sh
   ```

2. `run.sh`ファイルを`vi`などのテキストエディターで開きます。
3. `run.sh`の内容を次の各行に置き換えます。

   ```
   echo "This is a simple APX that uses apxprogress."


   totalsteps=100
   apxprogress -i $totalsteps -c 1
   for i in `seq $totalsteps`; do
           apxprogress -c $i -m "myshellapx is running, working on step $i" #-d
           sleep 10
   done
   ```

これらのapxprogressコマンドでは、APXに100のステップがあることを指定し、apxprogressを各ステップに1回、合計100回呼び出して、呼び出しの間に10秒間ずつ待ちます。詳細については、「APXの進行状況の表示」(121ページ)を参照してください。

デバッグ用には、"#-d" を "-d" に変更し、シェルスクリプトを手動で実行することで、apxprogressコマンドの出力をstdoutに表示できます。

4. run.shファイルを保存し、テキストエディターを終了します。
5. 変更したAPXを公開します。

   次のapxtool importコマンドは、APXの新しいバージョンをロードし、現在のバージョンを2に設定します。-Fオプションは確認プロンプトを抑制します。

```
$ apxtool import -f "/home/jdoe/myshellapp" \
-c --version=2 -F
Info: Successfully published a new version '2' for APX 'myshellapp'
Info: Successfully set APX 'myshellapp'(20001) current version as '2'.
```

変更したAPXの実行

これでAPXが変更され、再公開されたので、前のようにSAクライアントからAPXを実行できます。

1. SAクライアントで、**[表示] > [更新]** メニューを使用して、プログラム拡張の表示を更新します。myshellappのバージョン2が表示されるはずです。
2. APXを選択します。
3. [**アクション**] > [**実行**] メニュー項目を選択します。[プログラム拡張の実行] ウィザードが表示されます。
4. [次へ] ボタンを選択します。
5. [ジョブの開始] ボタンを選択します。

TwisterインタフェースへのAPXの進行状況の表示

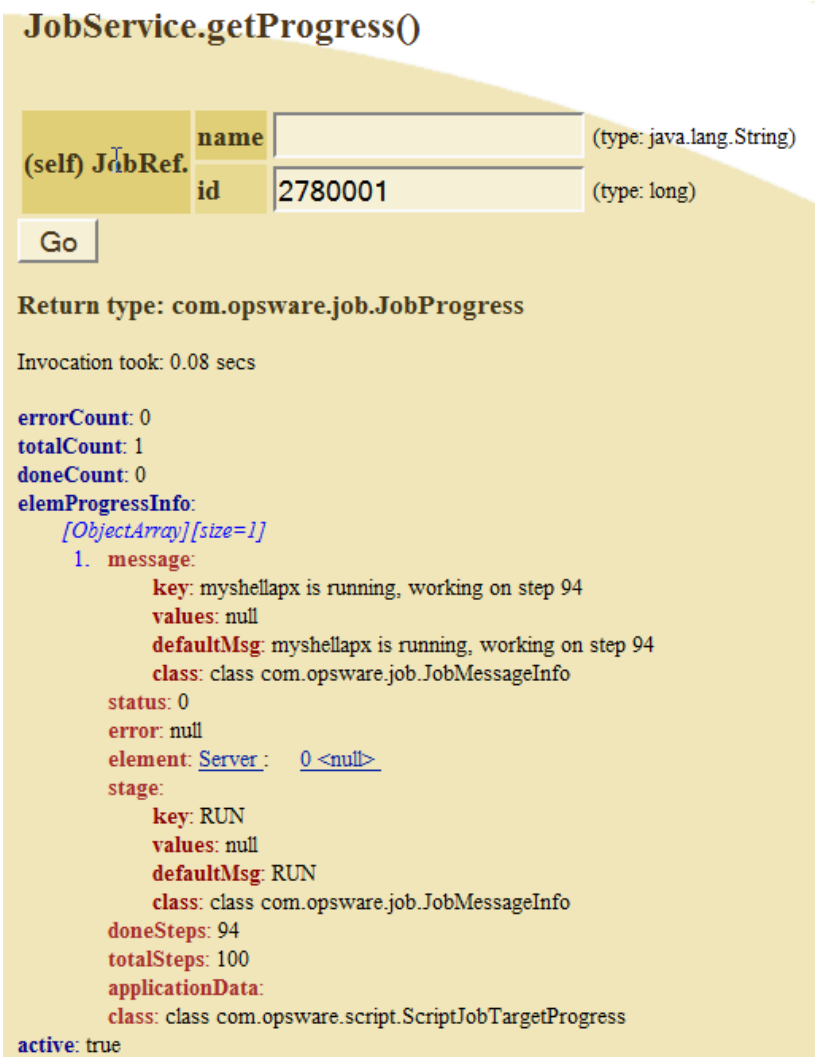
apxprogressコマンドは、実行中のAPXの進行状況を報告します。この進行状況情報は、APIメソッドJobService.getProgress()を呼び出すことによって得られます。ここでは、このメソッドをTwisterインタフェースから実行する方法を示します。SA APIに対するTwisterインタフェースの詳細については、「APIドキュメントとTwister」(37ページ)を参照してください。

1. SAクライアントで、[ジョブとセッション] タブを選択します。
2. ジョブのリストで、作成したAPXを見つけます。
3. APXジョブのジョブID番号を記録します。これは後のステップで使用します。
4. Webブラウザーに次のURLを入力して、SA Twistインタフェースを実行します。

   ```
   https://<コアホスト>:1032
   ```

   ここで、<コアホスト>はSAコアサーバーのIPアドレスまたはホスト名です。これは、SA APIに対するTwistインタフェースをWebブラウザーに表示します。
5. “Twister” リンクを選択します。これにより、SA APIに対するTwisterインタフェースが表示されます。ここでは、APIインタフェース、パッケージ、メソッドに関する詳細な情報を取得したり、メソッドを実行したりすることができます。
6. JobServiceインタフェースを見つけて選択します。これはcom.opsware.jobパッケージにあります。
7. 下の方へスクロールして、getProgress()メソッドを見つけます。
8. getProgress()メソッドのすぐ上の [試行] ボタンを選択します。
9. SAの資格情報を入力します。
10. [ログイン] ボタンを選択します。
11. [ID] フィールドに、上の手順3で記録した実行中のAPXのジョブ番号を入力します。
12. [**実行**] ボタンを選択します。getProgress()メソッドが呼び出され、apxprogressコマンドによるAPXの現在の進行状況情報が次のように表示されます。このスナップショットでは、総ステップ数が100で、完了したステップ数は94です。getProgress()メソッドの出力の詳細については、Twister Webブラウザーの [ナビゲーション] ペインでgetProgress()メソッドを選択して、Javadocsドキュメントを参

照してください。


JobService.getProgress()
(self) JobRef.
name
(type: java.lang.String)
id
2780001
(type: long)
Go
Return type: com.opsware.job.JobProgress
Invocation took: 0.08 secs
errorCount: 0
totalCount: 1
doneCount: 0
elemProgressInfo:
[ObjectArray][size=1]
1. message:
key: myshellapx is running, working on step 94
values: null
defaultMsg: myshellapx is running, working on step 94
class: class com.opsware.job.JobMessageInfo
status: 0
error: null
element: Server : 0 <null>
stage:
key: RUN
values: null
defaultMsg: RUN
class: class com.opsware.job.JobMessageInfo
doneSteps: 94
totalSteps: 100
applicationData:
class: class com.opsware.script.ScriptJobTargetProgress
active: true

エージェントツール

エージェントツールの概要

エージェントツールは、管理対象サーバーに関する情報の取得と変更のためのシェルスクリプト、バッチファイル、Pythonスクリプトの集合です。情報の取得と変更は、SAデータベースから行われます。

これらのスクリプトを使用すると、カスタムフィールド、カスタマー割り当て、カスタム属性などを取得し、変更することができます。この機能により、従来サーバー単位で実行する必要があったさまざまな手順を自動化できます。

さらに、スクリプトで取得した情報を、独自設計のカスタムスクリプトに組み込むこともできます。カスタマー割り当てやカスタム属性といった情報は管理対象サーバーごとに異なるため、カスタムスクリプトでこのような情報を「実行時」に取得して使用できれば非常に便利です。

以下に例を挙げます。

- アプリケーションのインストール後の構成作業を行うスクリプトで、サーバーが登録されているファシリティの名前を知る必要があるとします。エージェントツールに含まれるスクリプトを使用すれば、ファシリティ名を取得して、インストール後スクリプトに自動的に挿入することができます。
- 監視エージェントをインストールする際に、インストール後スクリプトは、構成ファイルを変更して、当該ファシリティでの監視サーバーのIPアドレスを書き込む必要があります。エージェントツールに含まれるスクリプトを使用すれば、コアのカスタム属性を読み取って監視サーバーのIPアドレスを取得し、構成ファイルに挿入することができます。
- 多数のサーバーのEEPROMバージョンを取得して、その情報をカスタム属性またはカスタムフィールドに格納するDSEを作成できます。

エージェントツールのスクリプトには、他にも次のような使用法があります。

- ソフトウェアのインストール時に構成に使用する情報をSAコアから取得。
- DSE、Global Shellスクリプト、ソフトウェアのインストールの実行時に、管理対象サーバーからのメタデータをSAデータベースに格納。
- 管理対象サーバーに関するカスタム属性情報を取得。

インストール要件

エージェントツールの要件を次に示します。

オペレーティングシステムのサポート

エージェントツールは、SA管理対象サーバーがサポートするオペレーティングシステムをサポートします。サポートされるオペレーティングシステムのリストについては、『Server Automationインストールガイド』を参照してください。

セキュリティ、アクセス制御、認証

エージェントツールは、Unix/Linuxシステムではrootユーザーで、Windowsシステムでは管理者で実行する必要があります。エージェントツールは、サーバーエージェントの証明書を使用して、Webサービスデータアクセスエンジン (twist) に接続し (これはpyTwistのデフォルトの動作)、Webサービスデータアクセスエンジンがエージェントに与える権限を付与されます。これは通常、エージェントツールが実行されるサーバー上の読み取り/書き込み権限に当てはまるので、ユーザー認証は不要です。

例外として、`set_customer`スクリプトがあります。サーバーをカスタマーに関連付けるには、そのカスタマーに対する読み取りアクセス権が必要です。エージェント証明書は他のカスタマーに対する読み取りアクセス権を持たないため、このスクリプトを実行する際にユーザーは認証を行う必要があります。

UAC (ユーザーアクセス制御) が有効な場合、Windows上でのエージェントツールのスクリプトの実行はサポートされません。

その他の要件

- pyTwistへのアクセス権限
- SA APIへのアクセス権限
- インストール済みのPython 2.4 (サーバーエージェントに付属)

インストール

エージェントツールは、HPE SAインストーラーの通常のコアインストールプロセス中にコアにインストールされます。ただし、エージェントツールを管理対象サーバー上で使用できるようにするには、これらのサーバーにもエージェントツールをインストールする必要があります。ここではそのプロセスについて説明します。

エージェントツールは、実行可能スクリプトの集合として管理対象サーバーにインストールされます。これには、シェルまたはバッチスクリプト (オペレーティングシステムに依存) と、これらのスクリプトから呼び出されるPythonスクリプトがあります。これらのスクリプトを管理対象サーバーから実行することにより、SAコアの情報の取得と変更が可能です。これらのスクリプトは、手動で実行することも、パッケージインストールスクリプト、DSE、Global Shellシェルスクリプトなどから呼び出すこともできます。

エージェントツールは、Python SA APIアクセス (pyTwist) ソフトウェアポリシーの一部として提供されています。このポリシーは次のディレクトリにあります。

```
/Opsware/Tools/Python Opsware API Access
```

エージェントツールの手動インストール

管理対象サーバーにエージェントツールをインストールするには、次の手順を実行します。

1. SAクライアントを起動します。
2. [**管理対象サーバー**] リストを表示し、エージェントツールをインストールする管理対象サーバーを選択します。
3. 右クリックして [**ソフトウェアのインストール**] を選択します。
4. [**Python Opsware APIアクセス**] ソフトウェアポリシーを選択します。
5. ソフトウェアポリシーのインストールウィザードが、プロセスの残りの手順をガイドします。

エージェントのインストール時のエージェントツールのインストール

別の方法として、エージェントのインストール時にPython SA APIアクセスソフトウェアポリシーのIDを指定して、これを修復するように指定することもできます。エージェントのインストールの詳細については、管理を

参照してください。

エージェントツールのアップグレード

エージェントツールはソフトウェアポリシー (pyTwistソフトウェアポリシーの一部) として提供されているため、コアのアップグレード後に修復を実行することで、エージェントツールの新しいバージョンにアップグレードすることができます。

SAコアをアップグレードすると、Python SA APIアクセスソフトウェアポリシーも更新されます。エージェントツールの古いバージョンは削除され、新しいバージョンがポリシーにアタッチされます。SAコアのアップグレード (この際にエージェントツールもコアのアップグレードの一部として自動的にアップグレードされます) の後で、管理対象サーバー上のエージェントツールをアップグレードするには、次の作業を実行します。

1. エージェントツールがインストールされている管理対象サーバーを選択します。Python SA APIアクセスソフトウェアポリシーにアタッチされているサーバーとグループのリストを表示するには、ポリシー自体を開きます。
2. 選択したサーバーを右クリックして [**修復**] を選択します。
3. [**Python Opsware APIアクセス**] ソフトウェアポリシーを選択します。
4. pyTwistとエージェントツールパッケージの古いバージョンが削除され、新しいバージョンがインストールされます。

データ移行

エージェントツールは管理対象サーバー上に永続的なデータを保持しないため、データの移行や保存の必要はありません。

エージェントツールのスクリプト

使用法

<スクリプト名>.py|bat|sh --引数

エージェントツールのスクリプト

スクリプト	機能
get_all_cust_attr	サーバーレコードのすべてのカスタム属性を取得します。 使用法: get_all_cust_attr.py [--localonly] [--mode=python|shell|pretty] modeは、出力の形式 (Python辞書、シェルステートメントなど) を指定します。デフォルトはprettyです。 **注:** 複数行のカスタム属性がある場合、modeにshellは使用できません。
get_cust_attr	1つのカスタム属性の値を取得します。 **使用法**: get_cust_attr.py [--localonly] <カスタム属性名>
set_cust_attr	サーバー上の1つのカスタム属性の値を設定します。 **使用法**: set_cust_attr.py <カスタム属性名> <カスタム属性値>|--valuefile <値を記録したファイルのパス>
del_cust_attr	データベース内のサーバーのレコードからカスタム属性を削除します。 **使用法**: del_cust_attr.py <カスタム属性名>
get_cust_field	1つのカスタムフィールドの値を取得します。 **使用法**: get_cust_field.py <カスタムフィールド名>
set_cust_field	サーバー上の1つのカスタムフィールドの値を設定します。 **使用法**: set_cust_field.py <カスタムフィールド名> <カスタムフィールド値>|--valuefile <値を記録したファイルのパス>
get_customer	サーバーが関連付けられているカスタマー名を取得します。

エージェントツールのスクリプト (続き)

スクリプト	機能
	使用法: ./get_customer.py
set_customer	サーバーが関連付けられているカスタマー名を設定します。 **使用法**: set_customer.py<カスタマー名>
get_facility	サーバーが関連付けられているファシリティの名前を取得します。 **使用法**: ./get_facility.py
get_info	サーバーのすべてのフィールドを (OGSHのサーバーの情報ファイルと似た形式で) 出力します。 **使用法**: get_info.py
get_history	サーバー固有のイベントを印刷します。 **使用法**: get_history.py --startdate <エポックを基準とした開始日 (秒単位)> [--enddate <エポックを基準とした終了日 (秒単位)>] [--username <SASユーザー名>] [--password <SASパスワード>]
sub_text_file	テキストファイルを読み込み、ファイルからトークン/パラメーターを探して、カスタム属性の値に置き換え、変更したファイルをstdoutに出力します。使用できるファイル形式については下を参照してください。 **使用法**: sub_text_file.py [--localonly] <トークンを記録したファイルのパス>

sub_text_fileスクリプトに使用できる形式

sub_text_fileスクリプトに渡すテキストファイルは任意の内容を持つことができますが、スクリプトは@文字が2個ある行を探し、@文字のペアとその間の文字列をトークンとして扱います。行に@文字が1個あってもその行は無視されますが、同じ行にもう1個@文字があると、2個の@文字の間のすべての文字列がトークンと見なされます。

トークンは、@文字の間に指定されたカスタム属性の値に置き換えられます。たとえば、@dns_server@という文字列は、カスタム属性dns_serverの値に置き換えられます。このカスタム属性が存在しないか値が空の場合は、トークンは空文字列に置き換えられます。

次のエントリを含むテキストファイルがあるとします。

```
IP: @monitoring_server_ip@
```

スクリプトの出力は次のようになります。

```
IP: 82.159.202.117
```

ここで、IPは`monitoring_server_ip`によって取得される値です。

出力

`sub_text_file`スクリプトはstdoutに出力します。必要な場合、出力をファイルにリダイレクトできます。また、zipファイルに格納されている`.template`ファイルを使用して、出力をフォーマットできます。以下に例を挙げます。

```
$AGENTTOOLSPATH/sub_text_file.sh petstore_config.template > petstore_config.cfg
```

エージェントツールのスクリプトの例

エージェントツールのスクリプトの簡単な使用例を次に示します。

Unix/Linux

この例は、ファシリティの名前を含むメッセージを、ユーザーがUnixサーバーにログインしたときに表示されるMOTD (Message of the Day) に挿入します。

```
. /etc/opt/opsware/pytwist/pytwist.conf
facility_name=`$AGENTTOOLSPATH/get_facility.sh`
echo "You have connected to a server in the $facility_name facility.For hardware
information on this server as stored in Opsware, run $AGENTTOOLSPATH/get_info.sh.">
/etc/motd
```

Windows

このWindowsの例は、サーバーに関する情報を記録したテキストファイルを、すべてのユーザーのデスクトップに置きます。

```
call "C:\Program Files\Common Files\Opsware\etc\pytwist\
pytwist_conf.bat"
```

```
call"%AGENTTOOLSPATH%\get_info.bat" > "%SYSTEMDRIVE%\Documents and Settings\All
Users\Desktop\server_info_from_Opsware.txt"
```

1. エージェントツールへのパスはコードに直接書き込まず、次の方法を使用してください。

   PyTwist構成ファイルを実行します。

UNIXの場合:
`./etc/opt/opsware/pytwist/pytwist.conf`

Windowsの場合:
次のファイルを呼び出します。
`C:\Program Files\Common Files\Opsware\etc\pytwist\pytwist_conf.bat`

2. 環境変数を使用します。

UNIXの場合:
`$AGENTTOOLSPATH`

Windowsの場合:
`%AGENTTOOLSPATH%`

この方法を使用すれば、エージェントツールへのパスが将来変更されても、スクリプトのエラーを防ぐことができます。

Microsoft Windows PowerShell - SA統合

Microsoft Windows PowerShellの概要

Windows PowerShellは、Microsoftの.Net 2.0 Frameworkクラスライブラリに統合された、システム管理者やプログラマーのための拡張可能なコマンドシェルです。.NETの共通言語ランタイムと.NET Frameworkを使用し、.NETオブジェクトを受け取って返します。Windowsの管理と構成のためのツールと方法を強化する役割を果たします。

Windows PowerShellには多数の「コマンドレット」が装備されており、さまざまな機能を提供します。コマンドレットは個別に使用することも、組み合わせてもっと複雑な作業のために使用することもできます。

Windows PowerShellは、コンピューターのファイルシステムへのアクセスを提供するだけでなく、PowerShellプロバイダーによりレジストリやデジタル署名証明書ストアなどのデータストアへのアクセスを可能にします。プロバイダーとは、サービスとデータソースの間の一様なインタフェースを提供するソフトウェアモジュールです。

SAでWindows PowerShellを使用するには、Microsoft Windows PowerShellの知識と使用経験が前提となります。PowerShellに関する情報や手順については、http://www.microsoft.com/ja-jpを参照してください。

> **注意:** 付属のコマンドレットは管理対象サーバー上のデータを変更できるので、Windows PowerShellとその使用法について十分に理解しておくことが重要です。

Windows PowerShellとSAとの統合

SAでは、Windowsが動作する管理対象サーバーでのMicrosoft Windows PowerShellとの初期統合が提供されています。PowerShellはSAユーザーインタフェースから使用でき、SAデータは標準PowerShell環境または任意のPowerShell実行空間から利用できます。PowerShell実行空間とは、PowerShellランタイムシステムのホスティング環境です。

SAでは次のPowerShellコマンドレットが使用可能です。

- Get-SASServer
- Set-SASServer
- Get-SASJob

また、SAにはPowerShell SASプロバイダー(PowerShell環境でSAコア内のオブジェクトにアクセスするためのコンポーネント)も付属しています。

統合PowerShell/SAコマンドレット

次に、SA付属の統合PowerShell/SAコマンドレットの一覧と説明を示します。

PowerShellコマンドレット

コマンドレット	説明	引数

`Get-SASServer`	Retrieves server data from specified server(s)	`-Credential <PSCredential>`
		`-Core <Hostname|IPAddress>`
		`-Name <ListOfHostnameFragments> |`
		`-Id <ListOfServerIDs>`

`Get-SASJob`	Retrieves data for specified jobs	`-Credential <PSCredential>`
		`-Core <Hostname|IPAddress>`
		`-JobFilter <ListOfJobIDs>`

`Set-SASServer`	Retrieves a list of managed servers	`-Credential <PSCredential>`
		`-Core <Hostname|IPAddress>`
		`-Server <ServerVO>`

注意: ターゲットコアでTLSv1.xの最小プロトコルバージョンが実行されている場合は、Powershellバージョン(バインドされた.NET Frameworkのバージョン)がこれをサポートしている必要があります。詳細については、https://msdn.microsoft.com/en-us/library/system.net.securityprotocoltype(v=vs.110).aspxを参照してください。

インストール要件

システム管理者のWindowsデスクトップにインストールするコマンドレットとPowerShell SAプロバイダーのアセンブリ、構成、セットアップファイルを含むMSIインストーラーパッケージ。

オペレーティングシステムのサポート

- Windows Server 2003
- Windows Server 2008
- Windows Server 2008 R2 x64
- Windows Server 2012

インストール

Microsoft Windows PowerShell/SA統合を実装するには、次の作業を実行します。

- OCCの [**ライブラリ**] > [**ソフトウェアポリシー**] で、Microsoft Windows PowerShell/SAコネクターMSIを見つけます。
- MSIを実行して、SA固有のコマンドレットとSAプロバイダーを定義するアセンブリをインストールします。ファイル`readme.rtf`に最新情報が記載されています。Microsoft Windows PowerShell初期化スクリプトprofile.ps1 (.bashrcと同様のもの) と、SA環境でのPowerShellの使用方法を示すサンプルPowerShellスクリプトのセットもインストールされます。

デフォルトでは、MSIはコネクターを`C:\Program Files\Opsware\PsSas`にインストールします。

ファイル`SAS-WSAPI.ps1`は、コマンドレットを使用せずにPowerShellから直接WS-APIにアクセスする方法を示します。

Microsoft Windows PowerShell/SA統合の機能

Microsoft Windows PowerShellは次の分野でオプションとして使用できます。

管理対象サーバーへのリモートアクセス

SAクライアントから、リモートターミナルを開くのと同じ方法で、任意の管理対象サーバー(サーバーのグループは不可)に対するリモートPowerShellセッションを開くことができます。

1. SAクライアントを起動します。
2. ナビゲーションペインで [**デバイス**] > [**すべての管理対象サーバー**] を選択します。
3. 管理対象サーバーを選択して開きます。

   デバイスエクスプローラーウィンドウで、[**アクション**] メニューから [**リモートPowerShellの起動**] を選択します。

リモートPowerShellセッションにログインしている間は、「WMI呼び出し」を含むスクリプトは実行できません。WMI呼び出しを含むスクリプトを実行しようとすると、そのスクリプトの実行に必要なアクセス権を持つグループのメンバーであっても、アクセスが拒否されましたエラーが発生します。

監査とスナップショットのルール

Microsoft PowerShellは、SA監査と統合されています。カスタムスクリプトルールを構成する際のオプションとして、バッチ、Python 2、Visual Basicに加えてMicrosoft PowerShellスクリプトが使用できるようになりました。監査の詳細については、HPE SSOポータルで『Server Automation管理ガイド』を参照してください。

DSEスクリプト統合

管理対象サーバーに対して、Pytwistを使用してSA APIを呼び出すPowerShellスクリプトを用意すれば、エンドユーザーはスクリプトをDSEまたはISMコントロールとして呼び出すことができます。Pytwist APIを呼び出すスクリプトの作成方法については、「PytwistによるPython APIアクセス」(78ページ)を参照してください。

サンプルセッション

ここでは、Windows PowerShell/SA統合の使用方法を示す4つのシナリオを紹介します。

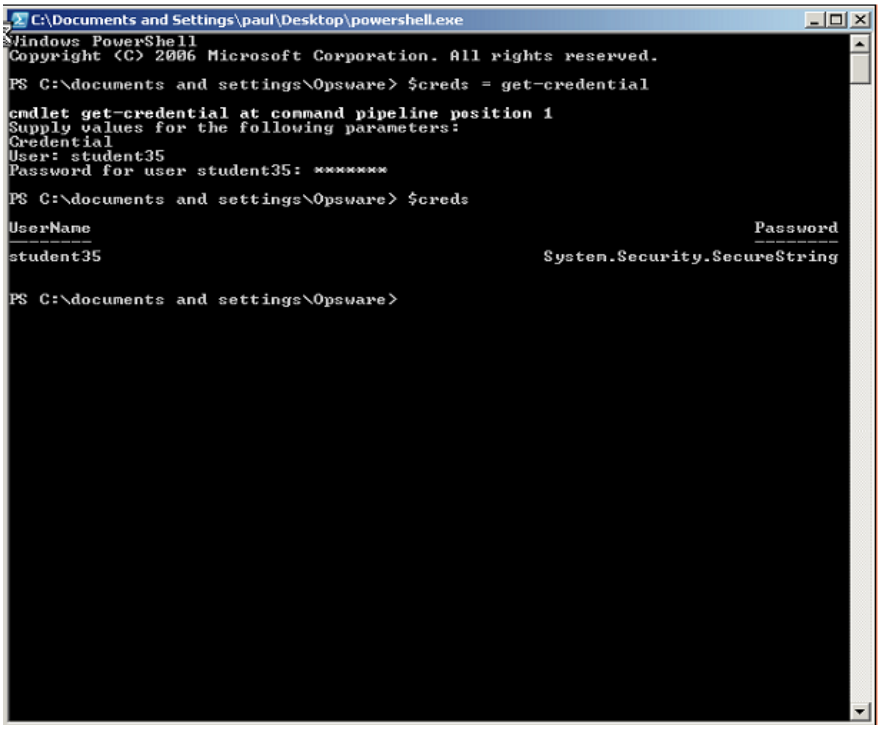
- 「シナリオ1」(155ページ)は、SAコアから管理対象サーバーデータを抽出して変更し、コアに書き戻す方法を示します。
- 「シナリオ2」(160ページ)は、SA管理対象サーバーデータをWindows PowerShell/SA統合を使用してExcelスプレッドシートにエクスポートする方法を示します。
- 「シナリオ3」(162ページ)は、SAコアをWindows PowerShell PSdriveとしてマウントし、仮想ファイルシステム内を調べる方法を示します。
- 「シナリオ4」(165ページ)は、Windows PowerShell環境で使用可能なすべてのタイプのSAオブジェクトをリストする方法を示します。

シナリオ1

SAコアに対する認証を行い、管理対象サーバーに関するデータを取得し、データを変更し、SAコアに書き戻します。

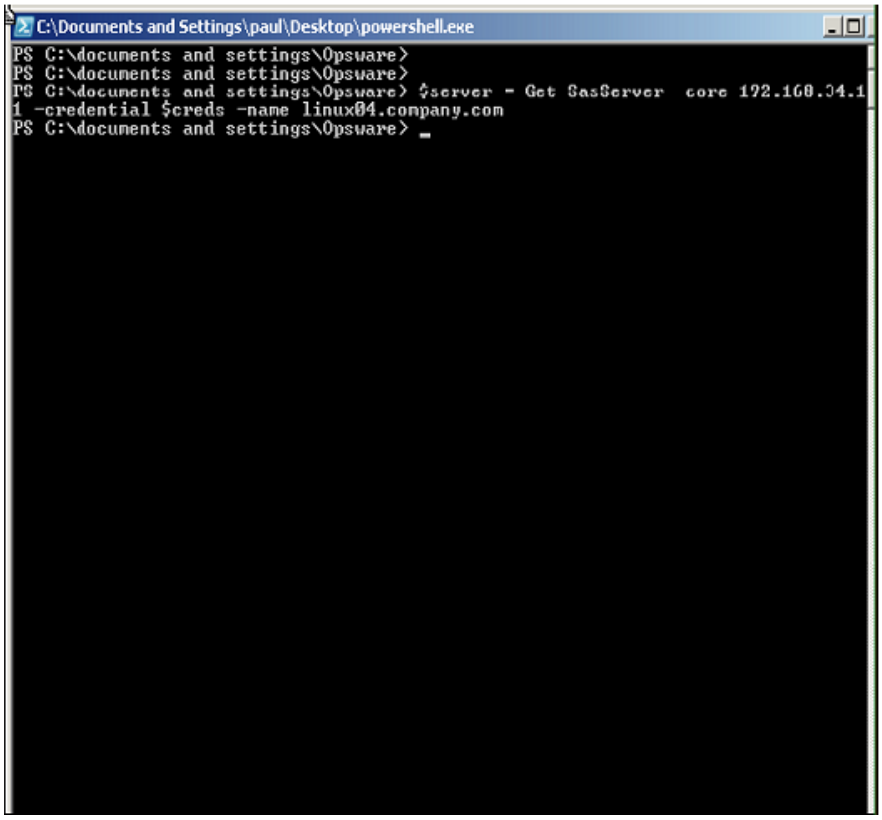
1. デスクトップアイコンからPowerShellプロンプトを開きます。
2. SAコア資格情報をPowerShellシェル変数にセキュアに格納します。次の図を参照してください。

SA資格情報のPowerShell変数への格納



3. Get-SasServerコマンドレットを使用すると、次の図に示すようにサーバーを表すSAレコードを取得できます。

   Get-SasServerコマンドレットの使用



返されたオブジェクトはシェル変数に格納されます。

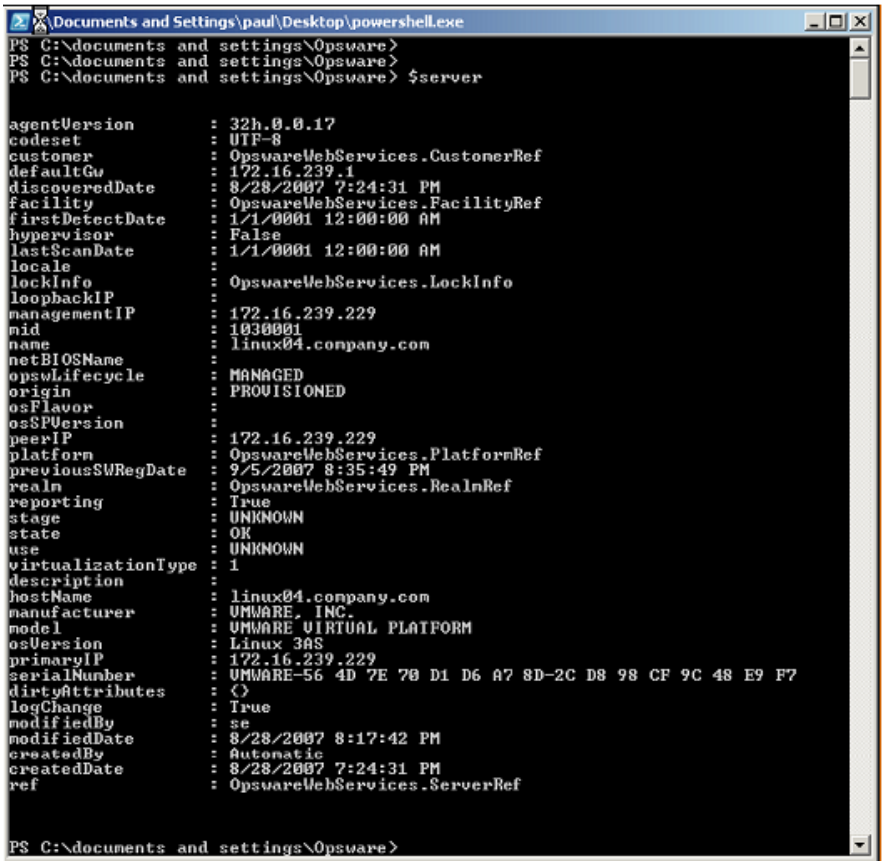
Get-SasServerコマンドレットは、サーバーデータを取得するコアを識別するパラメーター、コアに操作のための資格情報 (操作を実行するユーザーアカウントの識別と認証に使用) を提供するパラメーター、要求対象のサーバーを識別するパラメーターを取ります。

Get-SasServerを初め、すべてのコマンドレットの引数に関する詳細を知るには、PowerShellのGet-Helpベースコマンドレットを次のように使用します。

```
Get-Help Get-SasServer -detailed
```

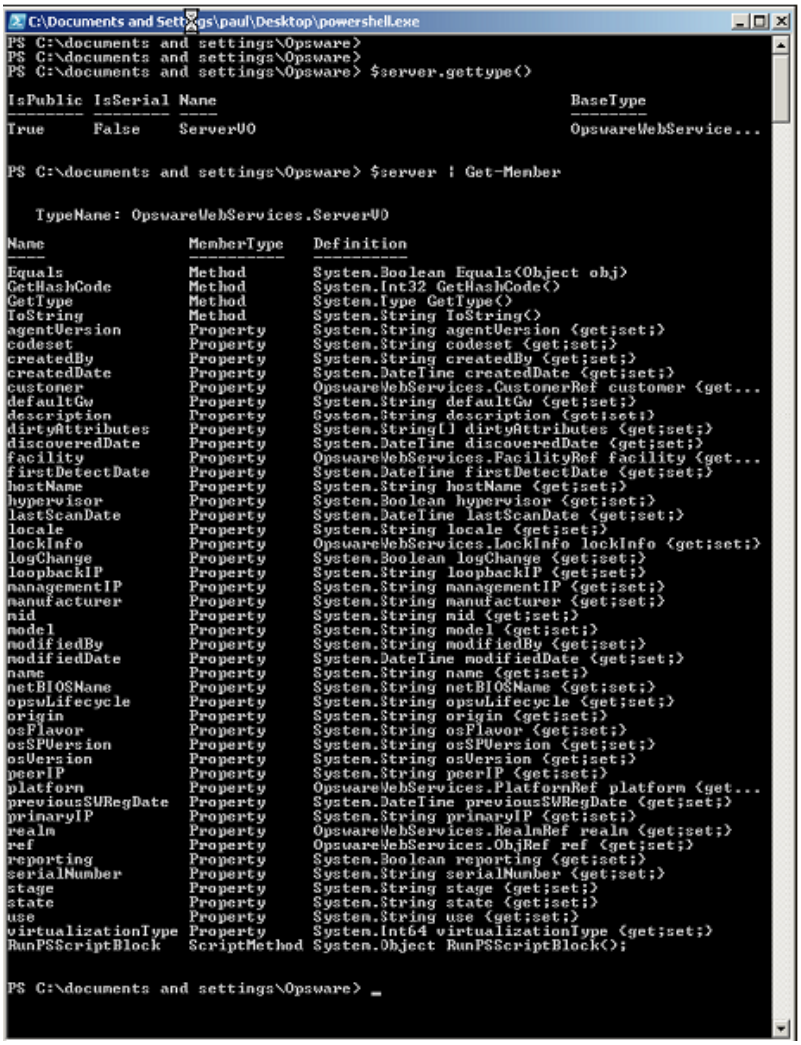
4. 返されたオブジェクトのプロパティを見るには、シェル変数の名前を入力します。次の図を参照してください。

   SAサーバーのプロパティの表示



5. オブジェクトのプロパティ、プロパティのタイプ、PowerShellスクリプトからオブジェクトに対して呼び出せるメソッドのリストを表示する方法については、次の図を参照してください。

   オブジェクトのプロパティのリストの表示



6. オブジェクトの**説明**属性をWindows PowerShellで変更した後、Set-SasServerコマンドレットを呼び出して、変更したServerVOオブジェクトをコマンドレットに渡すことができます。このコマンドレットは、ServerVOオブジェクトを受け取って、SAコアの管理対象サーバーレコードを更新します。Set-SasServerコマンドレットは、更新したデータを書き込むSAコアを識別するパラメーターと、操作を実行するSAユーザーアカウントを識別する資格情報のパラメーターを受け取ります。

   更新操作が完了すると、更新されたServerVOがWindows PowerShellに返され、次の図に示すようにプロパティがプロンプトに表示されます。

   オブジェクトの説明の変更

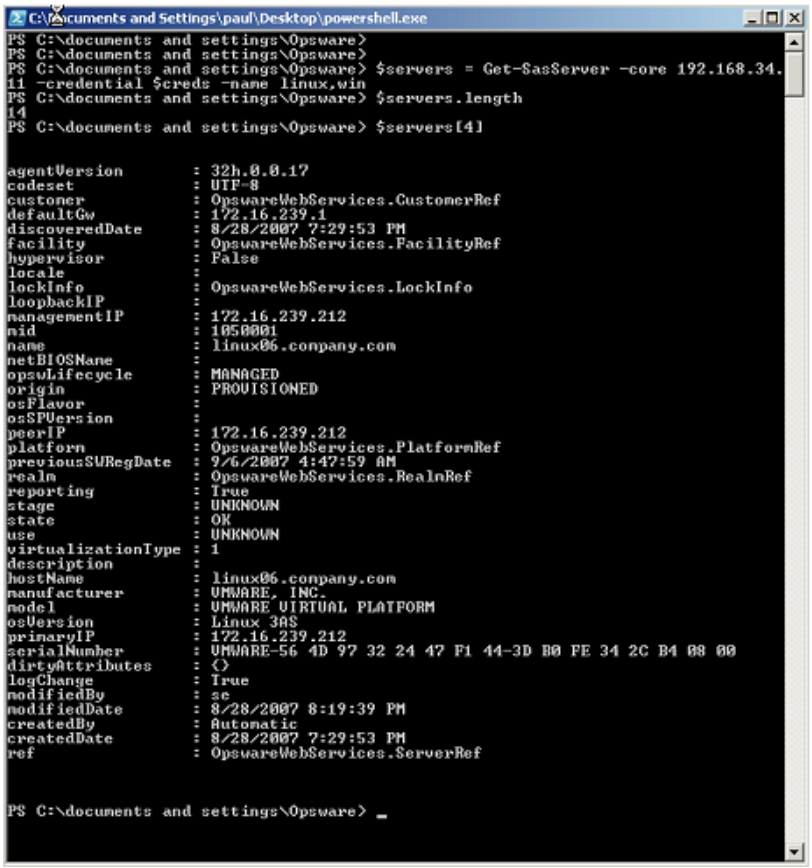
シナリオ2

このシナリオは、SAコアからすべての管理対象サーバーデータを取得し、Microsoft Excelに表示する方法を示します。

1. `Get-SasServer`コマンドレットを使用して、各LinuxおよびWindows管理対象サーバーに対応するServerVOをSAコアから取得します。次に示すセッションで、`-name`パラメーターは名前に一致するフィルターのリスト (例、`-name linux,win`) をSAコアに提供するために使用されています。

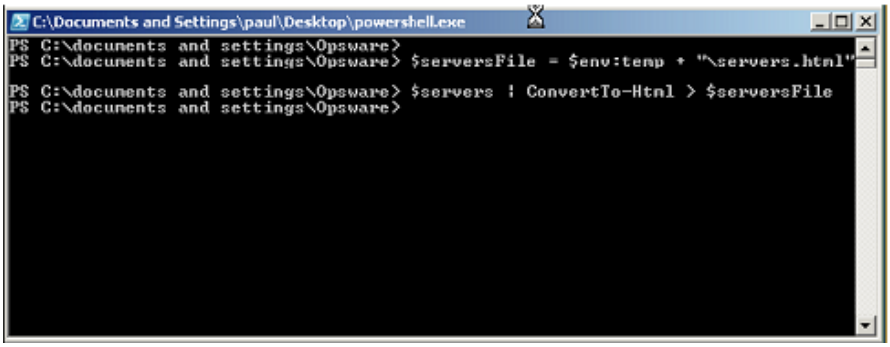
   `Get-SasServer`コマンドレットはServerVOの配列を返します。この例では配列の長さは14です。この配列にインデックスを指定することで、任意のServerVOオブジェクトを得ることができます。次の図を参照してください。

   Get-SasServerコマンドレットでの名前フィルターの使用



2. その後、ServerVOデータをHTMLにフォーマットして、一時ファイルに保存できます。一時ファイルはTEMPディレクトリに作成されます。PowerShellセッションで、%TEMP%環境変数の値を取得するには、「$env:temp」と入力します。次の図を参照してください。

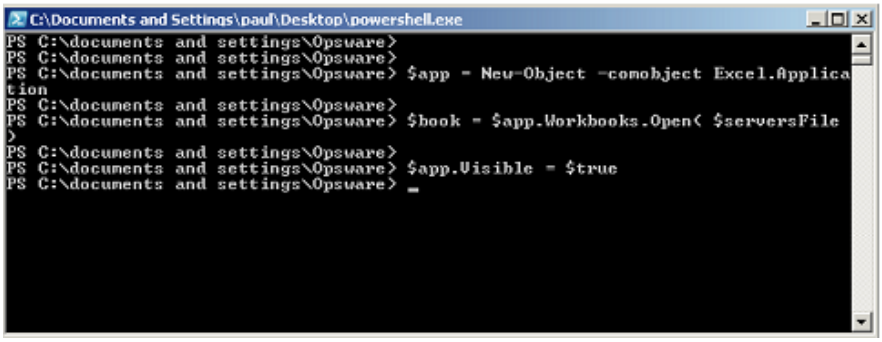
ServerVOデータをHTMLに変換して一時ファイルに保存



3. ベースWindows PowerShellコマンドレットのNew-Objectを使用して、Microsoft Excelを起動し、そのExcelインスタンス内部に新規ワークブックを作成し、一時ファイルの内容をワークブックに入力することができます。最後に、実行中のExcelインスタンスを表示します。これにより、Excelがフォアグラウ

ンドに移動します。その後、データを日付や列の値などで並べ替えて、たとえば、各サーバーがコアの管理対象になった日付を知ることができます。次の図を参照してください。

New-ObjectコマンドレットによるMicrosoft Excelの起動

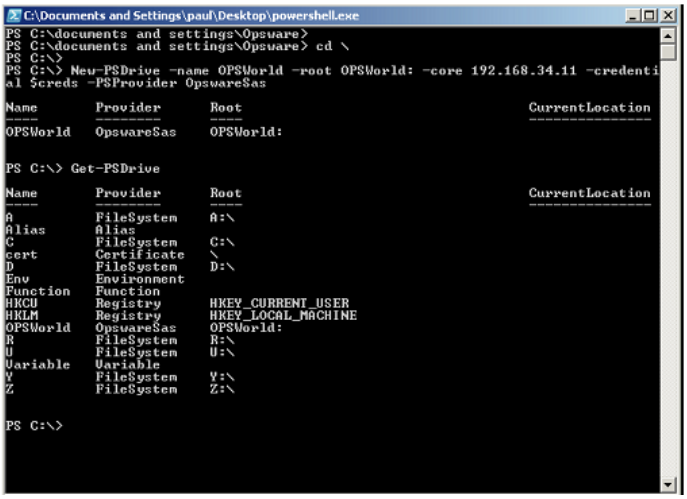


シナリオ3

このシナリオでは、SAコアをWindows PowerShell PSDriveとしてマウントし、SAのSA **Jobs**フォルダーを表示して、その内容を読み取る方法を示します。

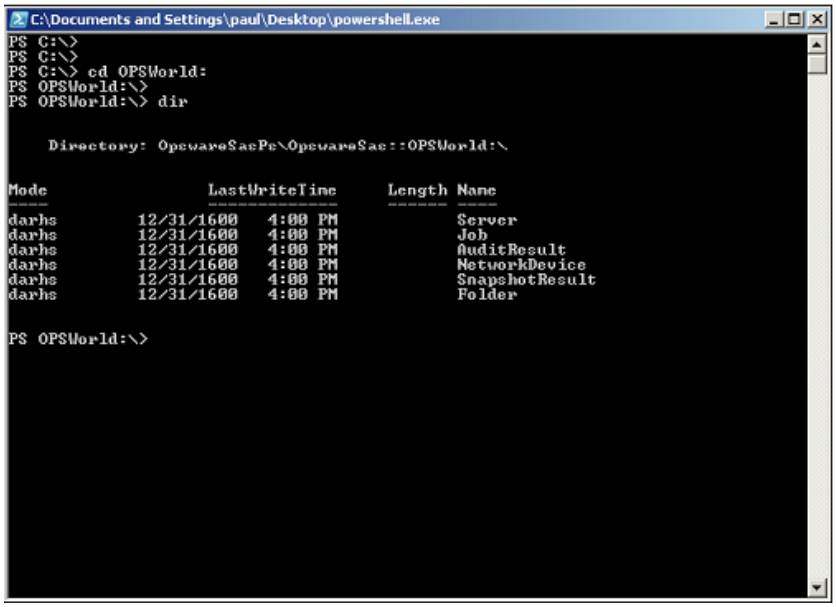
1. SAコアをWindows PowerShell PSDriveとしてマウントします。PowerShellでは、さまざまなデータストアやリポジトリを、ファイルシステムと同じ方法で操作できます。このシナリオでは、SAコア(具体的には管理対象環境データストア)を「マウント」して、OPSWorldという名前のドライブとして操作できるようにします。この仮想ファイルシステムに対するデータの読み書きが行われるか、クライアントからファイルシステムのナビゲーションが実行されるたびに、Windows PowerShellベースシステムは、PowerShell SASプロバイダー(-PSProvider OpswareSas)を呼び出します。次の図を参照してください。

   SAコアをWindows PowerShell PSDriveとしてマウント



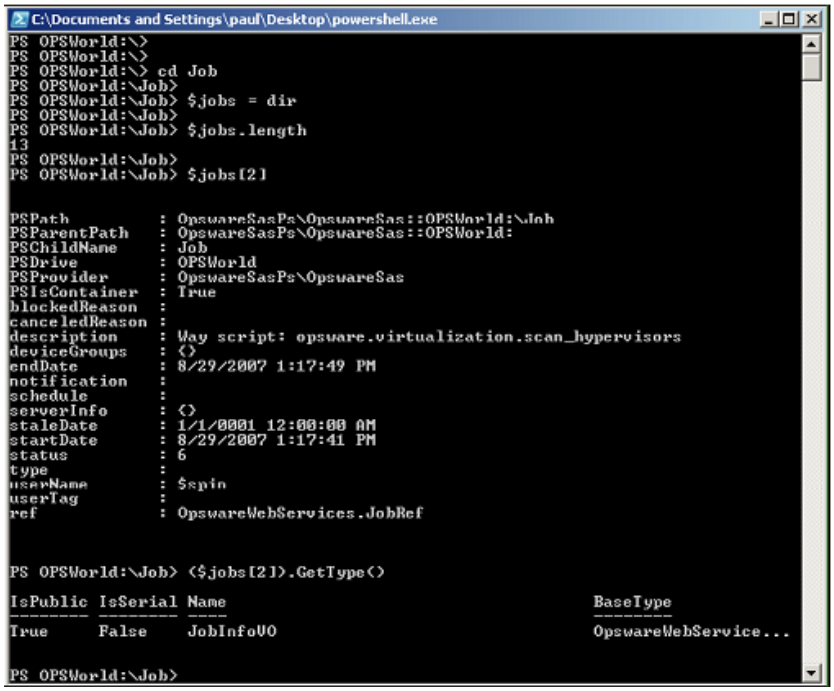
2. 新しくマウントしたドライブにディレクトリを変更し、ディレクトリリストを取得します。dirはGet-ChildItemコマンドレットに対するPowerShellエイリアスです。次の図を参照してください。

Get-Childコマンドレットの別名であるDIR



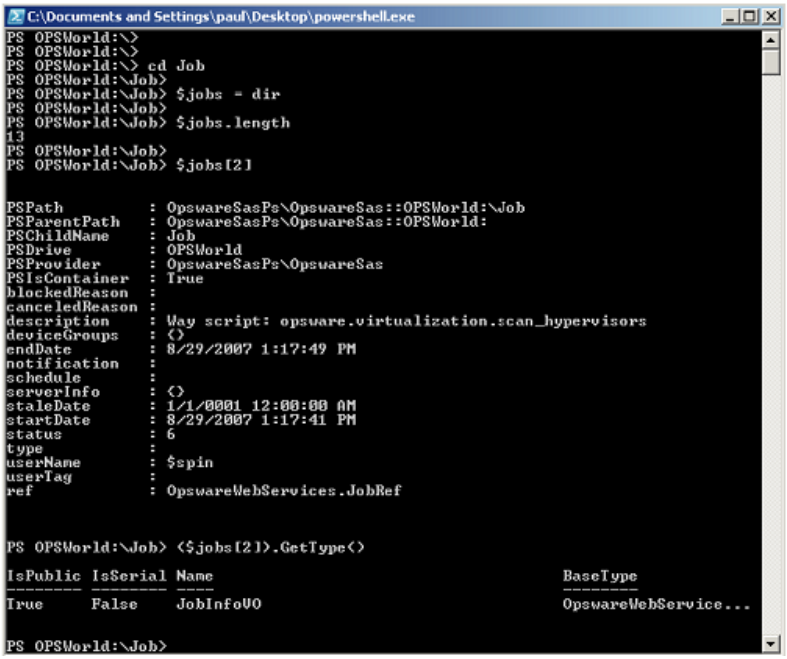
3. **Jobs**フォルダーに移動し、ディレクトリリストを取得し、ディレクトリリストをシェル変数に保存します。コアから取得されたJobInfoVOオブジェクトの配列がシェル変数に格納され、インデックスを使用してアクセスできます。

ディレクトリリストのPowerShell変数への保存



4. C:ドライブに移動し、OPSWorld PSDrivexを削除します。

OPSWorld PSDriveの削除

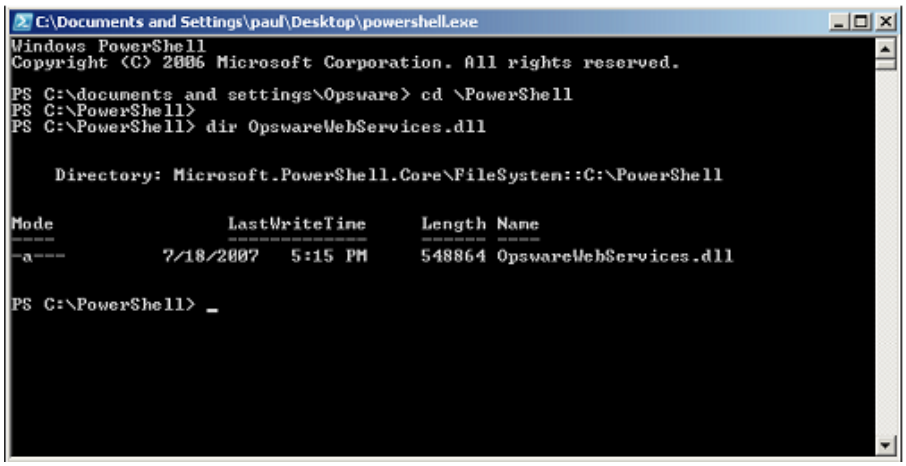


シナリオ4

このシナリオは、Windows PowerShell環境で使用可能なすべてのタイプのSAオブジェクトを調べる方法を示します。

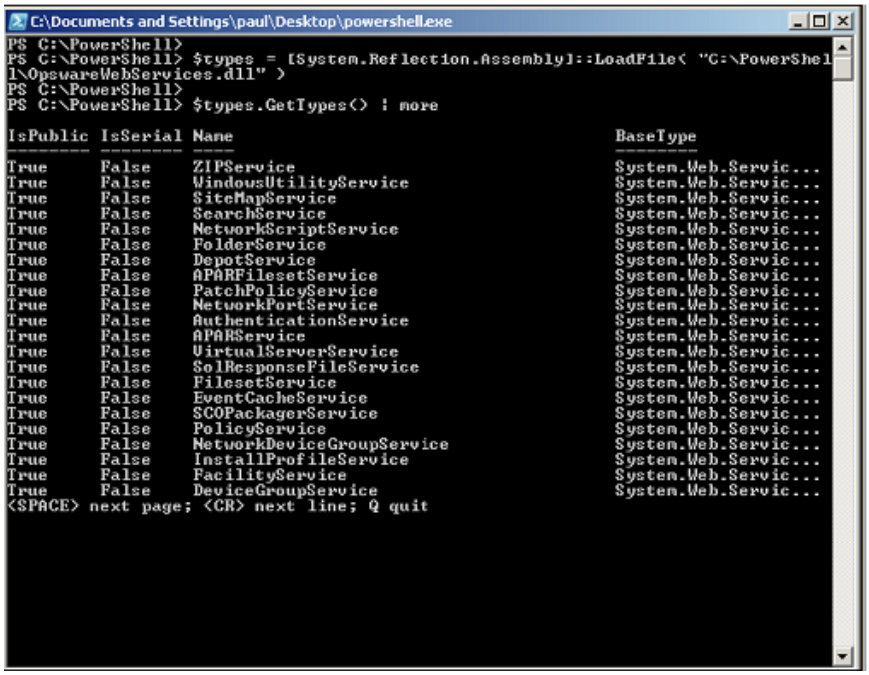
1. PowerShell SASプロバイダーとコマンドレットがある.NETアセンブリを見つけます。次の図を参照してください。

PowerShell SASプロバイダーとコマンドレットがある.NETアセンブリを見つける



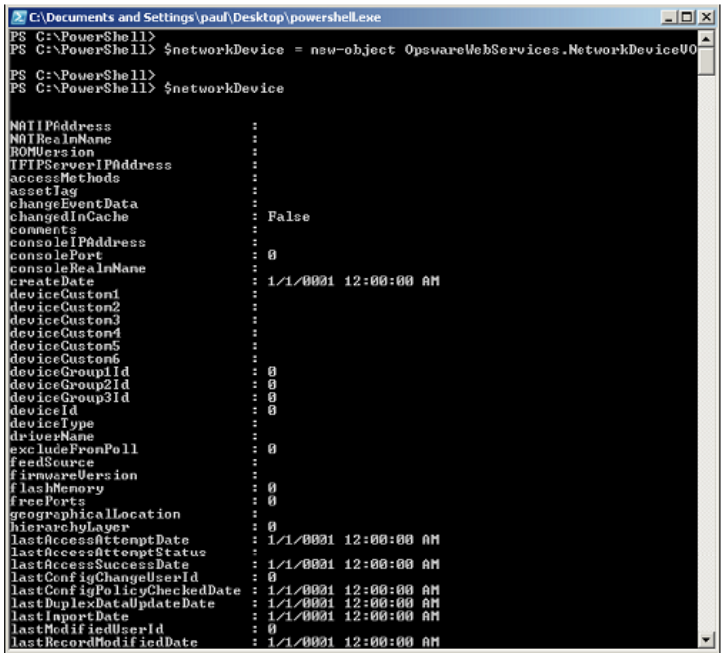
2. .NET Reflectionを使用して.NETアセンブリをロードし、ロードされたタイプを調べます。Windows PowerShell環境で使用可能なすべてのSAタイプが表示されます。次の図を参照してください。

.NETアセンブリをロードしてタイプを調べる



3. NetworkDeviceVOのインスタンスを作成します。これは作成直後のNetworkDeviceVOで、PowerShell環境でスクリプトやレポートの作成に使用できるネットワークデバイスのすべての属性を示します。次の図を参照してください。

NetworkDeviceVOのインスタンスの作成



Java RMIクライアント

Java Remote Invocation (RMI) クライアントを使えば、ネットワークを通じてSAコアにアクセスできるサーバーから、SA APIのメソッドを呼び出すことができます。クライアントを実行しているサーバーは、SAコアや管理対象サーバーでなくてもかまいません。クライアントは、コアに接続する際に、エンドユーザーがSAクライアントにログオンする場合と同様に、SAのユーザー名とパスワードを指定します。ユーザーが属するグループによって、クライアントから使用できるSAリソースとタスクが決まります。

このトピックの内容は、SAの基礎とJavaプログラミング言語に関する知識があるソフトウェア開発者を対象としています。

Java RMIクライアントの設定

SA API用のJava RMIクライアントを開発する前に、次の手順を実行します。

1. SAコアを開発環境にインストールします。プロダクションコアは使用しないでください。
2. Java RMIクライアントを構築して実行する開発サーバーを用意します。
3. 開発サーバーにJava SE 7 SDKをインストールします。
4. 開発サーバーから、OCCコンポーネントを実行しているSAコアサーバーへのネットワーク接続が可能であることを確認します。
5. SAコアサーバーから開発サーバーに`opswclient.jar`ファイルをダウンロードします。`opswclient.jar`ファイルには、SA API用のJava RMIスタブが含まれています。Java RMIクライアントをコンパイルして実行する際に、`opswclient.jar`を`classpath`オプションに指定します。
6. `opswclient.jar`をダウンロードするには、次のいずれかの操作を実行します。
   a. 次のURLを指定します。ここで、OCCホストはOCCコンポーネントを実行しているコアサーバーです。

      ```
      https://occ_host/twister/opswclient.jar
      ```

   b. 次のディレクトリに移動します。`/opt/opsware/twist/extlib/client`

また、`spinclient-latest.jar`ファイルと`opsware_common-latest.jar`ファイルも必要です。このファイルは、次のディレクトリで稼働しているSAコアから取得できます。

```
/opt/opsware/twist/lib/
```

さらに、このサンプルをコンパイルして実行するには、.jarファイルをclasspathパラメーターに追加する必要があります。

Java RMIの例

ここでは、GetServerInfoという名前の単純なJava RMIクライアントを示します。

GetServerInfoクライアントは、コマンドライン引数に指定したホスト名の全部または一部から、管理対象サーバーを検索します。見つかった管理対象サーバーのそれぞれに対して、クライアントはサーバー名、管理IPアドレス、OSバージョンを出力します。

GetServerInfoクライアントは、次のステップを実行します。

1. SAに接続します。
```
// Set the JNDI provider
client.setJNDIProvider( "https" , host , ( short ) 1032 , null, newString[] {
OPSWARE_CA_CRT_PATH } , null ) ;
// Force a reconnection
client.getContext( true ) ;
// Set and authenticate the user
client.setAPIUser( new APIUserImpl(username , password)) ;
```
2. ServerServiceインタフェースへの参照を取得します。
```
serverSvc = (ServerService)OpswareClient.getService
(ServerService.class);
```
3. ServerServiceのメソッドを呼び出します。
```
ServerRef[] serverRefs = serverSvc.findServerRefs(filter);
. . .
ServerVO[] serverVOs = serverSvc.getServerVOs(serverRefs);
. . .
System.out.println(serverVOs[i].getName());
```

GetServerInfoの例のコンパイルと実行

例をコンパイルして実行する前に、次の作業を実行します。

1. opsware_common-latest.jar、spinclient-latest.jar、opswclient.jarの各ファイルを取得します (Java RMIクライアントの設定を参照してください)。
2. デモプログラムGetServerInfo.javaファイルを含むZIPファイルをダウンロードします。

3. クライアントをコンパイルするには、opsware_common-latest.jar、spinclient-latest.jar、opswclient.jarの各ファイルをclasspathパラメーターで指定します。

   ```
   javac -classpath :path/opswclient.jar:path/opsware_common-
   latest.jar:path/spinclient-latest.jar GetServerInfo.java
   ```

4. クライアントを実行するには、次のコマンドを入力します。ターゲットには、SAで管理するサーバーの名前の全部または一部を指定します。注: WindowsでのJava classpathの区切り文字は";"です。

   ```
   java -classpath .:path/opswclient.jar:path/opsware_common-
   latest.jar:path/spinclient-latest.jar \GetServerInfo [オプション]     ターゲット
   ```

   次の例で、GetServerInfoはホストc44 (OCCコアコンポーネントが実行されているホスト) のポート443でSAに接続します。プログラムは、ホスト名に文字列opswを含む管理対象サーバーの情報を表示します。

   ```
   java -classpath .:/home/jdoe/opswclient.jar:/home/jdoe/opsware_common-
   latest.jar:/home/jdoe/spinclient-latest.jar \GetServerInfo --host
   c44.dev.example.com --port 443 opsw
   ```

5. プロンプトに応じてSAのユーザー名とパスワードを入力します。SAユーザーは、コマンドラインに指定されたターゲットに一致するサーバーに対する読み取りアクセス権を持つ必要があります。

Webサービスクライアント

Webサービスクライアントの概要

SA APIはWebサービスをサポートします。Webサービスとは、SOAP (Simple Object Access Protocol) やWSDL (Web Services Definition Language) などのオープンな業界標準に基づくプログラミング環境です。Webサービスクライアントは、PerlやC#などのさまざまなプログラミング言語を使用して (これについてはこの項の後の方で説明します)、またはMicrosoft Visual Studio .NETなどのWebサービス対応の開発環境を使用して作成できます。

このトピックの内容は、SAの基礎とWebサービス開発に関する知識があるソフトウェア開発者を対象としています。

このリリースで提供されているプログラム言語のバインド

SAのこのリリースには、C#用のWebサービスクライアントスタブが付属しています。Perlで作成したWebサービスクライアントにはクライアントスタブは不要です。

このリリースには、JavaおよびPython用のWebサービスクライアントスタブは付属していません。ただし、前の各項で説明したように、JavaクライアントはRMIを通じて、PythonクライアントはPytwistを通じてSA APIにアクセスできます。

サービスの場所とWSDLのURL

クライアントは次の構文のURLでWebサービスにアクセスします。ここで、ホストはOCCコアコンポーネントを実行しているサーバーであり、ポートはHTTPSプロキシのポートです(デフォルトのプロキシポートは443です)。パッケージ名は、サービスが属するJavaライブラリに対応します。

```
https://host:port/osapi/packageName/WebServiceName
```

WSDLファイルは、次の構文のURLにあります。

```
https://host:port/osapi/packageName/WebServiceName?WSDL
```

たとえば、次のURLはFolderServiceの場所とWSDLを表します。

```
https://occ.c38.example.com:443/osapi/com/opsware/folder/FolderService
```

```
https://occ.c39.example.com:443/osapi/com/opsware/folder/FolderService?wsdl
```

SOAPバインドスタイルはRPC (リモートプロシージャーコール) であり、トランスポートプロトコルはHTTPSです。

Webサービスクライアントのセキュリティ

SA APIの他のクライアントと同様、WebサービスクライアントはSAの操作を実行するために認証され、承認される必要があります。クライアントとSAコア内のWebサービスコンポーネントとの間の通信は暗号化されます。アクセスは、OCCコアコンポーネントのHTTPSプロキシポートを通じて接続するHTTPSクライアントに限定されています(デフォルトのポートは443です)。

オーバーロードされた操作

SA APIにはオーバーロードされた操作がありますが、WSDL 2.0仕様はオーバーロードをサポートしません。SA APIのオーバーロードされた操作は、Webサービスからは単一の操作として公開されます。

Javaインタフェースのサポート

SA APIはJavaインタフェースを使用しますが、Webサービスはインタフェースをサポートしません。回避策として、WSDLファイルはインタフェースを`xsd:anyType`にマッピングします。C#などのオブジェクト指向プログラミング言語でクライアントをコーディングする場合、インタフェースを返すAPIメソッドの戻り値型は、具象クラスにキャストする必要があります。インタフェースの配列は`Object[]`に変換されます。配列メンバーの固有の型は、シリアル化/逆シリアル化の際に保持されます。C#コードの例については、「インタフェース戻り値型の処理」(184ページ)を参照してください。

サポートされないデータ型

SA APIで使用されている次のデータ型は、SOAPではサポートされません。

```
java.util.Properties
com.opsware.common.ModifiableMap
```

```
com.opsware.acm.ValueSet
com.opsware.swmgmt.PolicyOverrideFilter
```

Webサービスから除外されたメソッド

次のSA APIメソッドは、サポートされないデータ型をパラメーターまたは戻り値型として使用します。このため、これらはWebサービスの操作としては公開されていません。

```
com.opsware.custattr.CustomAttribute.getCustAttrs
com.opsware.custattr.CustomAttribute.setCustAttrs
com.opsware.custattr.CustomField.getCustomFields
com.opsware.custattr.CustomField.setCustomFields
com.opsware.pkg.Patch.getPolicyOverrideRefs
```

java.util.Mapの部分的サポート

Axisは`java.util.Map`を、キーと値のペアの集合である`apachesoap:Map`に変換します。.NETでは、この変換は動作しません。たとえば、C#クライアントは、キーと値のペアの空の配列を受け取ります。一方、PerlのSoap::Liteではこの変換が動作します。したがって、`java.util.Map`を使用するSA APIは、Webサービスの操作として使用できます。

次のメソッドは、`java.util.Map`をパラメーターまたは戻り値型として使用します。

```
com.opsware.acm.GroupConfigurable.getApplicationInstances
com.opsware.acm.ServerConfigurable.getCustAttrsWithRC
com.opsware.compliance.sco.CMLSnapshot.getValueSet
com.opsware.compliance.sco.CMLSnapshot.setValueSet
com.opsware.compliance.sco.SnapshotResultService.remediateCMLSnapshot
com.opsware.custattr.VirtualColumnVO.getConfigInfo
com.opsware.custattr.VirtualColumnVO.setConfigInfo
```

サポートされないデータ型を使用するVOのメソッド

次のVOのメソッドは、サポートされないデータ型をパラメーターまたは戻り値型として使用します。

```
com.opsware.acm.ApplicationInstanceVO.getValueset
com.opsware.acm.ApplicationInstanceVO.setValueset
com.opsware.acm.ConfigurableVO.getValueset
com.opsware.acm.ConfigurableVO.setValueset
com.opsware.virtualization.VirtualConfigNode.getProperties
com.opsware.virtualization.VirtualConfigNode.setProperties
com.opsware.virtualization.VirtualServerConfig.getProperties
com.opsware.virtualization.VirtualServerConfig.setProperties
```

VOの作成または更新の際のsetDirtyAtrributesの呼び出し

Webサービスクライアントは、サービスに対してcreateまたはupdateメソッドを呼び出す前に、setDirtyAttributesを呼び出す必要があります。setDirtyAttributesメソッドは、createまたはupdateの呼び出しによって設定する必要があるVOの属性 (フィールド) を明示的にマークします。setDirtyAttributesで指定される属性名は、大文字と小文字が区別されます。

たとえば、FolderVOオブジェクトのdescription属性を変更する場合、次のコードのようにupdateを呼び出す前にsetDirtyAttributesを呼び出します。

```
// fs is FolderService
FolderVO folderVO = fs.getFolderVO(folderRef);
folderVO.setDescription("credit card processing");
folderVO.setDirtyAttributes(new String[]{"description"});
fs.update(folderRef, folderVO, true, true);
```

WebサービスクライアントがsetDirtyAttributesを呼び出す必要があるのは、AxisがXMLからのXMLオブジェクトを逆シリアル化する方法に原因があります。setDirtyAttributesを呼び出さない場合、Axisは読み取り専用の属性を含めてVOのすべての属性に対してsetterを呼び出すため、ReadOnlyExceptionが発生します。

SA WebサービスAPI 2.2との互換性

SA WebサービスAPI 2.2は、本書で記述しているSA APIとは互換性がありません。メソッドのシグネチャ、サービス、WSDL、ポートバインドが異なります。新しくWebサービスクライアントを作成する場合は、SA WebサービスAPI 2.2ではなくSA APIを使用してください。

Perl Webサービスクライアント

ここでは、SA APIにアクセスするPerl Webサービスクライアントを作成するための詳細な手順とサンプルコードを示します。

Perlクライアントに必要なソフトウェア

開発環境には次のPerlモジュールが必要です。

- Crypt-SSLeay-0.57
- IO-Socket-SSL-1.31
- Net-SSLeay-1.35
- HTML-Parser-3.64
- MIME-Base64-3.08
- URI-1.40
- libwww-perl-5.833
- SOAP-Lite-0.710

Perlのバージョンによっては、新しいバージョンのモジュールが必要になる場合があります。

> **注意:** `"500 SSL negotiation failed:"` のエラーが解消されない場合は、OpenSSLをバージョン1.0.1以上に更新する必要があります。RHELファミリーの場合は、OpenSSLをバージョン1.0.1e-30以上に更新する必要があります。

Perlデモプログラムの実行

デモプログラムを実行するには、次の手順を実行します。

1. サポートサイトから、**All Manuals Download SA 10.5**のフォルダーでバンドルされた**SA_Platform_Developer_Guide_examples.zip**ファイルを取得します。
2. デモプログラムの`uapisample.pl`ファイルを`SA_Platform_Developer_Guide_examples.zip\SA_Platform_Developer_Guide_examples\api_examples\web_services\perl`から取得します。
3. `uapisample.pl`ファイルを編集し、コードに直接書き込まれている`host`、`username`、`password`、および`serverID`などのオブジェクトIDの値を変更します。
4. `uapisample.pl`を実行します。
5. 「`Certificate Verify Failed`」というエラーが発生する場合は、サンプルファイルの次の行をコメント解除し、証明書ファイルの有効なパスを指定します。

```
#$ENV{HTTPS_CA_FILE} = "path_to/opsware-ca.crt";
```

証明書ファイルは、SAコアの次の場所にあります。

```
/var/opt/opsware/crypto/twist/opsware-ca.crt
```

Perlサンプルコード

次のコードは、先ほどダウンロードしたZIPファイルに含まれるPerlプログラム`uapisample.pl`の一部です。

サービスURIの設定

```
# Construct the URI for the service.
#
my $username = "integration";
my $password = "integration";
my $protocol = "https";
my $host = "occ.c38.dev.example.com";
my $port = "443";
my $contextUri = "osapi/com/opsware/";
my $folderServiceName = "folder/FolderService";
my $folderUri = "http://www.example.com/" .$contextUri .
$folderServiceName;
# Create a proxy to the FolderService.
#
my $folderProxy = $protocol . "://". $username . ":". $password . "@" .
$host . ":". $port . "/" . $contextUri . $folderServiceName;
```

新しいサービスの開始

```
my $folderPort = SOAP::Lite
-> uri($folderUri)
-> proxy($folderProxy);
```

サービスメソッドの呼び出し

```
my $root = $folderPort->getRoot()->result();
print 'Got root folder: ' . $root->{'name'} . "\n";
# Alternative:
my $root = $folderPort->SOAP::getRoot();
print 'Got root folder: ' . $root->{'name'} . "\n";
```

VOの取得

```
$rootVO = $folderPort->getFolderVO(SOAP::Data->name('self')
->value(\SOAP::Data->name('id')->type('long')->value(0)))
->result();
```

```
# The preceding call to getFolderVO does not pass a FolderRef
# parameter.If a method such as FolderService.remove accepts a
# FolderRef parameter, use the following code:
#
my $folderToBeRemoved = SOAP::Data->name('self')
->attr({ 'xmlns:ns_fs' => 'http://folder.example.com/FolderService'}) -
>type('ns_fs:FolderRef')->value(\SOAP::Data->name('id')->type('long') -
>value(123456));
$folderPort->remove($folderToBeRemoved);
# To see the Perl representation of the returned VO, you can use
# the Dumper method.This will help you understand how to
# construct the dirty attributes of a VO for a create or update
# method.
#
use Data::Dumper;
print Dumper($folderVO);
```

配列の取得

```
# Construct $folder, the FolderRef before getting the array.
#
my $folder = SOAP::Data->name('self') ->attr({ 'xmlns:ns_fs' => 'http://
folder.example.com'}) ->type('ns_fs:FolderRef')->value(\SOAP::Data-
>name('id')->type('long') ->value($root->{'id'}));
# The getChildren method returns an array of FNodeReference
# objects.
#
my $children = $folderPort->getChildren($folder, SOAP::Data->name('type')-
>type('string')->value(''))->result();
foreach $child (@{$children}){
print 'Get child: ' . $child-{'name'} ."\n";
}
```

オブジェクト配列の構築

```
# For a function that takes an object array as a parameter,
# such the getVOs method, take the following approach:
#First, construct the Array object elements individually
# and put them in an array.
#
my @refs = [];
foreach my $ref (@{$myRefs}){
       # Assume myRefs was returned from a previous
               # Web Services call.
       my $object = SOAP::Data->name('FacilityRef')
               ->value(\SOAP::Data->name('id')
                       ->type('long')
                       ->value($ref->{'id'}
               )
       )->attr({ 'xmlns:facility' => 'http://locality.example.com'})
```

```
       ->type('facility:FacilityRef');
   push @refs, $object;
}
# Second, construct an Array Object and put the array in it.
#
my $selves = SOAP::Data->name("selves" =>\SOAP::Data->name("element" => @refs)-
>type("facility:FacilityRef"))
               ->attr({ 'xmlns:facility' => 'http://locality.example.com'})
               ->type("facility:ArrayOfFacilityRef");
```

VOの更新または作成

```
# This example updates the description attribute of a ServerVO.
#
my $serverID = 40038;
my $server = SOAP::Data->name('self')->value(\SOAP::Data->name('id')-
>type('long')->value($serverID));
# Don’t forget to set dirtyAttributes for the attributes
# you want to update.You also need dirtyAttributes for
# create methods that pass a VO.
#
my @dirtyAttrs = ('description');
my $serverVO = SOAP::Data->name('vo') ->attr({ 'xmlns:ns_ss' => 'http://
server.example.com'}) ->value(\SOAP::Data->value( SOAP::Data-
>name('description')->value('PERL_UPDATE_DESC')->type('string'), SOAP::Data-
>name('logChange')->value('false')->type('boolean'), SOAP::Data-
>name('dirtyAttributes' => \SOAP::Data->name("element" => @dirtyAttrs)-
>type("string")) ->type("ns_ss:ArrayOf_soapenc_string"), ));
my $force = SOAP::Data->name('force')->value('true')->type('boolean');
my $refetch = SOAP::Data->name('refetch')->value('true')->type('boolean');
# Call the update method.
#
print 'Invoking method serverWSPort.update...', "\n";
my $updatedServerVO = $serverWSPort->update(
               $server,
               $serverVO,
               $force,
               $refetch)->result();
print "New description: ", $updatedServerVO->{'description'}, "\n";
```

SOAPフォールトの処理

```
# Make sure that you turn off on_fault subroutine in the
# "use SOAP::Lite ..." statement.
#
# The fault member of a SOAP return will be set if the Web
# Service call throws an exception.
#The following code tries to get a folder that does not exist:
#
my $testVO = $folderPort->getFolderVO(SOAP::Data->name('self') -
```

```
>value(\SOAP::Data->name('id')->type('long')->value(123456)));
if($testVO->fault){
       print $testVO->faultstring ."\n";
       # This will print the error msg.
       print "ExceptionName: " . getExceptionName($testVO) ."\n"; # A
NotFoundException should be displayed here
       # The code that deals with the error goes here....
}
. . .
#The following subroutine extracts the exception name from the
# returned faultdetail.
#
sub getExceptionName {
       my $fault = shift; #get the fault object
       if($fault->faultdetail->{'fault'}){
               return ref($fault->faultdetail->{'fault'});
       }
}
. . .
#As shown in the preceding code, it's easier to handle SOAP
# faults if you execute functions like this:
#
# my $data = $port->function(...);
# Not like this:
#$port->SOAP::function(...);
# $port->function(...)->result;
```

Webサービス用のPerlオブジェクトの構築

Webサービス操作を呼び出す前に、Perlクライアントは入力パラメーターに必要なデータ構造を作成する必要があります。データ構造の作成に必要な情報は、APIドキュメント (javadocs) とサービスのWSDLファイルに記述されています。ここに示すPerlコードの例は、getServerVO操作に対する入力パラメーターの構築方法を示します。コードの後の手順は、APIドキュメントとWSDLファイルから入力パラメーターに関する情報を入手する方法を示します。

getServerVOを呼び出すためのソースコード

次のPerlコードは、入力パラメーターselfを設定してから、getServerVO操作を呼び出します。この呼び出しは、IDが12345の管理対象サーバーのVO (値オブジェクト) を取得します。

```
# Create a top-level SOAP::Data object
#
$self = SOAP::Data->name('self')
# The namespace corresponds to the schema of the data type
# of the SOAP:Data object.The name chosen (ns_ss) is
```

```
# arbitrary.
#
$self->attr({’xmlns:ns_ss =>
’http://server.example.com/ServerService’});
# Specify the type (ServerRef) for the parameter self, using the
# name of the namespace from the preceding statement.

#
$self->type(’ns_ss:ServerRef’);
# Create the value for the parameter.The value is a pointer
# to a SOAP::Data object.The number 12345 is the SA ID of a managed server.
#
my $id = SOAP::Data->name(’id’)->type(’long’)->value(12345);
# From the self object, point to the value.
#
$self->value(\$id);
# Finally, call getServerVO:
#
my $data = $serverPort->getServerVO($self);
if($data->fault){
       # Handle exceptions here ...
}
else{
       my $serverVO = $data->result;
}
. . .
```

getServerVOの設定のための情報の入手方法

getServerVOの呼び出しのコードを作成するために必要な情報を入手するには、次の手順を実行します。

1. ブラウザーで、次のURLのAPIドキュメント (javadocs) にアクセスします。

   https://OCCホスト:1032/twister/docs/index.html

   OCCホストは、コマンドセンターコンポーネントを実行しているコアサーバーのIPアドレスまたはホスト名です (Twisterでメソッドを呼び出す手順については、「APIドキュメントとTwister」(37ページ)を参照してください)。

2. APIドキュメントを調べて、メソッドの入力パラメーターと戻り値を確認します。

   getServerVOメソッドは、インタフェースcom.opsware.server.ServerServiceで定義されています。次のメソッドシグネチャから、getServerVOがServerRefをパラメーターとして受け取り、ServerVOを返すことがわかります。

```
public ServerVO getServerVO(ServerRef self)
       throws java.rmi.RemoteException,
              NotFoundException,
              AuthorizationException
```

3. ブラウザーで次のURLを指定して、ServerServiceのWSDLファイルを開きます。

   ```
   https://occ_host/osapi/com/opsware/server/ServerService?wsdl
   ```

4. WSDLファイルで、ServerServiceの名前空間を見つけます。

   ```
   <schema targetNamespace="http://server.example.com"
   xmlns="http://www.w3.org/2001/XMLSchema">
   ```

   次のPerlステートメント(前記のコードリスト内に存在)が名前空間を指定しています。

   ```
   $self->attr({'xmlns:ns_ss =>
   'http://server.example.com/ServerService'});
   ```

5. WSDLファイルで、getServerVO操作を見つけ、入力メッセージ名getServerVORequestを確認します。

   ```
   <wsdl:operation name="getServerVO" parameterOrder="self">
          <wsdl:input message="impl:getServerVORequest" name="getServerVORequest"/>
          <wsdl:output message="impl:getServerVOResponse" name="getServerVOResponse"/
   >
          <wsdl:fault message="impl:NotFoundException" name="NotFoundException"/>
          <wsdl:fault message="impl:AuthorizationException"
   name="AuthorizationException"/>
   </wsdl:operation>
   ```

6. WSDLファイルで、getServerVORequestメッセージを見つけます。

   ```
   <wsdl:message name="getServerVORequest">
          <wsdl:part name="self" type="impl:ServerRef"/>
   </wsdl:message>
   ```

   getServerVORequestメッセージ要素は、getServerVOの入力パラメーターの名前(self)と型(ServerRef)を定義します。次のPerlステートメントは、ServerRefを指定します。

   ```
   $self->type('ns_ss:ServerRef');
   ```

7. WSDLファイルで、ServerRefのcomplexTypeを見つけます。

   ```
   <complexType name="ServerRef">
          <complexContent>
                 <extension base="tns1:ObjRef">
                        <sequence>
                               <element name="secureResourceTypeName" nillable="true"
   type="soapenc:string"/>
   ```

```
                        </sequence>
                    </extension>
                </complexContent>
        </complexType>
```

ServerRefはObjRefを拡張しています。

8. WSDLファイルで、ObjRefのcomplexTypeを見つけます。

```
<complexType abstract="true" name="ObjRef">
        <sequence>
                <element name="id" type="xsd:long"/>
                <element name="idAsLong" nillable="true" type="soapenc:long"/>
                <element name="name" nillable="true" type="soapenc:string"/>
        </sequence>
</complexType>
```

ObjRefで、名前 (id) と型 (long) を確認します。これらのデータ型は、次のPerlステートメントで指定されています。

```
my $id = SOAP::Data->name('id')->type('long')->value(12345);
```

C# Webサービスクライアント

ここでは、SA APIにアクセスするC# Webサービスクライアントを作成するための詳細な手順とサンプルコードを示します。

C#クライアントに必要なソフトウェア

C# Webサービスクライアントを開発するには、開発環境に次のソフトウェアが必要です。

- Microsoft .NET Framework SDKバージョン1.1
- SA API用C#クライアントスタブ

C#クライアントスタブの入手方法

SAには、各サービス用のスタブファイル (例、FolderService.csが用意されます。すべてのスタブは、同じ名前空間OpswareWebServicesを持ちます。スタブの他に、SAにはServerRefなどの共有クラスを含むshared.csファイルもあります。

C#スタブを含むZIPファイルをダウンロードするには、次のURLを指定します。ここで、OCCホストはOCCコンポーネントを実行しているコアサーバーです。

```
https://occ_host:1032/twister/opswcsharpclient.zip
```

サービスとオブジェクトで定義されている定数は、C#スタブでは定義されていません。定数に関する情報を得るには、APIドキュメント (javadocs) を使用します。詳細については「定数のフィールド値」(37ページ)を参照してください。

C#デモプログラムのビルド

デモプログラムをビルドするには、次の手順を実行します。

1. サポートサイトから、**All Manuals Download SA 10.5**のフォルダーでバンドルされた**SA_Platform_Developer_Guide_examples.zip**ファイルを取得します。

   SA_Platform_Developer_Guide_examples.zipには、`SA_Platform_Developer_Guide_examples\api_examples\web_services\csharp`に次のデモプログラムファイルが含まれています。

   - `App.config` - アプリケーション設定
   - `WebServicesDemo.cs` - サービスメソッドを呼び出すクライアントコード
   - `MyCertificateValidation.cs` - 証明書検証クラス
2. 次のディレクトリを作成します。
   C:\wsapi
3. Visual Studio 2008スタートページで、[新規プロジェクト] を選択し、次の値を持つプロジェクトを作成します。
   - プロジェクトタイプ: Visual C#プロジェクト
   - テンプレート: コンソールアプリケーション
   - 名前: WSAPIDemo
   - 場所: `C:\wsapi`

   この操作により、新しいディレクトリ`C:\wsapi\WSAPIDemo`が作成され、その下にいくつかのファイルが置かれます。
4. 新しいプロジェクトで、オブジェクトリストからデフォルトの`program`ファイルと`AssemblyInfo.cs`ファイルを削除します。
5. 手順1で入手したファイルを`C:\wsapi\WSAPIDemo`ディレクトリにコピーします。

6. 「C#クライアントスタブの入手方法」(181ページ)で指定したURLからクライアントスタブをダウンロードします。
7. C#クライアントスタブをC:\wsapi\WSAPIDemoディレクトリにコピーします。
8. 前の2つのステップでコピーしたファイルをWSAPIDemoプロジェクトに追加します。
   - Visual Studioで、[プロジェクト] メニューから [既存項目の追加] を選択します。
   - C:\wsapi\WSAPIDemoディレクトリに移動し、デモファイルをすべて選択します (.csと.config)。
9. System.Web.Services.dllへの参照を追加します。
   - Visual Studioで、[プロジェクト] メニューから [参照の追加] を選択します。
   - .NETタグの下で、System.Web.Services.dllという名前のコンポーネントを参照します。
   - System.Web.Services.dllをクリックし、[選択]、[OK] をクリックします。
10. プロジェクトの作成時に別のテンプレートを使用した場合、System、System.XML、System.Dataへの参照を追加することが必要な場合があります。[プロジェクトの参照] をチェックして、これらの参照を追加する必要があるかどうかを判断します。
11. App.configファイルで、username、password、hostの値と、serverIDなどの直接書き込まれているオブジェクトIDを変更します。
12. Visual Studioで、[ビルド] メニューから [WSAPIDemoのビルド] を選択します。

> **注意:** ターゲットコアでTLSv1.xの最小プロトコルバージョンが実行されている場合は、Powershellバージョン (バインドされた.NET Frameworkのバージョン) がこれをサポートしている必要があります。詳細については、https://msdn.microsoft.com/en-us/library/system.net.securityprotocoltype(v=vs.110).aspxを参照してください。
>
> また、C#アプリケーションで明示的に有効化する必要があります。
>
> サンプルコード: `System.Net.ServicePointManager.SecurityProtocol |= System.Net.SecurityProtocolType.Tls12 | System.Net.SecurityProtocolType.Tls11;"`

C#デモプログラムの実行

デモプログラムを実行するには、次の手順を実行します。

1. Visual Studio 2008のコマンドプロンプトを開きます。

   [スタート] > [すべてのプログラム] > [Microsoft Visual Studio 2008] > [Visual Studio ツール] > [Visual Studio 2008コマンドプロンプト]

2. 次のディレクトリに移動します。

   ```
   C:\wsapi\WSAPIDemo\bin\Debug
   ```

3. 次のコマンドを入力します。

   ```
   WSAPIDemo.exe
   ```

C#サンプルコード

次のコードは、先ほどダウンロードしたZIPファイルに含まれるC#プログラムWebServicesDemo.csの一部です。

証明書処理の設定

```
# This setup is required just once for the client.
#
ServicePointManager.CertificatePolicy = new MyCertificateValidation();
```

URLプレフィックスの割り当て

```
# This is the URL prefix for all services.
#
wsdlUrlPrefix = protocol + "://"+ host + ":"+ port + "/" + contextUri + "/";
```

サービスの開始

```
FolderService fs = new FolderService();
fs.Url = wsdlUrlPrefix + "com.opsware.folder/FolderService";
```

サービスメソッドの呼び出し

```
FolderRef root = fs.getRoot();
FolderVO vo = fs.getFolderVO(root);
```

インタフェース戻り値型の処理

```
            # In the API, FolderVO.getMembers returns an array of
# FNodeReference interfaces, but Web Services does not support
# interfaces.In the C# stub, the return type of
# FolderVO.members is Object[].If a returned Object type will
# be used as a parameter that must be a specific type, then you
# must cast it to that type.For example, the following code
# casts elements of the returned array to FolderRef as
# appropriate.
```

```
#
Object[] members = vo.members;
for(int i=0;i<members.Length;i++)
{
Console.WriteLine("Got object: " + members[i].GetType().FullName + " --> " +
((ObjRef)members[i]).name);
       if(members[i] is FolderRef) {
       Console.WriteLine("I am a FolderRef: " +
       ((FolderRef)members[i]).name);
       }
}
```

VOの更新または作成

```
            # When updating a VO, the changed attributes must be set in
# dirtyAttributes.(The VO passed to a create method has
# the same requirement.)
#
# Note: If you update a VO that was returned from a service
# method invocation, such as getFolderVO, then you must
# set the logChange attribute of the VO to false:
# vo.logChange = false;
#
# The following code changes the name of a folder.
#
Console.WriteLine("Changing name from " + vo.name +
" to yo_csharp.");
vo.name = "yo_csharp";
vo.dirtyAttributes = new String[]{"name"};
# Manually set dirty fields being changed.
#
vo = fs.update(folder, vo, true, true);
Console.WriteLine("Folder name changed to: " + vo.name);
```

例外の処理

```
            # .NET converts Web Services faults into SoapExceptions
# without trying to deserialize them into application
# exceptions first.As a result, your code cannot catch
# application exceptions.As a workaround, the C# stubs
# provided by SA include SOAPExceptionParser,
# a class that enables you to get information from
# SOAPExceptions.The following code shows how to get the
# exception name and error message by calling the getDetail
# method of SOAPExceptionParser.
#
try{
// Try to get a non-existent folder here.
```

```
} catch(SoapException e){
       SoapExceptionDetail detail =
       SoapExceptionParser.getDetail(e);
       Console.WriteLine("SoapExceptionDetail.name: " +
       detail.exceptionName);
       Console.WriteLine("SoapExceptionDetail.msg: " +
       detail.message);
...
}
```

C#でのパスワードセキュリティ

FolderServiceメソッドは、ユーザー名とパスワードのペアをApp.configファイルから読み取ります。このメソッドの例を次に示します。

```
            User user = new User();
user.username = "user";
user.password = "password";
FolderService fs = new FolderService();
fs.Url = wsdlUrlPrefix + "com.opsware.folder/FolderService";
fs.user = user;
```

パスワードを平文でApp.configファイルに記録したくない場合は、SecureUserクラスを使用してパスワードを暗号化できます。SecureUserクラスは、.NET 2.0のC# SecureStringを使用します。パスワードは暗号化されてSecureStringに記憶されます。また、getPassword()メソッドは内部からしか見えません。SecureUserは静的クラスなので、ユーザー名とパスワードの設定は1回だけ、あるいはユーザーを切り替えたときだけ行えば済みます。

各サービスは、ユーザー名とパスワードをまずSecureUserから取得し、その後に過去との互換性のためにuserメンバー変数、その後にApp.configから取得します。SecureUserは、パスワードをStringまたはSecureStringで受け取ります。どちらの場合も、クライアントはSecureUser.setUser()メソッドに渡されたパスワード変数をクリーンアップする責任があります。

パスワードはいずれかの時点でメモリ上の通常のC#文字列に変換する必要があります。これは次のガベージコレクションが発生するまで解放されません。SecureUserで保証されるのは、内部的なパスワードの記憶がセキュアであることだけです。

次の例は、ユーザー名とパスワードをセキュアに設定する方法を示します。

```
            SecureString passwd = new SecureString();
passwd.AppendChar('p');
passwd.AppendChar('a');
passwd.AppendChar('s');
```

```
passwd.AppendChar('s');
passwd.AppendChar('w');
passwd.AppendChar('d');
SecureUser.setUser("username", passwd); // that's it, no need to set up user
for each service.
passwd.Dispose(); // resets passwd and frees up memory so no copy remains from
caller.
```

プラグ可能チェック

SAの監査と修復機能では、SA管理対象サーバーに対するコンプライアンス情報の定義と監視が可能です。コンプライアンス標準は常に進化し続けるため、SAでは、特殊化されたカスタムチェックやポリシーを作成したり、SAに付属するチェックやポリシーを拡張したりできます。プラグ可能チェックとは、1つまたは複数の監査ポリシーに属する監査ルールです。コマンドライン環境でプラグ可能チェックを作成し、チェックをアップロードしてから、SAクライアントで監査ポリシーに追加します。

この項の内容は、XMLと、SAの監査と修復機能に関する知識があるソフトウェア開発者を対象としています。

プラグ可能チェックの設定

プラグ可能チェックを開発する前に、次の手順を実行します。

1. SAコアを開発環境にインストールします。プロダクションコアは使用しないでください。
2. エージェントがインストールされているサーバーに、OCLI 1.0をインストールします。OCLI 1.0の詳細については、使用を参照してください。

プラグ可能チェックのチュートリアル

このチュートリアルでは、HelloWorld Checkという名前のプラグ可能チェックを作成する方法を示します。この単純なチェックは、/var/tmp/helloworldファイルがUnix管理対象サーバーに存在することを確認します。このファイルが存在しない場合、プラグ可能チェックの修復スクリプトによってファイルが作成されます。

HelloWorld Checkを開発するには、次の手順を実行します。

1. プラグ可能チェックの設定の手順を実行します。OCLI 1.0をインストールするサーバーが、このチュートリアルの開発サーバーになります。
2. HelloWorld Checkのサンプルコードは、APIコードの例を収録したZIPファイルに含まれています。
3. 前のステップでダウンロードしたファイルを展開し、pluggable_checks/helloworldディレクトリに次のファイルがあることを確認します。
   - config.xml
   - gethelloworld.py
   - sethelloworld.py

   HelloWorld Checkは、これら3つのファイルから構成されます。config.xmlファイルは構成ファイルです。gethelloworld.pyは、監査を実行するPythonスクリプトです。sethelloworld.pyは、修復を実行するPythonスクリプトです。次の手順では、これらのファイルをZIPファイルにパッケージ化し、このZIPファイルをSAにインポートします。
4. 開発サーバーで、展開されたhelloworldファイルを次の例のように作業ディレクトリにコピーします。

   ```
   cd /home/jdoe/dev
   mkdir helloworld
   cd helloworld
   cp 展開先/pluggable_checks/helloworld/* .
   ```
5. グローバルに一意のID (GUID) を取得します。各プラグ可能チェックにGUIDが必要です。有効なGUIDを得るには、次のいずれかの方法を使用します。
   - 次のようなWebサイトにログオンします。
     http://kruithof.xs4all.nl/uuid/uuidgen（英語サイト）
   - 次のサイトから無料のWindowsツールguidgenをダウンロードします。
     http://www.microsoft.com/downloads/details.aspx?FamilyID=94551F58-484F-4A8C-BB39-ADB270833AFC&displaylang=en（英語サイト）

GUIDをプログラムで作成する場合、コードはRFC4122 (http://www.ietf.org/rfc/rfc4122.txt) (英語サイト) に準拠する必要があります。

6. テキストエディターで、次のようにconfig.xmlファイルにGUIDを挿入します。

```
<checkGUID>6c7ed38c-d8d6-11db-8314-0800200c9a66</checkGUID>
```

このチュートリアルで変更するconfig.xmlの要素はこれだけです。

7. テキストエディターで、GUIDを変更したconfig.xmlを保存します。

テキストエディターは開いたままにしておきます。このチュートリアル全体を通じて、config.xmlのさまざまな要素を参照することにより、HelloWorld CheckのPythonスクリプトとSAクライアント表示フィールドに要素がどのように対応するかを調べます。

8. config.xmlファイルで、次の要素を確認します。これらはHelloWorld Checkの監査 (get) および修復 (set) スクリプトに対応します。

```
<!-- The name of the script that performs the check.-->
<checkGetScriptName>gethelloworld.py</checkGetScriptName>

<!-- The name of the script that remediates the audit.-->
<checkSetScriptName>sethelloworld.py</checkSetScriptName>

<!-- The exit code of the gethelloworld.py script will be checked.-->
<checkReturnType>EXITCODE</checkReturnType>

<!-- A string argument is passed to gethelloworld.py.-->
<checkGetArgumentType>STRING</checkGetArgumentType>

<!-- The default argument for gethelloworld.py is the name of the file the
script is checking for.-->
<checkGetArgumentDefaultValue>/var/tmp/helloworld
</checkGetArgumentDefaultValue>

<!-- If the helloworld file exists, the exit code of gethelloworld.py is 0.
-->
<checkSuccessExitCodeValue>0</checkSuccessExitCodeValue>
<!-- If the helloworld file does not exist, the exit code of
gethelloworld.py is 1.-->
<checkSuccessExitCodeValue>1</checkSuccessExitCodeValue>
```

9. gethelloworld.pyスクリプトを調べます。これはファイル/var/tmp/helloworldの存在をチェックすることで監査を実行します。このチュートリアルでは、このスクリプトを編集する必要はありません。このチュートリアルの後の方 (手順30) で、SAクライアントで監査を実行するときに、このスクリプトが管理

対象サーバー上で実行されます。

config.xmlのcheckGetArgumentDefaultValueの値が示すように、このスクリプトのデフォルト引数は文字列/var/tmp/helloworldです。スクリプトの終了コード(result)は、<checkSuccessExitCodes>に指定された値に対応します。

gethelloworld.pyスクリプトのソースコードを次に示します。

```
import sys
import os
import string
if __name__ == "__main__":
       if len(sys.argv) != 2:
              sys.stderr.write("No argument found!Please enter a
                     file name!\n")
              sys.exit(220)
filename = sys.argv[1]
if os.path.isfile(filename) or os.path.isdir(filename):
       result = 0
else:
       result = 1
sys.stderr.write("Debugging: Found result %s\n"
% result)
sys.stdout.write("%s\n" % result)
sys.exit(result)
```

10. 次に、修復スクリプトsethelloworld.pyを調べます。これは/var/tmp/helloworldファイルを作成します。このスクリプトは、手順35で監査を修復する場合に、管理対象サーバーで実行されます。このチュートリアルではスクリプトは変更しません。

sethelloworld.pyのソースコードを次に示します。

```
import sys
import os
import string
if __name__ == "__main__":
       if len(sys.argv) != 2:

              sys.stderr.write("No argument found!
                     Please enter a file name!\n")
              sys.exit(220)
filename = sys.argv[1]
if os.path.isfile(filename) or os.path.isdir(filename):
       # Do nothing because the file already exists.
       pass
else:
       try:
```

```
                fd = open(filename, "w")
                fd.write(" ")
                fd.close()
        except:
                sys.stderr.write("Could not open file %s for
                        writing!\n" % filename)
                sys.exit(220)
# Exit successfully with a 0 exit code.
sys.stderr.write("Successfully created file\n")
sys.exit(0)
```

11. HelloWorld Checkをパッケージ化します。

    HelloWorldプラグ可能チェックをパッケージ化するには、作業ディレクトリの内容を1つのZIPファイルにアーカイブします。以下に例を示します。

    ```
    cd /home/jdoe/dev/helloworld
    zip ../helloworld.zip *
    ```

12. ZIPファイルに2つのPythonスクリプトと`config.xml`ファイルが含まれることを確認するために、次の`unzip`コマンドを入力します。

    ```
    unzip -t ../helloworld.zip
    testing: config.xml OK
    testing: gethelloworld.py OK
    testing: sethelloworld.py OK
    No errors detected in compressed data of ../helloworld.zip.
    ```

13. OCLI 1.0の`ouploadコマンド`で、プラグ可能チェックをSAにインポートします。

    ```
    oupload -C"Customer Independent" \
    -t"Server Configuration Check" \
    --forceoverwrite --old -O"SunOS 5.8" ../helloworld.zip
    ```

    > **注：** プラットフォームオプション (`-O`) は、UnixおよびLinuxチェックの場合はすべて`SunOS 5.8`です。Windowsチェックの場合、プラットフォームオプションは`Windows 2003`です。

    `oupload`が正常に実行されなかった場合、OCLI 1.0の正しいバージョンがインストールされており、PATH環境変数が正しく設定されており、`login`ファイルが環境に含まれていることを確認してください。これらの要件の詳細については、『ユーザーガイド』の「OCLI 1.0」を参照してください。

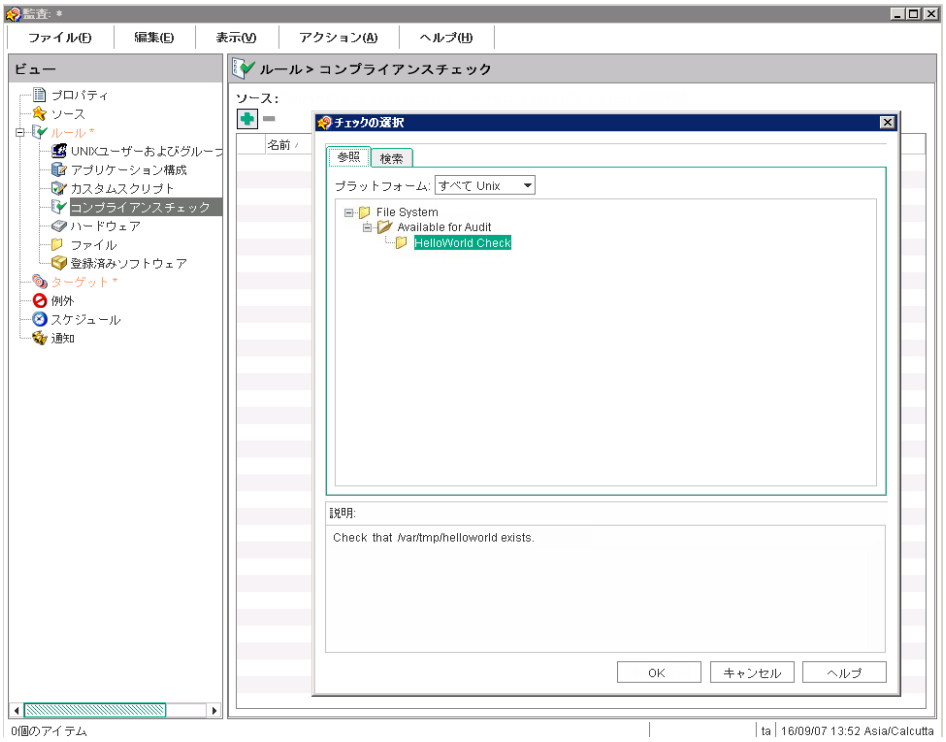
14. SAクライアントを開きます。

    この後の手順では、新しい監査を作成して、`oupload`コマンドでインポートしたHelloWorld Checkに追加します。

15. [**ツール**] メニューで [**キャッシュの更新**] を選択します。

16. [ナビゲーション] ペインで、[**ライブラリ**] > [**タイプ別**] > [**監査と修復**] > [**監査**] > [**Unix**] を選択します。
17. [**アクション**] メニューから [**新規**] を選択します。
18. [監査] ウィンドウの [プロパティ] ペインの [名前] フィールドに、「HelloWorld Audit」と入力します。
19. ビューペインで、[**ルール**] > [**ファイル**] を選択します。
20. [**追加**] ボタン  をクリックしてから、[**ファイルシステム**] をクリックします。
    次の図に示すように、内容ペインでHelloWorld Checkが [監査に使用可能] の下に表示されます。

    ファイルシステムに対するルール内のHelloWorld Check



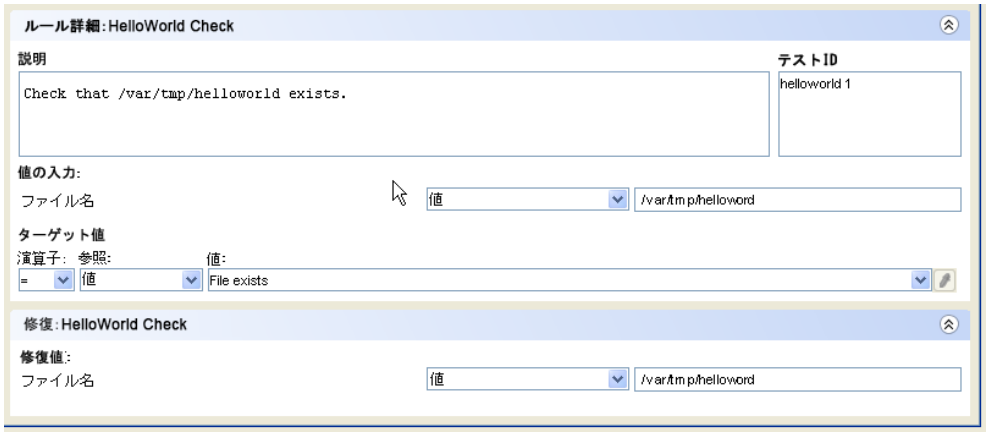
21. config.xmlファイルで、次の要素を確認します。これらは表示される情報に関連します。

```
<!-- The check name is the rule name shown in the SA Client.-->
<checkName>HelloWorld Check</checkName>

<!-- The category corresponds to the rule hierarchy displayed by the SA
Client.-->
<checkCategory>File System|My Custom Checks</checkCategory>
```

    SAクライアントの [監査] ウィンドウの [監査に使用可能] の下で、HelloWorld Checkを選択し、プラス記号をクリックします。

    次の図に示すように、内容ペインにHelloWorld Checkの詳細が表示されます。

HelloWorld Checkのルール詳細



22. config.xmlファイルで、次の要素を確認します。これらは「HelloWorld Checkのルール詳細」(195ページ)の図に表示される情報に関連します。

```
<!-- The following value appears under Description in the Rule Details of
the SA Client.-->
<checkDefaultDescription>
Check that /var/tmp/helloworld exists.
</checkDefaultDescription>

<!-- The following element corresponds to the Test ID in the SA Client.-->
<checkTestID>helloworld 1</checkTestID>

<!-- This label is under Input Values in the SA Client.-->
<checkGetArgumentDefaultLabel>File Name
</checkGetArgumentDefaultLabel>

<!-- The default argument to the gethelloworld.py script also appears
under Input Values in the SA Client.-->
<checkGetArgumentDefaultValue>/var/tmp/helloworld
</checkGetArgumentDefaultValue>
```

23. SAクライアントクライアントのビューペインで[ターゲット]を選択します。
24. 次のステップでは、HelloWorld Auditにターゲットサーバーを追加します。後の手順で、gethelloworld.pyスクリプトとsethelloworld.pyスクリプトがターゲットサーバーに対して実行されます。
25. 内容ペインで[**追加**]をクリックします。
26. [サーバーの選択]ウィンドウで、サーバーにドリルダウンして[**OK**]をクリックします。

[監査] ウィンドウで [**ファイル**] > [**保存**] を選択します。

27. この時点で、HelloWorld AuditはHelloWorld Check (ルール) を含み、ターゲットサーバーと関連付けられています。
28. [監査] ウィンドウで、[**アクション**] メニューから [**監査の実行**] を選択します。
29. [監査の実行] タスクのウィンドウを順次実行します。
30. [監査の実行] ウィンドウで [**ジョブの開始**] をクリックします。
このアクションは、ターゲットサーバーに対して`gethelloworld.py`スクリプトを実行するジョブを起動します。
31. ジョブが完了したら、[**結果の表示**] をクリックします。
32. [監査結果] ウィンドウの [ビュー] ペインで [ポリシールール (1)] を選択します。
33. [監査結果] ウィンドウの内容ペインでHelloWorld Checkを選択します。

次の図に示すように、[差異の詳細] ウィンドウが表示されます。

HelloWorld Checkの差異の詳細

34. `config.xml`ファイルで、次の要素を確認します。これらは「HelloWorld Checkの差異の詳細」(196ページ)に表示される情報に関連します。

```
<!-- The following value appears as the Policy Value in the Difference
Details window.-->
<checkSuccessExitCodeDefaultDisplayName>
File exists</checkSuccessExitCodeDefaultDisplayName>


<!-- The next value appears as the Actual Value in the same window.-->
<checkSuccessExitCodeDefaultDisplayName>
File does not exist</checkSuccessExitCodeDefaultDisplayName>
```

35. ターゲットサーバー上に`/var/tmp/helloworld`ファイルを作成するには、[差異] ウィンドウで [**修復**] をクリックします。

    このアクションは、`sethelloworld.py`スクリプトを実行します。詳細については、HPE SSOポータルで『Server Automation管理ガイド』を参照してください。

監査と修復

Sarbanes-Oxley (SoX) 法、Information Technology Infrastructure Library (ITIL)、ISO20000のために、サーバー構成のコンプライアンスを維持することが緊急の課題になっています。SAの監査と修復機能は、コンプライアンスの問題に対処するための整理されたポリシーのセットを提供します。グラフィカルインタフェースにより、指定したサーバーに対して簡単に監査を選択して実行し、プロフェッショナル標準にどの程度適合するかを判定できます。

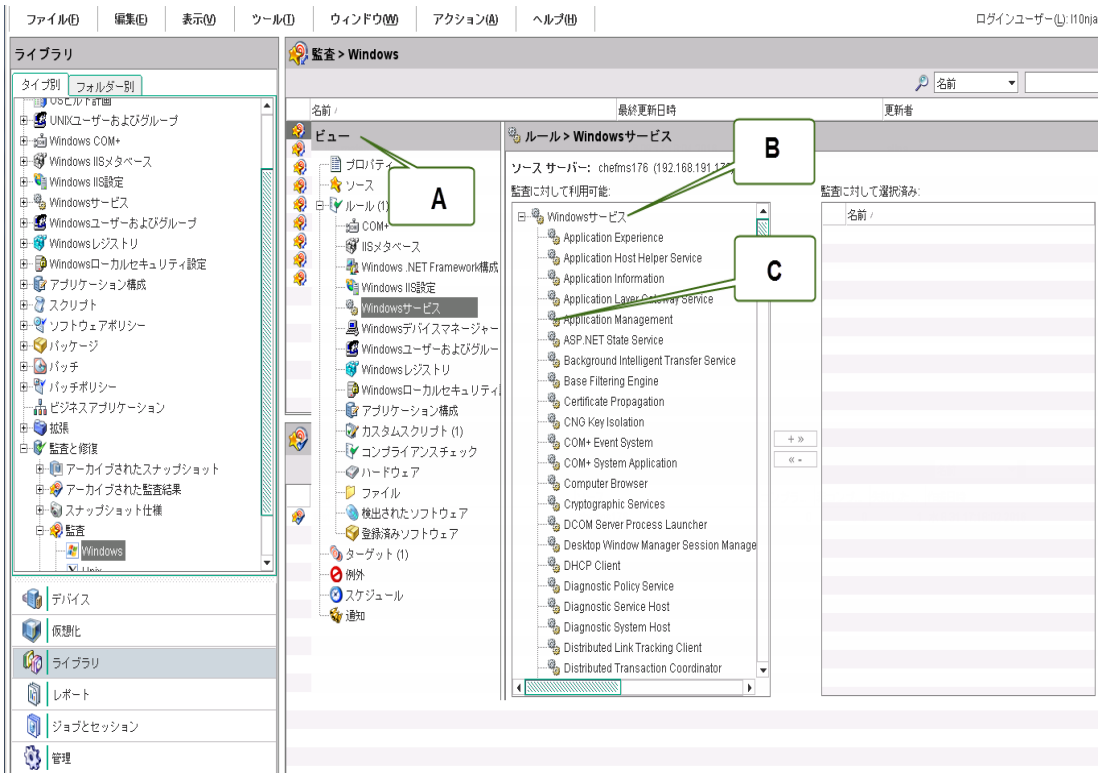
監査と修復は、システム管理を容易にする役割も果たします。たとえば、自社チームが開発したデータベースサーバーやミドルウェアアプリケーションなどのアプリケーションを実行するサーバーのクラスを監視するとします。アプリケーションを実行するサーバーを構成して監視する際に、構成の理想的な状態を記録したリストを作成します。このリストには、ファイル、ディレクトリ、ネットワーク共有のアクセス権などが含まれます。

これらの構成を定義した監査を作成し、アプリケーションをインストールした後でサーバーを監査できます。監査結果は、アプリケーションがインストールされ、基準に従って正しく構成されたかどうかを確認します。構成がコンプライアンス違反の場合、問題を解決するためのアドホック監査を作成できます。監査結果がエラーを示す場合、理想的な構成に合わせてサーバーを修復できます。構成変更が実務で動作するように、監査を実行するスケジュールを設定し、完了時に通知を受け取ることができます。

次の図は監査を選択するためのウィンドウです。マークの説明を次に示します。

- **A**: [ビュー] パネルに表示されたカテゴリは、SAの変更不可能な機能または変更可能なプラグ可能チェックを持ちます。
- **B**: これはWindowsサービスを扱うSA機能を指します。
- **C**: これはWindowsサービスを対象とするプラグ可能チェックを示します。

Windowsサービスの監査ルール



各チェックは1つのルールを評価します。複数のチェックを1つのポリシーにまとめることができます。

SAの監査と修復機能には、多数の標準チェックが付属しています。ほとんどの監査は、必要なチェックを選択することで実行できます。選択できる監査は、開発者がさらに多くのチェックを設計し、コーディングしてテストし、HPE Live Networkを通じてシステムに追加することで増え続けていきます。これらのチェックは、完成したポリシーとしてインポートできます。

ただし、各企業にはそれぞれ独自の課題やリソースがあるので、コンプライアンス判定のための監査に必要な基準が、SAの監査と修復フレームワーク内には存在しないことも考えられます。このために、独自のカスタムプラグ可能チェックを作成する方法が用意されています。

監査と修復機能は、個別のルールによって、SAで管理されているサーバーのコンプライアンス状態を評価します。この機能は、ルールに定義された必要な構成状態に一致しないサーバーを修復することもできます。これらのルールには、さまざまなサーバーパラメーター、レジストリ値、ファイルアクセス権、アプリケーション構成、ファイルの存在、COM+オブジェクトなどが含まれます。

Windows環境では、Webサーバールールをアプリケーション構成に指定することもできます。これは、Microsoft Internet Information Services (IIS) のWebサーバー構成ファイルUrlScan.iniに基づきます。SAは、値の一部または全部を特定の構成ファイルと比較し、ファイルから必要な要素を選択して、これらの値または構成ファイルエントリが存在することを確認します。詳細については、アプリケーション構成を参照してください。

SAには、設計済みの監査ルールが多数付属しています。各ルールは、サーバーまたはサーバーグループの構成の必要な状態を定義します。一部のルールは値に基づいており、比較演算子 (<、>、==、!=、次の値を含む、など)、1つの値または値のセット、1つ以上のチェック (監査対象アイテムの状態を評価するために使用されるコードを記述したもの) が用意されています。比較データは、コンプライアンスまたは非コンプライアンスを決定します。チェックが修復をサポートする場合、ルールには修復値が含まれる場合もあります。

ルールは1つのチェックから構成されます。カスタム内容オブジェクトを使用して、プラグ可能チェックという形式で新しい機能を作成できます。また、関連するプラグ可能チェックを、使用に便利なように監査ポリシーにまとめることもできます。

プラグ可能チェックの作成

プラグ可能チェックとは、管理対象サーバーにダウンロードされ、監査と修復フレームワークによって実行されるコードです。チェックを使用して、監査と修復のネイティブのプロパティを拡張し、特殊化された機能を追加することができます。各プラグ可能チェックには、カスタマイズされたconfig.xmlファイルと、監査対象の機能とconfig.xmlファイルに指定された値とを比較する1つ以上のスクリプトが含まれます。プラグ可能チェックには、監査対象サーバーの指定された変数をconfig.xmlファイルに指定された値に設定するスクリプトが含まれる場合もあります。プラグ可能チェックのスクリプトの作成には、Python、Visual Basic Scripting (VBS)、BAT、シェルスクリプトが使用できます。プラグ可能チェックはZIPアーカイブにパッケージ化されます。

CISチェックのほとんどは、CISベンチマークから直接変換されたものです。詳細についてはhttp://www.cisecurity.orgを参照してください。

ほとんどのタイプのチェックは、次のいずれかのカテゴリに分類されます。

- Windowsレジストリチェック
- Unixサービスチェック
- ユーザーチェック (パスワードまたはシャドウファイルの情報を使用する場合があります)

プラグ可能チェックのガイドライン

サーバーのメンテナンスを容易にするため、次のガイドラインに従ってください。

- 新しいプラグ可能チェックを作成する際には、名前に十分注意します。チェックの目的を示す名前を付け、スペースの代わりに下線を使用します。わかりやすい名前の例としては、`Users_Without_Password_Expiration`などが挙げられます。これは、サーバーが数百以上のチェックを処理するような場合に、目的のチェックを簡単に見つけられるようにするためです。
- 汎用的なチェックを作成します。これにより、数行のコードを変更するだけで、同じ実行タイプの別のチェックを容易に作成できます。たとえば、CIS2k3 Windowsサービスチェックでは、ほとんどの場合1行のコードを変更するだけで、新しいサービスに対する新しいチェックを作成できます。
- 監査 (get) および修復 (set) スクリプトの名前を付ける際には、ディレクトリ名のスペースと下線を削除し、getまたはsetを必要に応じて先頭に付けます。たとえば、`getUsersWithoutPasswordExpiration.sh`は適切な監査ファイル名の例です。カスタムチェックを使用する予定なのが自分だけであっても、この規則は守ってください。

- エラーチェックに注意してください。スクリプトでエラーが発生すると、予期しない戻り値のために監査結果がコンプライアンス違反として報告される可能性があります。予期しないエラーや例外をトラップし、トラブルシューティングを容易にするため、それに関する情報をstdoutまたはstderrに出力します。
- チェックはできる限り単純な真または偽の2値のケースに変換します。
- 特定のベンチマークケースだけでなく、それと相補的なケースも常に処理するようにします。たとえば、「サービスXの無効化」プラグ可能チェックを作成する際には、ほとんどのコードを再使用して、「サービスXの有効化」も容易に作成できます。そうしておけば、後で逆の条件のテストが必要になった場合に役立ちます。
- できる限り、フレームワークによって定義された標準の終了コードを使用します。次に示すのがそうです。

```
EXIT_FAILURE=220
EXIT_ERR_USAGE=221
EXIT_ERR_INVALID_OS=222
```

- ブール値型のチェックで無効または有効を返す場合は、0が無効、1が有効を表すようにします。
- 各プラグ可能チェックはZIPアーカイブにパッケージ化します。1つのファイルシステムディレクトリには、次の表に示すファイルが含まれます。

プラグ可能チェックの内容

ファイル名	説明
`config.xml`	(必須) このプラグ可能チェックがどのように実行され、戻り、最終的にコンプライアンスまたは非コンプライアンスを報告するかを定義するXML構成ファイル。
`get`名前`.{py | sh | BAT | vbs}`	(必須) Python、VBS、BAT、シェルのいずれかで作成された監査スクリプト。監査対象のオブジェクトを評価し、config.xmlの定義に従ってテキストと終了コードを返します。
`set`名前`.{py | sh | BAT | vbs}`	(オプション) Python、VBS、BAT、シェルのいずれかで作成された修復スクリプト。監査スクリプトでチェックされた条件を修復します。
その他のコード、スクリプト、ライブラリ	(オプション) 監査スクリプトまたは修復スクリプトから使用される補助または補足用のスクリプト。

監査スクリプトと修復スクリプトのファイル名はgetおよびsetで始まる必要はありませんが、この規則に従えばメンテナンスが容易になります。

次の例は、プラグ可能チェックのディレクトリ構造を示します。

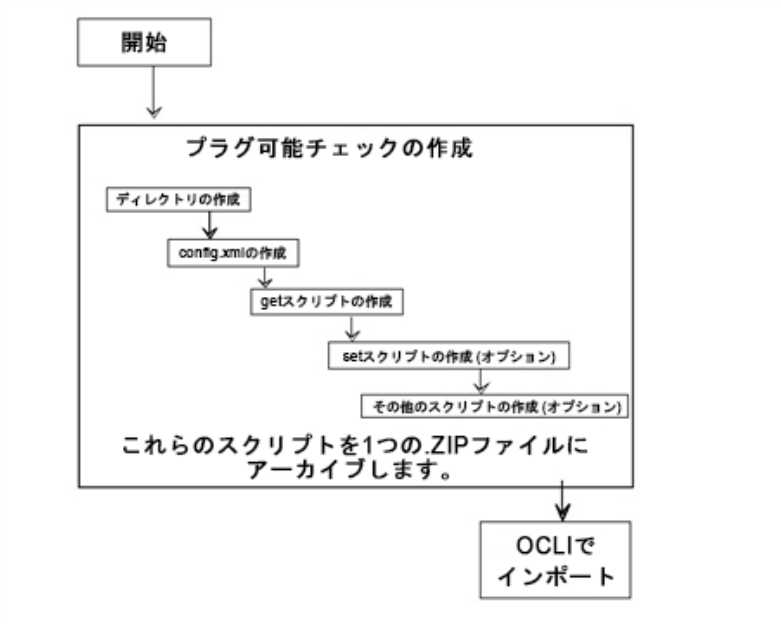
```
./check_name/
./check_name/config.xml
```

```
./check_name/getcheckname.py
./check_name/setcheckname.py
```

プラグ可能チェックの開発プロセス

次の図には、コマンドライン環境で行われる開発プロセスの概要を示します。

開発プロセス



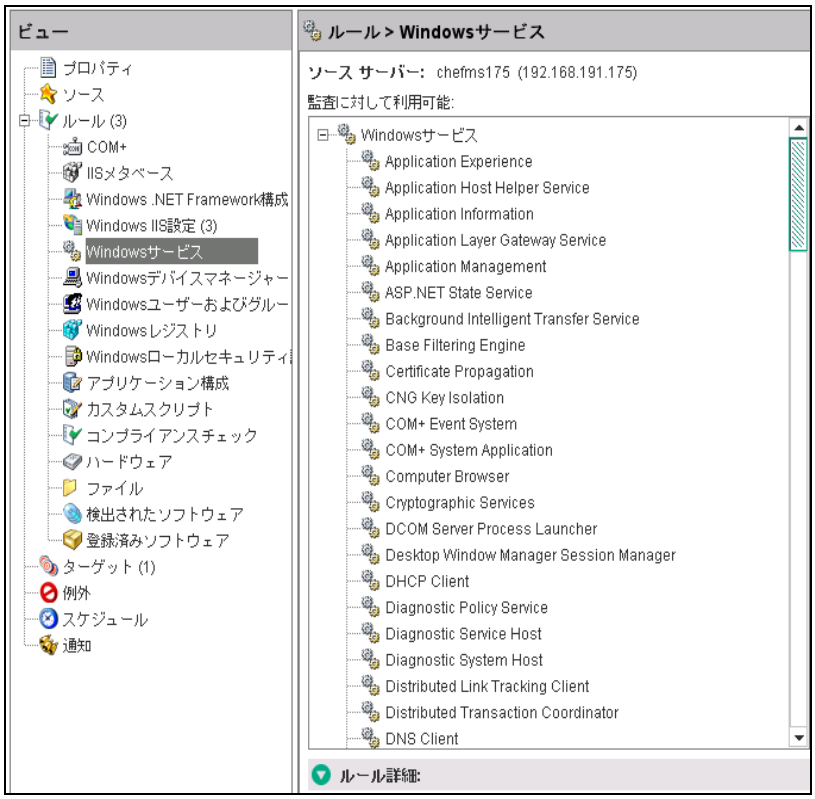
プラグ可能チェック構成 (config.xml)

config.xmlファイルはプラグ可能チェックの仕様ファイルであり、このチェックがSAクライアントに表示される方法、デフォルト値、比較の値タイプ、チェックのカテゴリを制御する要素が含まれています。たとえば、`config.xml`ファイルの次の要素は、SAクライアントでのプラグ可能チェックのルールカテゴリを決定します。

```
<checkCategory>Windows Services</checkCategory>
```

標準カテゴリは、次の図に示すように、それぞれ固有のアイコンで表され、ハードウェア、ソフトウェア、オペレーティングシステム、ユーザーとグループ、ファイルシステムなどがあります。

ルール階層内のプラグ可能チェックカテゴリ



次のリストは、config.xmlファイルのテンプレートを示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE checkConfiguration SYSTEM "check.dtd">
<checkConfiguration version="1.0">
<checkName>$CHECKNAME</checkName>
<checkGUID>$CHECKGUID</checkGUID>
<checkDefaultDescription>$CHECKDESCRIPTION</checkDefaultDescription>
<checkRemediationDefaultDescription> $CHECKREMEDIATIONDESCRIPTION </
checkRemediationDefaultDescription>
<checkGetScriptName>$GETSCRIPTNAME</checkGetScriptName>
<checkGetScriptType>PY</checkGetScriptType><!-- Or SH for shell, BAT for Bat,
VBS for Visual Basic -->
<checkSetScriptName>$SETSCRIPTNAME</checkSetScriptName><!-- Optional -->
<checkSetScriptType>PY</checkSetScriptType><!-- Optional -->
<checkVersion>32b.0-1.0</checkVersion>
<checkReturnType>$RETURNTYPE</checkReturnType> <!-- EXITCODE, STRING, or
NUMBER -->
<checkTestIDs>
<checkTestID>$CHECKTESTID</checkTestID> <!-- Optional -->
</checkTestIDs>
<checkPlatformTypes>
<checkPlatform>$PLATFORMTYPE</checkPlatform> <!-- Currently Unix or Windows --
>
</checkPlatformTypes>
```

```
<checkCategories>
<checkCategory>$CATEGORY</checkCategory> <!-- Top-level GUI category -->
</checkCategories>
<checkGetArguments> <!-- All arguments are optional -->
<checkGetArgument>
<checkGetArgumentType>$GETARGTYPE</checkGetArgumentType> <!-- STRING or NUMBER
-->
       <checkGetArgumentDefaultLabel>$GETDEFAULTLABEL</
checkGetArgumentDefaultLabel>
               <checkGetArgumentDefaultDescription>$GETDEFAULTDESCRIPTION</
checkGetArgumentDefaultDescription>
               <checkGetArgumentDefaultValue>$GETDEFAULTVALUE</
checkGetArgumentDefaultValue>
       </checkGetArgument>
</checkGetArguments>
<checkSetArguments> <!-- Also optional -->
<checkSetArgument>
<checkSetArgumentType>$SETARGTYPE</checkSetArgumentType>
       <checkSetArgumentDefaultLabel>$SETDEFAULTLABEL</
checkSetArgumentDefaultLabel>
               <checkSetArgumentDefaultDescription>$SETDEFAULTDESCRIPTION</
checkSetArgumentDefaultDescription>
       <checkSetArgumentDefaultValue>$SETDEFAULTVALUE</
checkSetArgumentDefaultValue>
</checkSetArgument>
</checkSetArguments>
<checkSuccessExitCodes> <!-- Only for EXITCODE type checks, generally at least
two entries -->
       <checkSuccessExitCode>
<checkSuccessExitCodeValue>$EXITCODEVALUE</checkSuccessExitCodeValue>
               <checkSuccessExitCodeDefaultDescription>$EXITCODEDESCRIPTION
       </checkSuccessExitCodeDefaultDescription>
               <checkSuccessExitCodeDefaultDisplayName>$EXITCODEDISPLAYNAME
       </checkSuccessExitCodeDefaultDisplayName>
</checkSuccessExitCode>
</checkSuccessExitCodes>
</checkConfiguration>
```

詳細については、config.xmlファイルの文書型定義 (DTD)を参照してください。

監査 (get) スクリプト

監査スクリプト (getスクリプトとも呼ばれる) は、管理対象サーバーから値を取得するために作成します。スクリプトは、config.xmlファイルの指定に従って、オプションのパラメーター付きで実行されます。スクリプトでEXITCODEチェックを実行する場合、スクリプトの結果がconfig.xmlファイルに指定された終了コー

ドと比較されます。STRINGおよびNUMBER戻り値型チェックの場合、結果はSTDOUTへの出力と比較されます。

監査スクリプトには、定義済みのリターンコードのセットがあります。チェックのconfig.xmlファイルで追加のリターンコードを定義することもできます。

監査スクリプトは、情報メッセージを表示することができます。メッセージは、監査スクリプトのエラーのトラブルシューティングの際に有用です。次のPython監査スクリプトの例を見てください。

修復 (set) スクリプト

修復スクリプト (setスクリプトとも呼ばれる) は、監査スクリプトが終了時に成功を返すように管理対象サーバーを変更するために作成します。スクリプトは、チェックのconfig.xmlファイルの指定に従って、オプションのパラメーター付きで実行されます。

setスクリプトはオプションであり、対応するgetスクリプトとよく似た性質を持つ場合も、まったく異なる (そしてもっと長い) 場合もあります。

シェルの観点からは、スクリプト自体には、使用されるリターンコードを別として、特別なことは何もありません。ほとんどのチェックは、何らかのデバッグ出力または情報メッセージを表示します。これは通常はユーザーには見えませんが、スクリプトでエラーが発生した場合には、メッセージがトラブルシューティングの役に立つことがあります。

標準的慣行として、setスクリプトは常に少なくとも1つのパラメーターを取るようにします。また、既存のチェックにsetスクリプトを追加した場合、SAクライアントにうまく表示されるようにconfig.xmlファイルを修正するようにしてください。

修復スクリプトが成功した場合は、終了コード0を返すようにします。それ以外の終了コードは、修復操作の失敗を示します。

次のPython setスクリプトの例を見てください。

```
import sys
import os
import string
if __name__ == "__main__":
#If there are set arguments they will be loaded into
# sys.argv
# Enter the desired set code here.Stdout may be used for
# debugging.
#Uses exitcode 0 for success, and all other values for
# failure.
# enter condition where set script if successful. for this
# example, use ‘if 1’
if 1:
```

```
        sys.exit(0)
else:
        sys.exit(-1)
```

プラグ可能チェックのその他のコード

プラグ可能チェックには、getまたはsetスクリプト以外のコードが含まれる場合もあります。ライブラリ、実行可能ファイル、他のスクリプトをチェックに追加して、setまたはgetスクリプトから実行時に使用することができます。

ZIPに追加のコードを含めることもできます。

プラグ可能チェックのZIP化

config.xmlファイル、監査 (get) スクリプト、オプションの修復 (set) スクリプトを作成したら、これらのファイルを含むZIPアーカイブを作成します。次のシェル履歴は、UNIX環境での作成プロセスを示します。

```
# ls
check_name
# cd check_name
# zip ../checkname.zip *
adding: config.xml
adding: getcheckname.py
adding: setcheckname.py
# unzip -t ../checkname.zip
testing: config.xml OK
testing: getcheckname.py OK
testing: setcheckname.py OK
No errors detected in compressed data of ../checkname.zip.
```

プラグ可能チェックのインポート

プラグ可能チェックをSAコアまたはメッシュにインポートするには、SAコンテンツユーティリティに記述されているOCLI 1.0ユーティリティを使用します。次のシェル履歴は、Linuxのインポートプロセスの例を示します。

```
# cp checkname.zip /var/tmp/checks
# cd /var/tmp/checks
# cp opsware_32.a.692.0-upload/disk001/packages/Linux/3AS/ocli-32a.2.0.5-
linux-3AS .
# chmod 755 ocli-32a.2.0.5-linux-3AS
```

```
# ./ocli-32a.2.0.5-linux-3AS
# ../ocli/login.sh
# export PATH=/opt/opsware/bin:$PATH
# oupload -C"Customer Independent" -t"Server Configuration Check" --
forceoverwrite --old -O"SunOS 5.8" your_Pluggable_check.zip
```

ouploadコマンドでは、"SunOS 5.8" を使用して、チェックがSAクライアントのUnix一般のカテゴリに分類されることを指定します。Windowsカテゴリのチェックを指定するには、"Windows 2003" を使用します。

監査ポリシーの作成

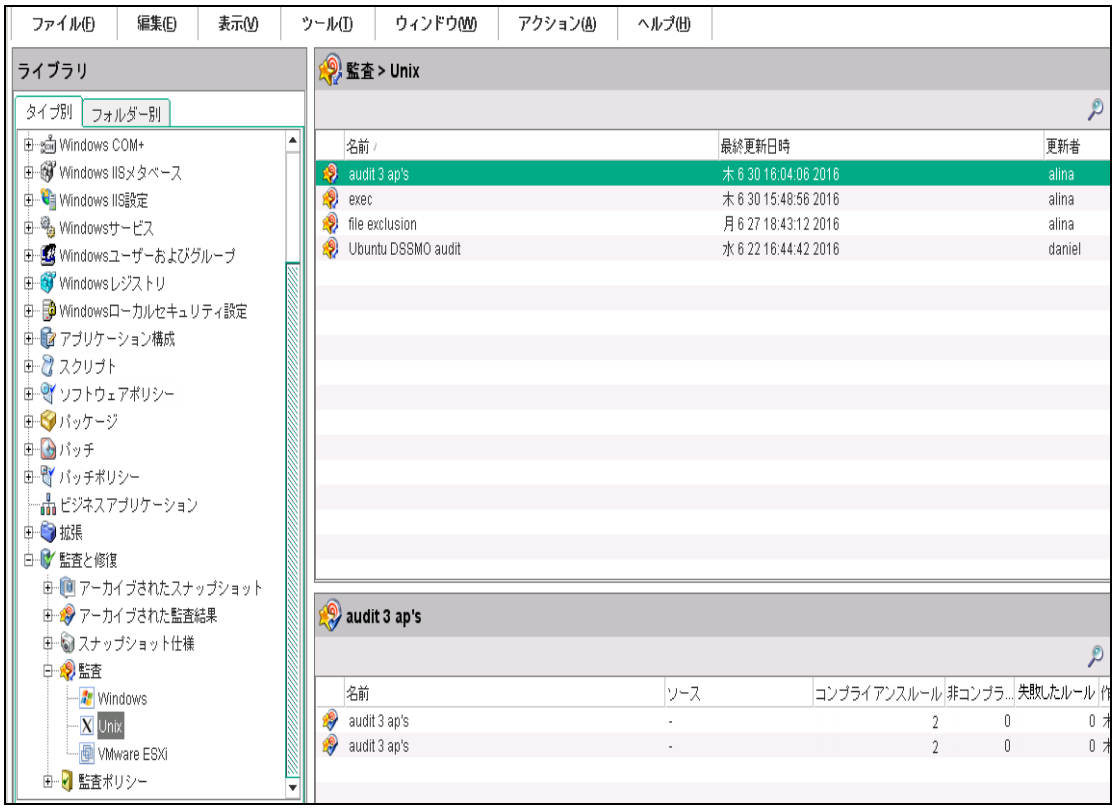
監査ポリシーの作成手順を下の図に示します。

監査ポリシーの作成手順

監査ポリシーの作成

監査ポリシーは、ルールから構成されます。各ルールは1つ以上のチェックから構成され、ユーザーが作成したプラグ可能チェックを含むことができます。監査ポリシーとルールの表示、作成、編集は、SAクライアントで行われます。次の図に、モデルシステムで使用可能な監査ルールのリストを示します。

監査ルールのリスト



監査ポリシーの作成の詳細については、HPE SSOポータルで『Server Automation管理ガイド』を参照してください。

監査ポリシーのエクスポート

新しい監査ポリシーを別のSAコアに移動するには、現在のコアからポリシーをエクスポートして別のコアにインポートします。このためには、DCML Exchange Tool (DET) コマンドラインユーティリティを使用します。このツールは、新しくインストールしたSAコアに既存コアのポリシーなどのコンテンツを入れるために使用します。この手順の詳細については、HPE SSOポータルで『Server Automation管理ガイド』を参照してください。

config.xmlファイルの文書型定義 (DTD)

このファイルは、SAクライアントの表示名と説明、デフォルト値、チェックコードから返された値に対して実行する比較、これらの値を表示するSAクライアントのカテゴリなどを規定します。

デフォルトのconfig.xmlファイルにあるcheckGetArgumentsとcheckSetArgumentsの2つの要素は、実行時にスクリプトにデータ値を渡すために使用されます。プログラマブルチェックに引数が不要な場合は、これらの要素をconfig.xmlファイルから削除します。

次に示すconfig.xmlのDTDは、SAによって動的に生成されます。

```
<!ELEMENT checkConfiguration (checkName, checkGUID, checkDefaultDescription,
checkRemediationDefaultDescription?, checkGetScriptName?,
checkGetScriptType?, checkSetScriptName?, checkSetScriptType?, checkVersion,
checkAllowRemediationOnFailure?, checkReturnType, checkTestIDs?,
checkPlatformTypes, checkExclusivePlatforms?, checkExcludePlatforms?,
checkCategories, checkGetArguments?, checkSetArguments?,
checkComparisonDefaults?, checkCompareValidValues?, checkSuccessExitCodes?)>
<!ATTLIST checkConfiguration version CDATA #REQUIRED>
<!ELEMENT checkName (#PCDATA)>
<!ELEMENT checkGUID (#PCDATA)>
<!ELEMENT checkDefaultDescription (#PCDATA)>
<!ELEMENT checkRemediationDefaultDescription (#PCDATA)>
<!ELEMENT checkGetScriptName (#PCDATA)>
<!ELEMENT checkGetScriptType (#PCDATA)>
<!ELEMENT checkSetScriptName (#PCDATA)>
<!ELEMENT checkSetScriptType (#PCDATA)>
<!ELEMENT checkVersion (#PCDATA)>
<!ELEMENT checkAllowRemediationOnFailure (#PCDATA)>
<!ELEMENT checkReturnType (#PCDATA)>
<!ELEMENT checkTestIDs (checkTestID+)>
<!ELEMENT checkTestID (#PCDATA)>
<!ELEMENT checkPlatformTypes (checkPlatform+)>
<!ELEMENT checkPlatform (#PCDATA)>
<!ELEMENT checkExclusivePlatforms (checkExclusivePlatform+)>
<!ELEMENT checkExclusivePlatform (#PCDATA)>
<!ELEMENT checkExcludePlatforms (checkExcludePlatform+)>
<!ELEMENT checkExcludePlatform (#PCDATA)>
<!ELEMENT checkCategories (checkCategory+)>
<!ELEMENT checkCategory (#PCDATA)>
<!ELEMENT checkGetArguments (checkGetArgument+)>
<!ELEMENT checkGetArgument (checkGetArgumentType,
checkGetArgumentDefaultLabel, checkGetArgumentDefaultDescription,
```

```
checkGetArgumentDefaultValue?, checkGetArgumentValidValues?)>
<!ELEMENT checkGetArgumentType (#PCDATA)>
<!ELEMENT checkGetArgumentDefaultLabel (#PCDATA)>
<!ELEMENT checkGetArgumentDefaultDescription (#PCDATA)>
<!ELEMENT checkGetArgumentDefaultValue (#PCDATA)>
<!ELEMENT checkGetArgumentValidValues (checkGetArgumentValidValue+)>
<!ELEMENT checkGetArgumentValidValue (checkGetArgumentValidValueItem,
checkGetArgumentValidValueDisplayName)>
<!ELEMENT checkGetArgumentValidValueItem (#PCDATA)>
<!ELEMENT checkGetArgumentValidValueDisplayName (#PCDATA)>
<!ELEMENT checkSetArguments (checkSetArgument+)>
<!ELEMENT checkSetArgument (checkSetArgumentType,
checkSetArgumentDefaultLabel, checkSetArgumentDefaultDescription,
checkSetArgumentDefaultValue?, checkSetArgumentValidValues?)>
<!ATTLIST checkSetArgument populateFromRule CDATA #IMPLIED>
<!ELEMENT checkSetArgumentType (#PCDATA)>
<!ELEMENT checkSetArgumentDefaultLabel (#PCDATA)>
<!ELEMENT checkSetArgumentDefaultDescription (#PCDATA)>
<!ELEMENT checkSetArgumentDefaultValue (#PCDATA)>
<!ELEMENT checkSetArgumentValidValues (checkSetArgumentValidValue+)>
<!ELEMENT checkSetArgumentValidValue (checkSetArgumentValidValueItem,
checkSetArgumentValidValueDisplayName)>
<!ELEMENT checkSetArgumentValidValueItem (#PCDATA)>
<!ELEMENT checkSetArgumentValidValueDisplayName (#PCDATA)>
<!ELEMENT checkComparisonDefaults (checkComparisonDefaultOperator?,
checkComparisonDefaultValues)>
<!ELEMENT checkComparisonDefaultOperator (#PCDATA)>
<!ATTLIST checkComparisonDefaultOperator not CDATA #IMPLIED>
<!ATTLIST checkComparisonDefaultOperator caseInsensitive CDATA #IMPLIED>
<!ELEMENT checkComparisonDefaultValues (checkComparisonDefaultValue+)>
<!ELEMENT checkComparisonDefaultValue (checkComparisonDefaultValueItem,
checkComparisonDefaultValueDisplayName)>
<!ELEMENT checkComparisonDefaultValueItem (#PCDATA)>
<!ELEMENT checkComparisonDefaultValueDisplayName (#PCDATA)>
<!ELEMENT checkCompareValidValues (checkCompareValidValue+)>
<!ELEMENT checkCompareValidValue (checkCompareValidValueItem,
checkCompareValidValueDisplayName)>
<!ELEMENT checkCompareValidValueItem (#PCDATA)>
<!ELEMENT checkCompareValidValueDisplayName (#PCDATA)>
<!ELEMENT checkSuccessExitCodes (checkSuccessExitCode+)>
<!ELEMENT checkSuccessExitCode (checkSuccessExitCodeValue,
checkSuccessExitCodeDefaultDescription,
checkSuccessExitCodeDefaultDisplayName)>
<!ELEMENT checkSuccessExitCodeValue (#PCDATA)>
<!ELEMENT checkSuccessExitCodeDefaultDescription (#PCDATA)>
<!ELEMENT checkSuccessExitCodeDefaultDisplayName (#PCDATA)>
```

次の表は、config.xmlのDTDの要素を示します。

DTD要素と属性

要素	属性
checkConfiguration version	1.0に設定。監査と修復フレームワークで必要な場合のみ変更します。
checkName	SAクライアントに表示されるチェック/ルールの英語名。
checkGUID	標準のGUID、例: 9500A4AE-EE9E-4383-87F2-BAD7DDC26C59 guidgen Windowsユーティリティで生成するか、Webサイトからダウンロードするか、その他の方法で生成できます。 GUIDは一意である必要があります。そうでないと、プラグ可能チェックをコアにアップロードする際にエラーが発生します。一意のGUIDを持つチェックをアップロードした後では、GUIDを変更することはできません。変更した場合、オリジナルを削除しない限り、再アップロード時に「データベース一意性制約エラー」が発生します。チェックはGUIDによって一意に識別されますが、アップロード時には (ZIPファイルの) 名前のみで識別されます。
checkDefaultDescription	SAクライアントの説明ボックスに表示されます。改行とHTMLは反映されます。HTMLの場合、HTMLタグは<と>を使用して変換する必要があります。
checkRemediationDefaultDescription	SAクライアントでチェック/ルールの [修復] セクションに表示されます。
checkGetScriptName	getスクリプトのファイル名。例: getUsersWithoutPasswordExpiration.sh
checkGetScriptType	コードのタイプによって、実行するインタープリターが決まります。getスクリプトとsetスクリプトのタイプとしては、SH、VBS、PY、BATが使用できます。
checkSetScriptName	修復スクリプトのファイル名。
checkSetScriptType	コードのタイプによって、実行するインタープリターが決まります。set (修復) スクリプトのタイプとしては、SH、VBS、PY、BAが使用できます。

DTD要素と属性 (続き)

要素	属性
checkVersion	これはSAとフレームワークのビルド番号に基づきます。例: 32b.0-1.0
checkAllowRemediationOnFailure	スクリプトによっては、get段階でエラーが発生しても、その条件を修復スクリプトで修正できることがあります。この場合、スクリプトがエラーになっても修復を実行することができます。たとえば、レジストリキーが存在されていないことが定義されていなくても、setコードで作成して設定することができます。
checkReturnType	使用可能な値は、EXITCODE、STRING、NUMBERです。 EXITCODE - Wscript.Quit()、exit、returnなどによる標準のスクリプトの戻り。 NUMBER - 監査と修復フレームワークがstdoutの出力を捕捉して、数値型として解釈します。 STRING - 監査と修復フレームワークがstdoutの出力を捕捉して、文字列型として解釈します。
checkTestIDs	テストIDのリスト。
checkTestID	CIS、MSFT、NSA、またはその他のポリシーの標準命名法を表示するために用いられます。例: CIS-RHEL 8.4。これは自由形式のフィールドであり、SAクライアントに表示されるので、TON内容と対応するように一貫した名前を付ける必要があります。
checkPlatformTypes	チェックに対する有効なプラットフォームタイプのリスト。
checkPlatform	WINDOWS | UNIX (または両方を個別の要素として)
checkExclusivePlatforms	排他的プラットフォームのリスト。監査と修復は、現在はデフォルトでWindowsとUnixの区分を採用していますが、実世界の標準や、オペレーティングシステムの間の制限や違いによっては、この区分では対応できない可能性もあります。監査と修復を、spinから取得されたプラットフォームIDによって指定される任意のプラットフォームに制限することができます。

DTD要素と属性 (続き)

要素	属性
	このパラメーターは、『SA Supported Platforms』ドキュメントに記載されたサポートされるオペレーティングシステムの1つを参照することができます。
checkExclusivePlatform	個別のプラットフォームID。
checkExcludePlatforms	除外されるプラットフォームのリスト。PlatformTypeがUNIXの場合、UNIXセット (すべてのLinux＋すべてのUnix) から除外するプラットフォームIDを指定できます。
checkExcludePlatform	個別のプラットフォームID
checkCategory	チェックが表示されるSAクライアントカテゴリ。現時点では、チェックは1つのカテゴリのみに表示できます。カテゴリが存在しない場合、アップロード時に作成されます。可能な場合は、既存のチェックに対する次の標準カテゴリを使用します。 イベントロギング ファイルシステム オペレーティングシステム オペレーティングシステム|ドメインコントローラー (サブカテゴリ) オペレーティングシステム|ネットワーク (サブカテゴリ) レジストリ Services ユーザーとグループ
checkGetArgument (checkGetArgumentType, checkGetArgumentDefaultLabel, checkGetArgumentDefaultDescription, checkGetArgumentDefaultValue?, checkGetArgumentValidValues?)	getスクリプトのパラメーターを指定します。
checkGetArgumentType	NUMBER | STRING
checkGetArgumentDefaultLabel	SAクライアントでの入力ボックスまたはドロップダウンの隣のタグ。
checkGetArgumentDefaultDescription	マウスでポイントすると表示される詳細な説明のテキスト。
checkGetArgumentDefaultValue	このgetパラメーターのデフォルト値。

DTD要素と属性 (続き)

要素	属性
checkGetArgumentValidValue (checkGetArgumentValidValueItem, checkGetArgumentValidValueDisplayName	checkGetArgumentValidValueItem (#PCDATA) checkGetArgumentValidValueDisplayName (#PCDATA)>
checkGetArgumentValidValues (checkGetArgumentValidValue+)	(オプション) パラメーターを0/無効、1/有効などに制限する場合に便利です。
checkSetArguments (checkSetArgument+)	checkSetArgument (checkSetArgumentType, checkSetArgumentDefaultLabel, checkSetArgumentDefaultDescription, checkSetArgumentDefaultValue?, checkSetArgumentValidValues?) setArgument要素はGetArgumentsと同じですが、修復/setスクリプト (存在する場合) に対して使用されます。 例外: checkSetArgument populateFromRule - setパラメーターのデフォルト値がconfig.xmlに指定されている場合に、デフォルト値をルールデータから取るかどうか。通常は常にtrueに設定します。
checkSetArgumentType	NUMBER | STRING
checkSetArgumentDefaultLabel	SAクライアントでの入力ボックスまたはドロップダウンの隣のタグ。
checkSetArgumentDefaultDescription	マウスでポイントすると表示される詳細な説明のテキスト。
checkSetArgumentDefaultValue	このsetパラメーターのデフォルト値。
checkSetArgumentValidValues (checkSetArgumentValidValue+)	
checkSetArgumentValidValue (checkSetArgumentValidValue Item, checkSetArgumentValidValue DisplayName)>	checkSetArgumentValidValueItem (#PCDATA)> checkSetArgumentValidValueDisplayName (#PCDATA)> checkSetArgumentValidValueItem (#PCDATA)> checkSetArgumentValidValueDisplayName (#PCDATA)>

DTD要素と属性 (続き)

要素	属性
checkSetArgumentValidValue Item	(オプション) パラメーターを0/無効、1/有効などに制限する場合に便利です。
checkSetArgumentValidValueDisplayName	
<!ELEMENT checkComparisonDefaults (checkComparisonDefaultOperator?, checkComparisonDefaultValues)>	checkComparisonDefaultOperator not — negation of operator specified, TRUE | FALSE checkComparisonDefaultOperator caseInsensitive - STRINGタイプのみで有効。
<!ELEMENT checkComparisonDefaultOperator (#PCDATA)>	比較演算子のデフォルト値のリスト。TONビルドフレームワーク外部のフィールドまたは開発に便利。
checkComparisonDefaultValues (checkComparisonDefaultValue+)	checkComparisonDefaultValue (checkComparisonDefaultValueItem, checkComparisonDefaultValueDisplayName).
checkComparisonDefaultValueIte	デフォルトの値。コードに渡されます。
checkComparisonDefaultValueDisplayName	SAクライアントに表示される値の表示名。
checkCompareValidValues (checkCompareValidValue+)> checkCompareValidValue (checkCompareValidValueItem, checkCompareValidValueDisplayName)> checkCompareValidValueItem (#PCDATA)> checkCompareValidValueDisplayName (#PCDATA)>	
checkSuccessExitCodes (checkSuccessExitCode+) checkSuccessExitCode (checkSuccessExitCodeValue, checkSuccessExitCodeDefaultDescription, checkSuccessExitCodeDefaultDisplayName)>	checkReturnTypeがEXITCODEの場合、スクリプトが正しく動作した場合の有効な値を定義する必要があります。これは通常、コンプライアンスと非コンプライアンスの期待される値を含みます。ここに指定された値以外が返された場合、スクリプトのエラーと見なされ、SAクライアントとレポートに異なる方法で表示されます。
checkSuccessExitCodeValue	スクリプト完了の値。例: 0 (通常は「無効」)。

DTD要素と属性 (続き)

要素	属性
checkSuccessExitCodeDefaultDescription	DisplayName/Valueに関する、マウスでポイントすると表示されるテキスト。
checkSuccessExitCodeDefaultDisplayName	この値に関してユーザーに表示される値またはテキスト。例: Disabled。

検索フィルターの構文

フィルターの文法

検索フィルターは、findServerRefsなどのメソッドのパラメーターです。検索フィルターの式を使うと、SAオブジェクト (サーバーやフォルダーなど) への参照を、オブジェクト属性の値に基づいて取得することができます。検索フィルターの形式的構文は次のとおりです。

```
<filter>                ::=     (<expression-junction>)+
<expression-junction>   ::=     <expression-list-open> <junction>
(<expression>)+ <expression-list-close>
<expression>            ::=     <expression-open> <attribute> <general-delimiter> <operator>
<general-delimiter> <value-list> <expression-close>

<attribute>             ::=     <resource_field>
<vo_member>             ::=     <text>
<resource_field>        ::=     <text>
<value-list>            ::=     (<double-quote> <text> <double-quote>)* |
(<number>)*
<text>                  ::=     [a-z] [A-Z] [0-9]
<number>                ::=     [0-9] [.]

<junction>              ::=      <union-junction> | <intersect-junction>
<union-junction>        ::=     '|'
<intersect-junction     ::=     '&'
<expression-list-open>  ::=     '('
<expression-list-close> ::=     ')'
<expression-open>       ::=     '(' | '{'
<expression-close>      ::=     '(' | '}'
<general-delimiter>     ::=     <whitespace>
<whitespace>            ::=     ' '
<double-quote>          ::=     '"'
<escape-character>      ::=     '\'
<operator>              ::=     <equal_to> |...| <contains_or_above>

Valid operators for the preceding line:

<equal_to>              ::=     '=' | 'EQUAL_TO'
<not_equal_to>          ::=     '!=' | '<>' | 'NOT_EQUAL_TO'
<in>                    ::=     '=' | 'IN'
<not_in>                ::=     '!=' | '<>' | 'NOT_IN'
<greater_than>          ::=     '>' | 'GREATER_THAN'
<less_than>             ::=     '<' | 'LESS_THAN'
```

```
<greater_than_or_equal> ::=     '>=' | 'GREATER_THAN_OR_EQUAL'
<less_than_or_equal>    ::=     '<=' | 'LESS_THAN_OR_EQUAL'
<begins_with>           ::=     '=*' | 'BEGINS_WITH'
<ends_with>             ::=     '*=' | 'ENDS_WITH'
<contains>              ::=    '*=*' | 'CONTAINS'
<not_contains>          ::=    '*<>*' | 'NOT_CONTAINS'
<in_or_below>           ::=    'IN_OR_BELOW'
<in_or_above>           ::=    'IN_OR_ABOVE'
<between>               ::=    'BETWEEN'
<not_between>           ::=    'NOT_BETWEEN'
<not_begins_with>       ::=    'NOT_BEGINS_WITH'
<not_ends_with>         ::=    'NOT_ENDS_WITH'
<is_today>              ::=    'IS_TODAY'
<is_not_today>          ::=    'IS_NOT_TODAY'
<within_last_days>      ::=    'WITHIN_LAST_DAYS'
<within_last_months>    ::=    'WITHIN_LAST_MONTHS'
<within_next_days>      ::=    'WITHIN_NEXT_DAYS'
<within_next_months>    ::=    'WITHIN_NEXT_MONTHS'
<not_within_last_days>  ::=    'NOT_WITHIN_LAST_DAYS'
<not_within_last_months>::=    'NOT_WITHIN_LAST_MONTHS'
<not_within_next_days>  ::=    'NOT_WITHIN_NEXT_DAYS'
<not_within_next_months>::=    'NOT_WITHIN_NEXT_MONTHS'
<contains_or_below>     ::=    'CONTAINS_OR_BELOW'
<contains_or_above>     ::=    'CONTAINS_OR_ABOVE'
```

Apache HTTPサーバーおよびPHPのリビルド

ここでは、SAでApache HTTPサーバーとPHPをリビルドして置き換える方法を説明します。SAにはApache HTTPサーバーとPHPが付属しているので、この情報が必要なのは、別のバージョンのApache HTTPサーバーを使用するか、他のライブラリやモジュールをコンパイルしてPHPに追加する必要がある場合に限られます。

SAでは、Web自動化プラットフォーム拡張 (APX) 用にApache HTTPサーバーとPHPが使用されます。詳細については、「自動化プラットフォーム拡張 (APX)」(97ページ)を参照してください。

APX HTTP環境の拡張

ここでは、Apache HTTPサーバーとPHPのリビルドによってAPX HTTP環境を拡張する方法を示します。

ここに示す作業は、コアアップグレードがすべて終了した後で行う必要があります。

マルチマスターメッシュを使用している場合、これらの作業はすべてのコアの各スライスに対して実行する必要があります。スライスコンポーネントバンドルの詳細については、HPE SSOポータルの『Server Automation管理ガイド』を参照してください。

PHPのリビルド

1. 次の作業を実行してPHPをリビルドします。

   PHPのソースを`http://www.php.net/` (英語サイト) からダウンロードします。
2. サーバー上のapxproxyがインストールされているディレクトリにソースを置きます。これは通常`/opt/opsware/apxproxy`.
3. 次のコマンドを入力します。ダウンロードしたPHPのバージョンが異なる場合はバージョン番号を書き換えてください。

```
mkdir /build ; cp php-4.4.8.tar.gz /build; cd /build
gzip -dc php-4.4.8.tar.gz | tar xvf -
cd php-4.4.8
./configure --prefix=/opt/opsware/apxphp
--with-pear=/opt/opsware/apxphp/lib/pear
```

```
--with-config-file-path=/opt/opsware/apxphp/lib
--with-apxs2=/opt/opsware/apxhttpd/bin/apxs <any other options you>
make clean
make
```

4. libphp4.soの古いコピーをバックアップします。

   ```
   cp /opt/opsware/apxhttpd/modules/libphp4.so
   /opt/opsware/apxhttpd/modules/libphp4.so.backup
   ```

5. 新しいlibphp4.soファイルをapxhhtpsディレクトリにコピーします。

   ```
   cp libs/libphp4.so /opt/opsware/apxhttpd/modules/libphp4.so
   ```

6. 参照ライブラリ全体がtool.listに存在することを確認します。

   ```
   ldd ./libs/libphp4.so
   ```

   出力の各エントリに対して、ファイルが
   /etc/opt/opsware/ogfs/tool.listに存在することを確認します。

   存在しないエントリがあれば、それを追加します。

7. apxphpフォルダーをバックアップします。

   ```
   mv /opt/opsware/apxphp /opt/opsware/apxphp.orig
   ```

8. PHPをインストールします。

   ```
   make install
   ```

9. OGFSを再ロードして再リンクし、/etc/opt/opsware/ogfs/tools.listに追加したものがすべてOGFSに表示されることを確認します。

   ```
   /opt/opsware/ogfs/tools/rewink && /opt/opsware/ogfs/
   tools/reload
   ```

10. apxproxyを再起動します。

    ```
    /etc/opt/opsware/startup/apxproxy restart
    ```

Apacheのリビルド

次の作業を実行してApache HTTPサーバーをリビルドします。

1. Apache HTTPサーバーのソースコードをhttp://httpd.apache.org/ (英語サイト) からダウンロードします。
2. サーバー上のスライスコンポーネントバンドルがあるディレクトリにソースを置きます。スライスコンポーネントバンドルの詳細については、HPE SSOポータルの『Server Automation管理ガイド』を参照してください。
3. 次のコマンドを入力します。ダウンロードしたhttpdのバージョンが異なる場合はバージョン番号を書き換えてください。

   ```
   mkdir /build; cp httpd-2.2.8.tar.gz /build; cd /build
   gzip -dc httpd-2.2.8.tar.gz | tar xf -
   cd httpd-2.2.8
   ./configure --prefix=/opt/opsware/apxhttpd <その他必要なオプション>
   ```

   SAでは現在次のオプションが使用されています。

   ```
   --enable-mods-shared="actions alias auth_basic auth_digest authn_file authz_
   user cgi deflate dir dumpio env expires headers ident logio log_config mime
   negotiation rewrite userdir vhost_alias imagemap status"
   --disable-dav
   --with-port=8021
   --with-expat=builtin
   --without-pgsql
   ```

   (SunOSのみ) 次のコマンドを入力します。

   ```
   perl -pi -e 's/#define HAVE_GETADDRINFO 1/#undef HAVE_GETADDRINFO/g'
   ./srclib/apr/include/arch/unix/apr_private.h
   make
   ```

4. apxhttpディレクトリのバックアップを作成します。

   ```
   mv /opt/opsware/apxhttpd /opt/opsware/apxhttpd.orig
   ```

5. Apacheをインストールします。

   ```
   make install
   ```

6. 新しいファイルをOGFSに再ロードして再リンクします。

   ```
   /opt/opsware/ogfs/tools/rewink && /opt/opsware/ogfs/tools/reload
   ```

7. モジュールディレクトリのHTTPDおよび.soファイルは、外部ライブラリを参照している可能性があります。これらのライブラリはOGFSから見える (またはリンクされている) 必要があります。

OGFSにログインし、/opt/opsware/apxhttpd/bin/httpdと、/opt/opsware/apxhttpd/modules にある.soファイルに対してLDDを実行して、表示されるすべてのファイルがOGFSに存在することを確認します。存在しない場合、ファイルを/etc/opt/opsware/ogfs/tool.list (OGFS外部)に追加してから、ステップ6を再実行して、すべてのファイルが/opt/opsware/apxhttpd/bin/httpdで使用できるようにします。

8. その後、PHPをリビルドする必要があります。「PHPのリビルド」(221ページ)を参照してください。

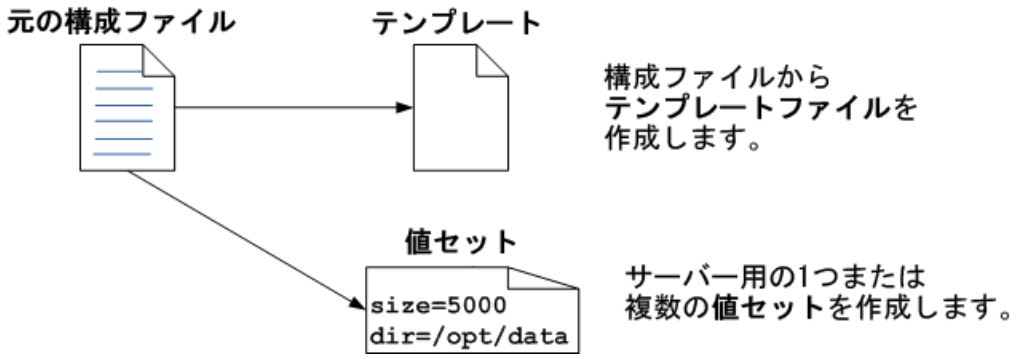
アプリケーション構成

この項では、アプリケーション構成の概要を紹介します。アプリケーション構成のセットアップと管理に必要な手順について説明します。

- これらの作業の詳細な実行方法については、「アプリケーション構成のタスク」(259ページ)を参照してください。
- アプリケーション構成の背景情報については、「アプリケーション構成の概念」(228ページ)を参照してください。

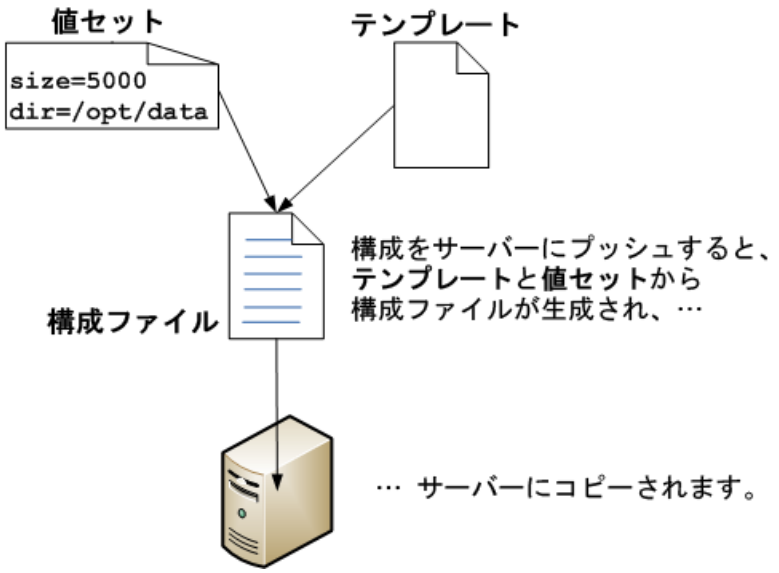
アプリケーション構成を作成して使用するには、まず次の図に示すように**テンプレート**ファイルと**値セット**を作成する必要があります。テンプレートは、構成ファイルのモデルです。値セットは、サーバーにプッシュされる構成ファイルのインスタンスを作成するために、テンプレートとマージされるデータ値です。手順については次の図を参照してください。

アプリケーション構成の作成



テンプレートと値セットを作成したら、次の図に示すように、アプリケーションを1つ以上のサーバーに**アタッチ**し、構成をサーバーに**プッシュ**することができます。手順については次の図を参照してください。

アプリケーション構成のプッシュの図



アプリケーション構成を作成して使用する方法

1. **管理する構成ファイルを決めます。**管理するアプリケーションまたはシステム構成ファイルを選択します。たとえば、Apache Webサーバーの場合、http.conf、password.conf、obj.conf、mimetypes、magnus.confファイルを管理することができます。
2. **各構成ファイルに対するテンプレートファイルを作成します。**管理する構成ファイルのそれぞれに対して、構成ファイルに基づいてテンプレートファイルを作成します。テンプレートは、元の構成ファイルのモデルであり、サーバーごとに異なる値がプレースホルダーで表されています。テンプレートは、構成ファイルのどの値が固定であり、どの値が可変でサーバーによって異なるかを指定する役割を果たします。次のトピックも参照してください。
   - 「アプリケーション構成の作成」(261ページ)
   - 「ビジュアルエディターによる構成テンプレートの作成」(264ページ)
   - XML構成ファイルに対しては、XMLまたはXML-DTDテンプレートファイルを作成します。詳細については、「XML構成ファイルの管理」(301ページ)を参照してください。
   - その他の構成ファイルに対しては、構成モデリング言語 (CML) を使用してテンプレートファイルを作成します。詳細については、「CMLリファレンス」(375ページ)および「CMLチュートリアル1 - 単純なWebアプリケーションサーバーに対するアプリケーション構成の作成」(327ページ)を参照してください。
   - 「構成テンプレートとスクリプトテンプレート」(230ページ)

3. **各スクリプトに対するテンプレートファイルを作成します。**構成ファイルを更新する前または後に、スクリプトを実行することが必要な場合があります。この場合、CMLを使用してスクリプトからテンプレートファイルを作成する必要があります。構成ファイルでスクリプトを実行する必要がない場合は、このステップはスキップします。参照情報:
   - 「スクリプトからのテンプレートの作成」(283ページ)
   - 「構成テンプレートとスクリプトテンプレート」(230ページ)
   - 「アプリケーション構成でのスクリプトの実行」(247ページ)
4. **テンプレートをSAライブラリにインポートします。**構成ファイルとスクリプトからすべてのテンプレートを作成したら、テンプレートをSAライブラリにインポートします。または、SAクライアントで直接作成することもできます。詳細については、「テンプレートファイルのインポートと検証」(280ページ)を参照してください。
5. **アプリケーション構成オブジェクトを作成します。**アプリケーション構成オブジェクトは、1つ以上のテンプレートを保持する単なるコンテナーです。参照情報:
   - 「アプリケーション構成の作成」(261ページ)
   - 「アプリケーション構成オブジェクト」(229ページ)
6. **テンプレートをアプリケーション構成オブジェクトに追加します。**テンプレートを作成してSAライブラリにインポートしたら、アプリケーション構成オブジェクトに追加します。また、ファイルがインストールされる順序と、スクリプトがある場合はそれらが実行されるタイミングも指定します。詳細については、「アプリケーション構成に対するテンプレートの追加または削除」(282ページ)を参照してください。
7. **アプリケーション構成をサーバーにアタッチします。**アプリケーション構成を作成して構成したら、それが使用されるサーバーにアタッチします。これにより、アプリケーション構成のインスタンスが作成されます。アプリケーション構成のインスタンスは、1つのサーバー上に複数存在することができます。各インスタンスは、それぞれ異なる目的に使用されます。たとえば、アプリケーションのステージングバージョンを構成するインスタンスと、アプリケーションのプロダクションバージョンを構成するインスタンスを使用することができます。また、サーバー上でアプリケーションのインスタンスが3つ実行されている場合は、各インスタンスを構成するためにアプリケーション構成の3つのインスタンスを使用することができます。詳細については、次のトピックを参照してください。
   - 「サーバーまたはデバイスグループへのアプリケーション構成のアタッチ」(286ページ)
   - 「ソフトウェアポリシーでのアプリケーション構成」(258ページ)
8. **サーバーに対する値セットを作成します。**サーバーにプッシュされる実際の構成ファイルを生成するためにテンプレートに記入される値を設定します。複数のレベルでデフォルト値を設定して、下のレベルでオーバーライドされない限り継承されるようにすることができます。詳細については、次のトピックを参照してください。
   - 「値セットエディターでの値の設定」(235ページ)
   - 「値セット」(232ページ)

9. **実際の構成ファイルと構成テンプレートを比較します。** (オプション) 構成テンプレートとサーバー上の実際の構成ファイルを比較して、違いがあるかどうかを確認します。これにより、サーバー上の構成ファイルの内容と、アプリケーション構成をサーバーにプッシュしたときにサーバーにコピーされる値を見ることができます。詳細については、次のトピックを参照してください。
   - 「構成テンプレートとターゲット構成ファイルの比較 - プレビュー」(296ページ)
   - 「構成テンプレートの比較 」(298ページ)
10. **構成の変更をプッシュします。** サーバー上の構成ファイルを更新するには、アプリケーション構成をサーバーに「プッシュ」します。サーバーに変更をプッシュしない限り、サーバー上の実際の構成ファイルが変更されることはありません。アプリケーション構成の変更は、個々のサーバーにプッシュすることも、サーバーのグループにプッシュすることもできます。詳細については、次のトピックを参照してください。
   - 「アプリケーション構成のプッシュ」(290ページ)
   - 「アプリケーション構成のサーバーへのプッシュ」(249ページ)
11. **構成を監査し、コンプライアンスを監視して修復します。** 構成を監査し、コンプライアンスを監視して、変更された構成を修復する方法については、「アプリケーション構成コンプライアンス」(250ページ)とHPE SSOポータルの『Server Automation管理ガイド』を参照してください。

アプリケーション構成の概念

Server Automationでは、XML構成ファイルなどの構成ファイルを1か所でまとめて管理し、データセンターで複数のサーバー間の変更および更新を容易に反映することができます。UNIXシステム上の/etc/hostsファイルのような単独の構成ファイルを管理したり、WebLogic、Websphereなどの大規模なビジネスアプリケーションに関連付けられている構成ファイルなどのような1つのアプリケーションに関連付けられている複数の複雑な構成ファイルを管理したりすることができます。

構成ファイルとは、サーバー上にある、内容を制御する必要があるファイルのことです。これには、アプリケーションの構成ファイルの他に、次のようなシステム構成ファイルも含まれます。

- アプリケーションサーバー、Webサーバー、データベース、またはその他のアプリケーションの動作を制御するファイル。たとえば、複数の異なるサーバーで動作しているApache Webサーバーのhttpd.confファイルを管理できます。このファイルに対してアプリケーション構成をアタッチし、各サーバー上のそのファイルの各インスタンスに対して特定の値を入力します。
- UNIXサーバー上の/etc/hosts、/etc/fstab、/etc/passwd、/etc/groupsなど、サーバー上のファイル。
- サーバーの構成に使用されるサーバー上のその他のファイル。

これに加えて、アプリケーション構成に対して次のことを実行できます。

- 構成ファイルをサーバー上に配置する前後に**スクリプトを実行**します。たとえば、構成をサーバー上にプッシュした後でアプリケーションを再起動するスクリプトを使用できます。詳細については、「アプリケーション構成でのスクリプトの実行」(247ページ)を参照してください。
- **ソフトウェアポリシー**の中で、アプリケーションのデプロイメントおよび継続的管理の一部としてアプリケーション構成を使用します。詳細については、「ソフトウェアポリシーでのアプリケーション構成」(258ページ)を参照してください。
- **監査**を行って、サーバーが必要なアプリケーション構成ファイルの内容に一致しているかどうかを判定します。詳細については、「アプリケーション構成の監査」(257ページ)を参照してください。
- 定義済みのアプリケーション構成に基づいて、サーバーの**コンプライアンスをチェック**します。詳細については、「アプリケーション構成コンプライアンス」(250ページ)を参照してください。
- 以前の構成ファイルを**復元**します。詳細については、「構成ファイルの過去の状態への復元」(293ページ)を参照してください。

アプリケーション構成オブジェクト

サーバーで構成ファイルを管理するには、SAライブラリでアプリケーション構成オブジェクトと少なくとも1つの構成テンプレートを作成する必要があります。アプリケーション構成オブジェクトは、テンプレートとオプションスクリプトのための単なるコンテナーです。

アプリケーション構成を1つまたは複数のサーバーにプッシュすると、SAは次のことを行います。

- 構成テンプレートと指定された値セットから構成ファイルを生成します。
- 生成した構成ファイルをサーバーにコピーします。

次の図に示すアプリケーション構成の例には、“exports.tpl” という名前のテンプレートと、“post-exports.sh” という名前のUnixシェルスクリプトファイルが含まれます。ファイル拡張子 “.tpl” はテンプレートを示します。

構成テンプレートとシェルスクリプトを含むアプリケーション構成

構成テンプレートとスクリプトテンプレート

構成テンプレートとは、構成ファイルのモデルであり、サーバーごとに異なるデータ値の代わりに変数が使用されます。また、プッシュ操作中にテンプレートや値セットから構成ファイルを再作成する方法を記述した命令も格納されます。構成テンプレートは、構成ファイルの固定部分と可変部分を定義します。テンプレートにはまた、構成ファイルを解析するための指示と、構成ファイルを再作成して管理対象サーバーにプッシュするための指示も含まれます。

構成テンプレートファイルの作成には、SA構成モデリング言語 (CML) またはXMLを使用します。XMLベースの構成ファイルにはXMLを、それ以外の構成ファイルにはCMLを使用します。CMLの詳細については、「CMLリファレンス」(375ページ)を参照してください。XML構成ファイルの詳細については、「XML構成ファイルの管理」(301ページ)を参照してください。

最終的な構成ファイルの生成に用いられる値を定義する必要があります。これらの値は、SAデータベースに値セットとして保存されます。構成テンプレートのCMLまたはXMLは、値セットの値をテンプレートファイルとマージして、ターゲット構成ファイルを生成します。詳細については、「値セット」(232ページ)を参照してください。

また、構成テンプレートを使用して、構成ファイルからデータをインポートして値セットを作成することもできます。詳細については、「テンプレートファイルのインポートと検証」(280ページ)を参照してください。

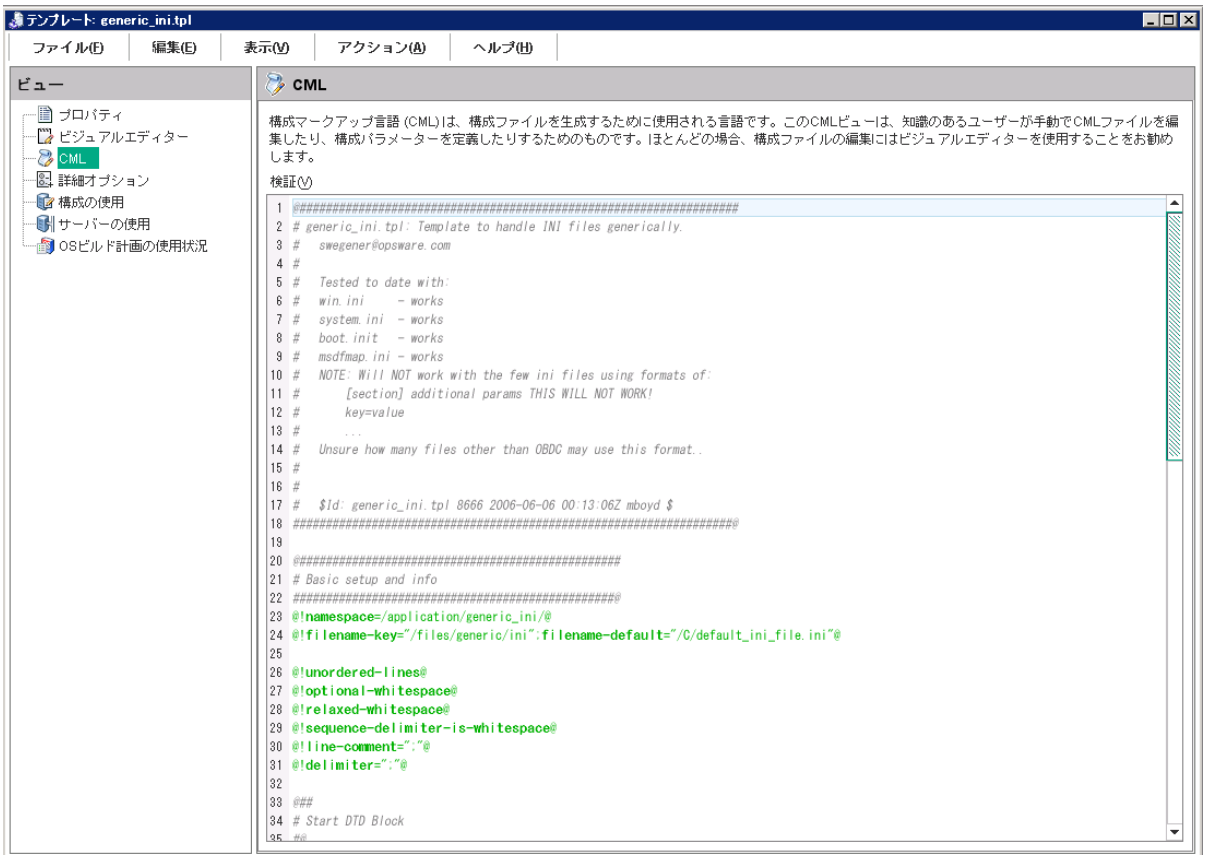
構成ファイルを管理するには、管理対象の各ファイルに対して構成テンプレートを作成し、アプリケーション構成に追加します。詳細については、「アプリケーション構成の作成」(261ページ)を参照してください。

CML構成テンプレート

構成モデリング言語 (CML) を使用すると、構成ファイルのテンプレートを作成し、それを値セットとマージすることで実際の構成ファイルを生成できます。次の図に示すのは、構成モデリング言語 (CML) による構成テンプレートファイルの例です。

詳細については、「CMLチュートリアル1 - 単純なWebアプリケーションサーバーに対するアプリケーション構成の作成」(327ページ)、「CMLチュートリアル2 - Webサーバー構成ファイルのテンプレートの作成」(338ページ)、「CMLリファレンス」(375ページ)を参照してください。

次に示すのは、CMLで作成された構成テンプレートの例です。



XMLおよびXML-DTD構成テンプレート

管理対象サーバー上のXML構成ファイルを管理することもできます。XML構成テンプレートを使用すると、XML構成ファイルの値をモデル化し、これらの値をターゲットサーバー上の実際のXMLと照合して、変更をターゲットファイルにプッシュすることができます。DTDを使用するXML構成ファイルだけでなく、DTDを使用しないものもモデル化できます。

XML構成テンプレートの機能はCMLテンプレートと似ていますが、コメントに定義されているいくつかのアプリケーション構成オプションが使用されます。XML構成テンプレートでは、タグを使用して、ファイルがSAクライアントに表示される方法をカスタマイズすることもできます。

詳細については、「XML構成ファイルの管理」(301ページ)を参照してください。

スクリプトテンプレート

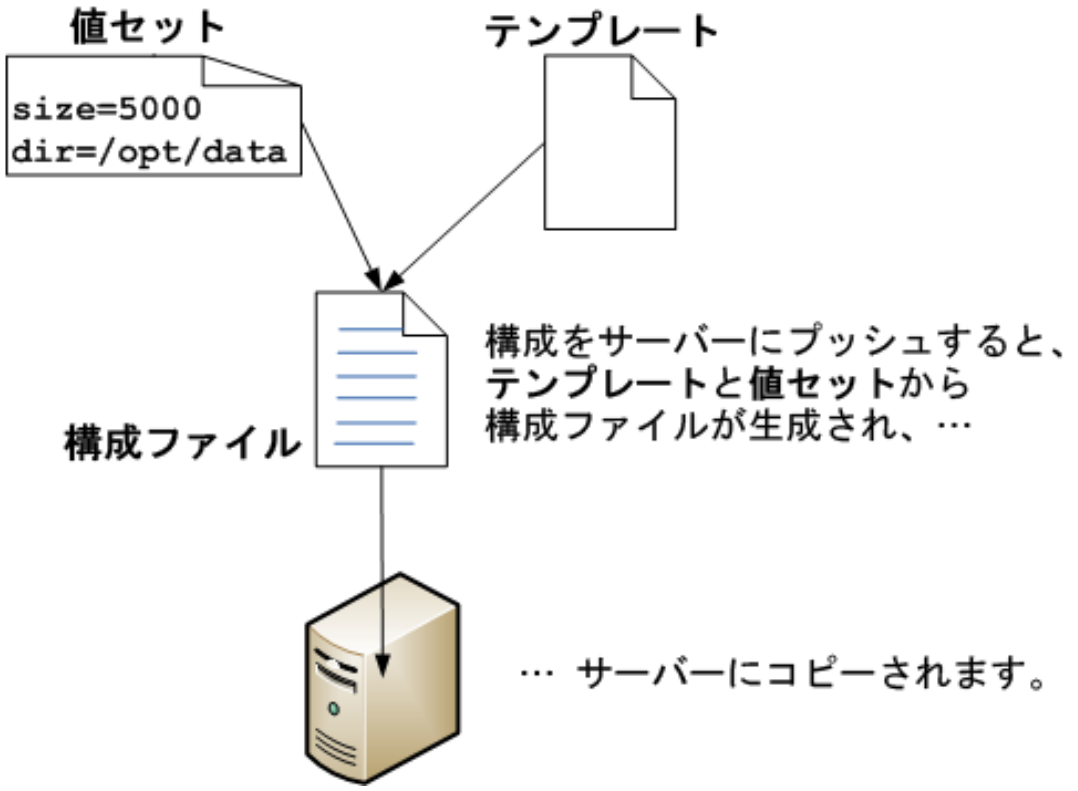
構成値がターゲットサーバーにコピーされる前または後に実行されるスクリプトをアプリケーション構成に追加できます。アプリケーション構成オブジェクトにスクリプトを含めるには、スクリプトをCMLテンプレートにコ

ピーしてから、スクリプトテンプレートをアプリケーション構成オブジェクトにインポートする必要があります。

詳細については、「アプリケーション構成でのスクリプトの実行」(247ページ)および「スクリプトからのテンプレートの作成」(283ページ)を参照してください。

値セット

値セットとは、ターゲット構成ファイルを生成するためにテンプレートファイルとマージされるデータ値のセットです。結果の構成ファイルは、サーバーにプッシュできます。



最も単純なケースでは、特定のサーバーの「サーバーインスタンス」レベルで値セットを定義します。「サーバーインスタンス」値セットの値は、構成テンプレートとマージされて、そのサーバーにプッシュされる実際の構成ファイルを生成します。

値セットのレベルと値セットの継承

個々のサーバーに対してサーバーインスタンスレベルで値を設定する方法は、少数のサーバーに対してはうまく行きますが、サーバーの数が多い場合には、値セットの継承を使用して、上位レベルでデフォルト値を設定し、下位レベルに継承することができます。レベルが上がるほど、関連する管理対象サーバーの数が多くなります。各レベルでは、明示的に継承をブロックしているか、継承した値をオーバーライドする値を設定していない限り、上のレベルから値が継承されます。

次の表は値セットの継承レベルの一覧であり、下位レベルに継承されるかオーバーライドされるか、および各レベルでの値の設定方法を示します。各レベルでの値の設定方法の詳細は、この後で説明します。

値セットのレベルと値セットの継承

レベル	レベルの説明	値の設定方法
アプリケーション	このレベルは、アプリケーション構成自体に適用される値を定義します。これは、下のどれかのレベルによってオーバーライドされない限り、アプリケーション構成がアタッチされたすべてのサーバーに適用されます。	アプリケーション構成オブジェクトを開きます。[内容] > [アプリケーション値] を選択します。テンプレートを選択します。[ファイルの値] ビューを選択します。「アプリケーションレベルでの値の設定」(240ページ)を参照してください。
ファシリティ	このレベルは、ファシリティの値を定義します。これは、下のどれかのレベルによってオーバーライドされない限り、指定されたファシリティの、アプリケーション構成がアタッチされたすべてのサーバーに適用されます。	アプリケーション構成オブジェクトを開きます。[内容] > [ファシリティ値] > <ファシリティ> を選択します。テンプレートを選択します。[ファイルの値] ビューを選択します。「ファシリティレベルでの値の設定」(241ページ)を参照してください。
カスタマー	このレベルは、カスタマーの値を定義します。これは、下のどれかのレベルによってオーバーライドされない限り、指定されたカスタマーに属する、アプリケーション構成がアタッチされたすべてのサーバーに適用されます。	アプリケーション構成オブジェクトを開きます。[内容] > [カスタマー値] > <カスタマー> を選択します。テンプレートを選択します。[ファイルの値] ビューを選択します。「カスタマーレベルでの値の設定」(241ページ)を参照してください。
グループ	このレベルは、デバイスグループの値を定義します。これは、下のどれかのレベルによってオーバーライドされない限り、指定されたデバイスグループの、アプリケーション構成がアタッチされたすべてのサーバーに適用されます。	デバイスグループを開きます。[構成されるアプリケーション] > <アプリケーション構成名 > を選択します。テンプレートを選択します。「グループレベルでの値の設定」(243ページ)を参照してください。
グループインスタンス	このレベルは、アプリケーション構成の特定の1つのインスタンスの値を定義します。これは、下のどれかのレベルによってオーバーライドされない限り、指	デバイスグループを開きます。[構成されるアプリケーション] > <アプリケーション構成名 > > <インスタンス名 > を選択します。テンプレートを

値セットのレベルと値セットの継承 (続き)

レベル	レベルの説明	値の設定方法
	定されたデバイスグループの、そのインスタンスがアタッチされたすべてのサーバーに適用されます。	選択します。「グループインスタンスレベルでの値の設定」(244ページ)を参照してください。
サーバー	このレベルは、サーバーの値を定義します。これは、下のレベルによってオーバーライドされない限り、指定されたサーバーのアプリケーション構成のすべてのインスタンスに適用されます。	サーバーを開きます。[管理ポリシー] タブを選択します。[構成されるアプリケーション] > <アプリケーション構成名 > を選択します。テンプレートを選択します。「サーバーレベルでの値の設定」(246ページ)を参照してください。
サーバーインスタンス	このレベルは、サーバー上のアプリケーション構成の特定の1つのインスタンスの値を定義します。これは、指定されたサーバーの指定されたアプリケーション構成インスタンスだけに適用され、上のすべてのレベルをオーバーライドします。	サーバーを開きます。[管理ポリシー] タブを選択します。[構成されるアプリケーション] > <アプリケーション構成名 > > <インスタンス名 > を選択します。テンプレートを選択します。「サーバーインスタンスレベルでの値の設定」(246ページ)を参照してください。

次の表は、アプリケーション構成値の継承方法を示します。各行の値は、そのレベルで設定される値を示します。一番下の行は、実際にサーバーにプッシュされる値を示します。

表 2: アプリケーション構成値の継承

レベル	各レベルで設定される値						
アプリケーション	2	1	Z	Y	X	W	V
ファシリティ		U	T	S	R	Q	P
カスタマー			O	N	M	L	K
グループ				J	I	H	G
グループインスタンス					F	E	D
サーバー						C	B
サーバーインスタンス							A
継承された結果	2	U	O	J	F	C	A

継承のブロック

継承は任意のレベルでブロックできます。このためには、値セットエディターで任意の変数に対して [継承のブロック] を選択します。ブロックされたレベルより上で設定されたすべての値は、継承されません。ブロックされたレベルより下のレベルの値は、引き続き継承されます。下のレベルで値を設定しないと、変数は値を持たなくなります。

1. 任意のレベルで値セットエディターを開きます。
2. 任意の変数の値列で、値を選択します。編集ボックスの右端に、ドロップダウンリストと [...] ボタンが表示されます。
3. ドロップダウンリストまたは [...] ボタンを選択します。変数の値に関するいくつかの選択肢が表示されます。
4. [**継承のブロック**] を選択します。
5. [**ファイル**] > [**保存**] または [**変更の保存**] ボタンを選択します。

詳細については、「値セットエディターでの値の設定」(235ページ)を参照してください。

値セットエディター

値セットエディターでは、テンプレートに定義されている変数が表示され、変数の値を設定できます。値セットエディターは、任意の継承レベルでアプリケーション構成の値セットを選択すると表示されます。この項では、値セットエディターの使用方法を説明します。

値セットエディターでの値の設定

値セットエディターでは、変数のデータ型に一致する任意の値を変数に設定できます。目的の変数の隣の [値] 列に値を入力します。

この他に、値を設定するには次の方法があります。

- 編集ボックスの右端のドロップダウンリストを選択して、次のいずれかの値を選択します。
  - 空文字列: 長さ0の文字列を値セットに設定することを指定します。
  - 継承のブロック: 上位レベルの値を継承しないことを指定します。下のレベルで値を設定しないと、変数は値を持たなくなります。

- エージェントバージョン、認証ドメイン、シャーシID、カスタマーID、カスタマー名など: 管理対象サーバーに関する選択した情報を値セットに設定することを指定します。

- 編集ボックスの右側の[...]ボタンを選択して、次の選択肢を表示します。
  - 値なし: 値セットで値を空白のままにします。
  - 継承のブロック: 上位レベルの値を継承しないことを指定します。下のレベルで値を設定しないと、変数は値を持たなくなります。
  - 任意の値: 任意の値を入力します。
  - オブジェクト属性: 上のリストの値のうち1つを選択します。
  - カスタム属性: カスタム属性を入力します。
  - デプロイメント自動化の値: デプロイメント自動化で管理されるアプリケーションからの値を入力します。
- 編集ボックスを右クリックして、編集メニューを表示します。

値セットエディターでのフィールドの設定

どのレベルで値を設定する場合でも、値セットエディターには、編集中の値セットに対する次に示すフィールドが表示されます。

「サーバーレベルの値セットエディター」(245ページ)の図は、“WAS-app-config” という名前のアプリケーション構成がアタッチされたサーバーを示しています。サーバーインスタンスレベルの値セットが表示されており、編集できます。この図には次のフィールドの例が示されています。

- **テンプレート**: アプリケーション構成オブジェクトに含まれるテンプレートファイルのリストが表示され、テンプレートを選択して表示し、変更できます。表示するテンプレートを選択します。
- **ファイル名**: 構成テンプレートのターゲットとなる管理対象サーバー上の構成ファイルの名前を指定します。ファイル名が設定されていない場合、ファイル名は継承されます。アプリケーション構成階層のどこでもファイル名が設定されていない場合、構成テンプレートに記載されたファイル名が使用されます。
  サーバー上にアプリケーションの複数のインスタンスが存在する場合、このフィールドを使用して、各ターゲット構成ファイルのフルパスを指定します。
- **エンコード**: ターゲット構成ファイルの文字エンコードを指定します。デフォルトは管理対象サーバーで使用されているエンコードです(なお、SAクライアントではUTF-16エンコードはサポートされていません)。
- **形式の保持**: ターゲット構成ファイルのスペース、コメント、順序を保持するかどうかを指定します。SAはターゲット構成ファイルのできるだけ多くの部分を保持しようとしますが、一部のコメントや形式指定

は保持できない場合があります。このオプションは、テンプレートで@!partial-template@ CMLタグが使用されている場合に必要です。
これをオンにしないと、すべてのコメントと形式指定がファイルから削除され、テンプレートのデフォルトの順序とスペースが用いられます。

> **注:** XMLベースのテンプレートの場合、形式の保持を指定しても、XMLタグ内部の空白と属性の順序は保持されません。形式の保持を指定した場合、タグ自体の中の空白と、タグ内部の属性の順序を除いて、空白と順序は保持されます。プッシュ後には、タグ内部の余分な空白は除去され、属性の順序は変更される場合があります。空白を除去し、属性の順序を変更しても、XMLの意味は変わりません。

- **値の保持**: 値セットに対応する値がない場合に、管理対象サーバー上のターゲット構成ファイルに含まれる値を保持するどうかを指定します。デフォルトでは、このオプションはオフになっています。

  [値の保持]を使用すると、サーバー上の構成ファイル中の値のうち、SAデータベースに保存されていないものを保持するように指定できます。これは、構成ファイルの現在の値をすべてSAにインポートするつもりがない場合や、ユーザーまたはプログラムによってSA外部で構成ファイルが変更される可能性がある場合に便利です。これにより、構成ファイルの変更の一部をSA外部で管理することができます。
- **継承された値の表示**: 上位レベルから継承された値を含めて、結果の値セットを表示します。これをオフにした場合、現在のレベルで設定された値のみを表示します。このビューは読み取り専用であり、サーバーインスタンスレベルで値セットを表示した場合のみ使用できます。

値セットエディターの列

値セットを編集する際に、SAクライアントには、構成テンプレートの値に関する次の情報が表示されます。

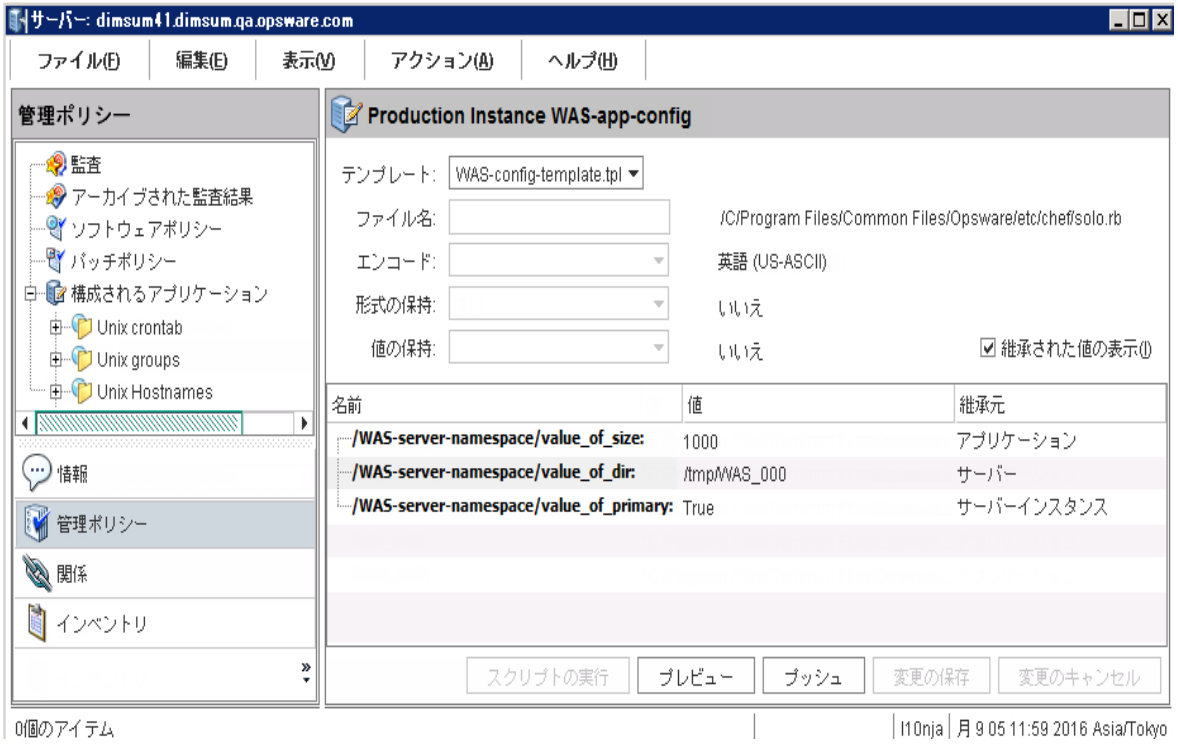
「サーバーレベルの値セットエディター」(245ページ)の図は、“WAS-app-config” という名前のアプリケーション構成のインスタンスがアタッチされたサーバーを示しています。サーバーインスタンスレベルの値セットが表示されており、編集できます。下には、[名前]、[値]、[継承元]の列が示されています。

- **名前**: テンプレートファイル内の変数の名前を示します。名前は、シンプルタイプ、シンプルタイプのリスト、または多次元リストです。多次元リストは、その親の下に表示されます。必須の要素は太字で表示されます。多次元リストをダブルクリックすると、表示と非表示を切り替えることができます。リストタイプの値にエントリを追加するには、親を右クリックして、[**アイテムの追加**]を選択します。必須フィールドは、構成テンプレートで設定されます。必須フィールドは空にすることができません。空である場合、アプリケーション構成のプレビューやプッシュは実行できません。

- **値**: 表示中の値セットの値を示します。リテラル値を入力することも、サーバーの設定から属性 (カスタマー名、カスタマーID、シャーシID、デバイスIDなど) を選択することもできます。設定を空白のままにした場合には、その親または先祖から設定が継承されます (その親または先祖に値が構成されている場合)。値に関連するカスタム属性に値を設定するには、[...] ボタンをクリックして、[値の設定] ダイアログボックスを使用します。
- **継承元**: 値の継承元を示します。この列が表示されるのは、サーバーインスタンスレベルを表示しており、[継承された値の表示] が選択されている場合だけです。

  [値の保持] オプションが設定されている場合、サーバー上の構成ファイルが継承階層の最も外側のレベルになります。すなわち、値セットに値が存在しない場合には、ファイル内の値が保持されます。

サーバーインスタンスレベルの値セットエディター



既存の構成ファイルからの値セットのインポート

値セットの値は手動で設定することもできますが、管理対象サーバー上の既存の構成ファイルから値セットに値をインポートすることもできます。

既存の構成ファイルから値セットに値をインポートするには、次の手順を実行します。

1. 値をインポートするレベルで値セットエディターを表示します。インポートオプションを利用できるのは、(サーバー) インスタンスレベルに限定されます。各レベルで値セットエディターを表示する方法については、この後の各項を参照してください。
2. 値セットエディターで、インポートする値を含む構成ファイルの絶対ファイル名が [ファイル名] フィールドに表示されていることを確認します。
3. [値] 列を右クリックして、[**値のインポート**] を選択します。構成テンプレートのすべての値が、実際の構成ファイルからの値に置き換えられます。

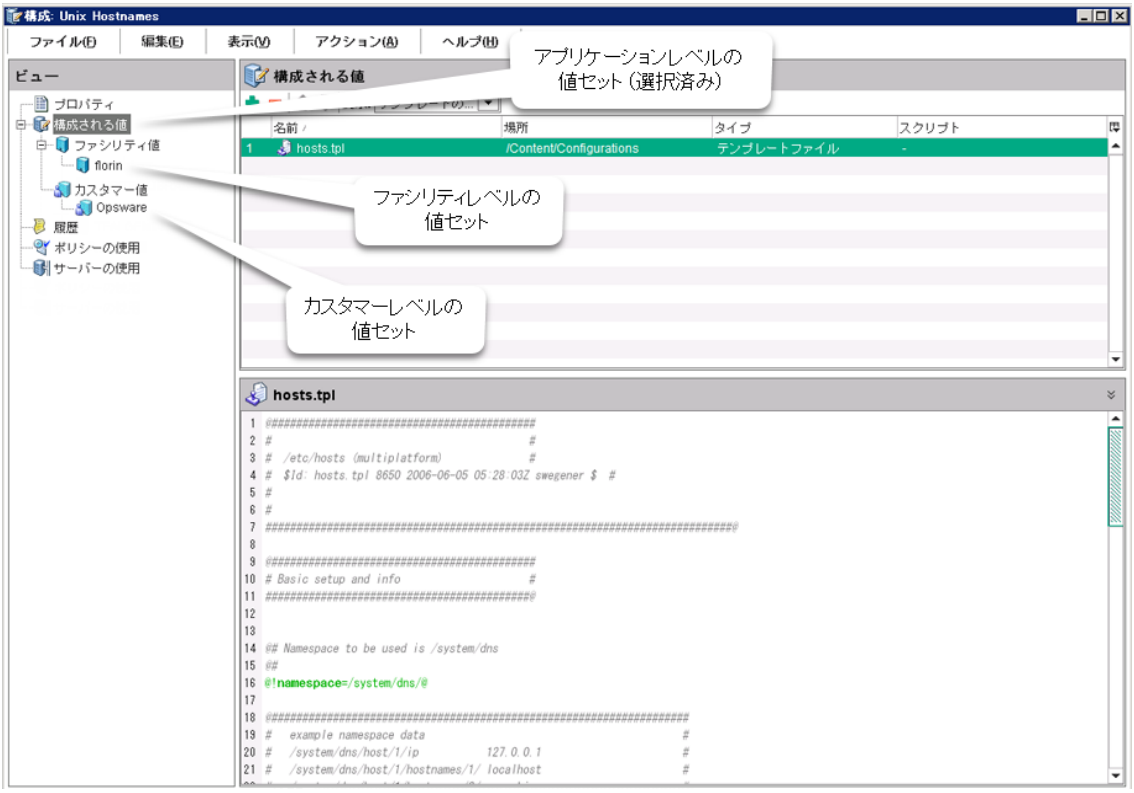
   [**値のインポート**] メニュー項目を選択すると、管理対象サーバー上の既存の構成ファイルが読み取られ、値が解析されて、選択したレベルに保存されます。値をインポートした後で、必要な場合は、任意の値を変更して、変更をサーバーにプッシュすることもできます。
4. [**変更の保存**] を選択します。

アプリケーション、ファシリティ、カスタマーレベルの値セットエディター

「アプリケーション、ファシリティ、カスタマーレベルの値セットエディター - アプリケーションレベルを選択」(239 ページ)の図は、アプリケーション、ファシリティ、カスタマーレベルの値セットエディターを示します。このビューは、アプリケーション構成オブジェクトを開いた場合のみ使用できます。

- [構成される値] を選択すると、アプリケーションレベルで値セットエディターが表示されます。
- [ファシリティ値] の下でファシリティを選択すると、ファシリティレベルで値セットエディターが表示されます。
- [カスタマー値] の下でカスタマーを選択すると、カスタマーレベルで値セットエディターが表示されます。

アプリケーション、ファシリティ、カスタマーレベルの値セットエディター - アプリケーションレベルを選択



アプリケーションレベルでの値の設定

アプリケーションレベルは、アプリケーション構成自体に適用される値を定義します。これは、下のどれかのレベルによってオーバーライドされない限り、アプリケーション構成がアタッチされたすべてのサーバーに適用されます。

アプリケーションレベルで値を設定するには、次の手順を実行します。

アプリケーション、ファシリティ、カスタマーレベルの値セットエディターの例については、「アプリケーション、ファシリティ、カスタマーレベルの値セットエディター」(239ページ)を参照してください。

1. SAクライアントでアプリケーション構成オブジェクトを開きます。
2. [構成される値]ノードを選択します。
3. アプリケーション構成でテンプレートを選択します。
4. [表示]ドロップダウンリストから[ファイルの値]を選択します。右下に値セットエディターが表示され、アプリケーションレベルのデフォルト値を設定できます。
5. テキストボックスに値を入力します。その他の編集機能を利用するには、右クリックメニュー項目を使用します。

6. オプションで、[表示]ドロップダウンリストで[ファイルのプレビュー]を選択して、結果のファイルを表示します。
7. 変更内容を保存する場合は、[**ファイル**] > [**保存**]を選択します。

ファシリティレベルでの値の設定

ファシリティレベルは、ファシリティの値を定義します。これは、下のどれかのレベルによってオーバーライドされない限り、指定されたファシリティの、アプリケーション構成がアタッチされたすべてのサーバーに適用されます。

ファシリティレベルで値を設定するには、次の手順を実行します。

アプリケーション、ファシリティ、カスタマーレベルの値セットエディターの例については、「アプリケーション、ファシリティ、カスタマーレベルの値セットエディター」(239ページ)を参照してください。

1. SAクライアントでアプリケーション構成オブジェクトを開きます。
2. [構成される値]ビューを開いて、[ファシリティ値]ノードを表示します。
3. [ファシリティ値]ノードを開いて、ファシリティを選択します。
4. アプリケーション構成でテンプレートを選択します。
5. [表示]ドロップダウンリストから[ファイルの値]を選択します。右下に値セットエディターが表示されます。
6. テキストボックスに値を入力します。その他の編集機能を利用するには、右クリックメニュー項目を使用します。
7. オプションで、[表示]ドロップダウンリストで[**ファイルのプレビュー**]を選択して、結果の値を表示します。
8. 変更内容を保存する場合は、[**ファイル**] > [**保存**]を選択します。

カスタマーレベルでの値の設定

カスタマーレベルは、カスタマーの値を定義します。これは、下のどれかのレベルによってオーバーライドされない限り、指定されたカスタマーに属する、アプリケーション構成がアタッチされたすべてのサーバーに適用されます。

カスタマーレベルで値を設定するには、次の手順を実行します。

アプリケーション、ファシリティ、カスタマーレベルの値セットエディターの例については、「アプリケーション、ファシリティ、カスタマーレベルの値セットエディター」(239ページ)を参照してください。

1. SAクライアントでアプリケーション構成オブジェクトを開きます。
2. [構成される値]ビューを開いて、[カスタマー値]ノードを表示します。
3. [カスタマー値]ノードを開いて、カスタマーを選択します。
4. アプリケーション構成でテンプレートを選択します。
5. [表示]ドロップダウンリストから[ファイルの値]を選択します。右下に値セットエディターが表示されます。
6. テキストボックスに値を入力します。その他の編集機能を利用するには、右クリックメニュー項目を使用します。詳細については、「値セットエディターでの値の設定」(235ページ)も参照してください。
7. オプションで、[表示]ドロップダウンリストで[ファイルのプレビュー]を選択して、結果の値を表示します。
8. 変更内容を保存する場合は、[**ファイル**] > [**保存**]を選択します。

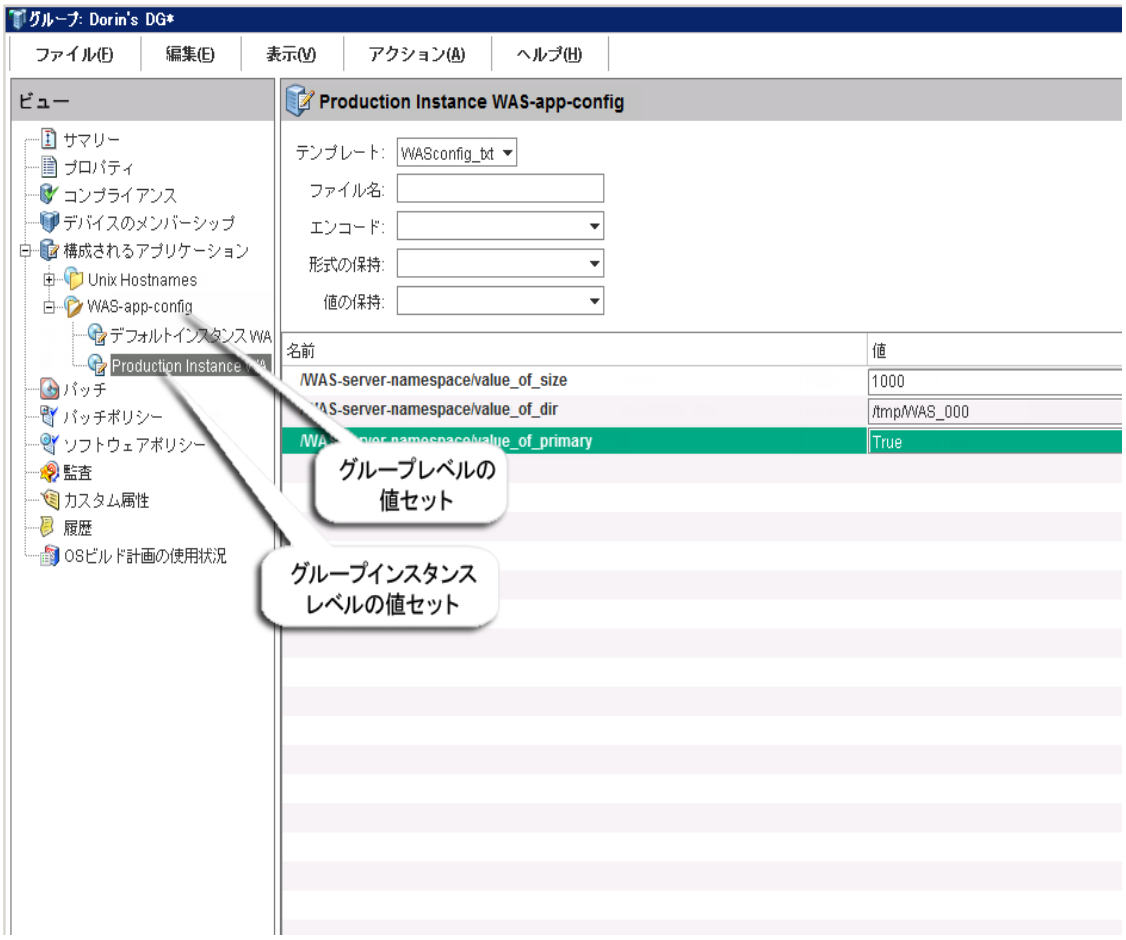
グループレベルの値セットエディター

グループレベルは、デバイスグループの値を定義します。これは、下のどれかのレベルによってオーバーライドされない限り、指定されたデバイスグループのすべてのサーバーに適用されます(アプリケーション構成がそのデバイスグループにアタッチされている場合)。

「グループレベルおよびグループインスタンスレベルの値セットエディター」(242ページ)は、グループおよびグループインスタンスレベルの値セットエディターを示します。グループおよびグループインスタンスレベルは、アプリケーション構成がデバイスグループにアタッチされている場合のみ使用できます。このビューが使用できるのは、アプリケーション構成がアタッチされているデバイスグループを開いたときだけです。詳細については、「サーバーまたはデバイスグループへのアプリケーション構成のアタッチ」(286ページ)を参照してください。

- “WAS-app-config” という名前のアプリケーション構成を選択すると、グループレベルで値セットエディターが表示されます。
- アプリケーション構成インスタンスの1つを選択すると、グループインスタンスレベルで値セットエディターが表示されます。この例には、“Production Instance WAS-appconfig” と“Staging Instance WAS-appconfig” という2つのアプリケーション構成インスタンスが示されています。値セットエディターには、“Production Instance WAS-appconfig” の値セットが表示されています。

グループレベルおよびグループインスタンスレベルの値セットエディター



グループレベルでの値の設定

グループレベルは、デバイスグループの値を定義します。これは、下のどれかのレベルによってオーバーライドされない限り、指定されたデバイスグループのすべてのサーバーに適用されます。

グループレベルで値を設定するには、アプリケーション構成をデバイスグループにアタッチする必要があります。グループレベルとグループインスタンスレベルの値セットエディターの例については、「グループレベルおよびグループインスタンスレベルの値セットエディター」(242ページ)の図を参照してください。

グループレベルで値を設定するには、次の手順を実行します。

1. SAクライアントでデバイスグループを開きます。アプリケーション構成がデバイスグループにアタッチされている必要があります。
2. 左側の[表示]ペインで、[構成されるアプリケーション]ノードを開きます。

3. [構成されるアプリケーション] ノードの下で、目的のアプリケーション構成ノードを選択します。右側に値セットエディターが表示されます。
4. [テンプレート] ドロップダウンリストで目的のテンプレートファイルを選択します。
5. テキストボックスに値を入力します。その他の編集機能を利用するには、右クリックメニュー項目を使用します。
6. [変更の保存] ボタンを選択します。

グループインスタンスレベルでの値の設定

グループインスタンスレベルは、デバイスグループにアタッチされているアプリケーション構成のインスタンスの値を定義します。これは、下のどれかのレベルによってオーバーライドされない限り、指定されたデバイスグループのすべてのサーバーに適用されます。

グループインスタンスレベルで値を設定するには、アプリケーション構成をデバイスグループにアタッチする必要があります。グループレベルとグループインスタンスレベルの値セットエディターの例については、次の図を参照してください。

グループインスタンスレベルで値を設定するには、次の手順を実行します。

1. SAクライアントでデバイスグループを開きます。アプリケーション構成がデバイスグループにアタッチされている必要があります。「サーバーまたはデバイスグループへのアプリケーション構成のアタッチ」(286ページ)を参照してください。
2. 左側の [表示] ペインで、[構成されるアプリケーション] ノードを開きます。
3. [構成されるアプリケーション] ノードの下で、目的のアプリケーション構成ノードを開きます。
4. アプリケーション構成の目的のインスタンスを選択します。たとえば、「グループレベルおよびグループインスタンスレベルの値セットエディター」(242ページ)では、“Production Instance WAS-appconfig” という名前のアプリケーション構成インスタンスが選択されています。これは、WAS-app-configという名前のアプリケーション構成オブジェクトのインスタンスです。このアプリケーション構成はデバイスグループにアタッチされており、アプリケーション構成の2つのインスタンスが定義されています。
5. [テンプレート] ドロップダウンリストで目的のテンプレートファイルを選択します。
6. テキストボックスに値を入力します。その他の編集機能を利用するには、右クリックメニュー項目を使用します。詳細については、「値セットエディターでの値の設定」(235ページ)も参照してください。
7. [変更の保存] ボタンを選択します。

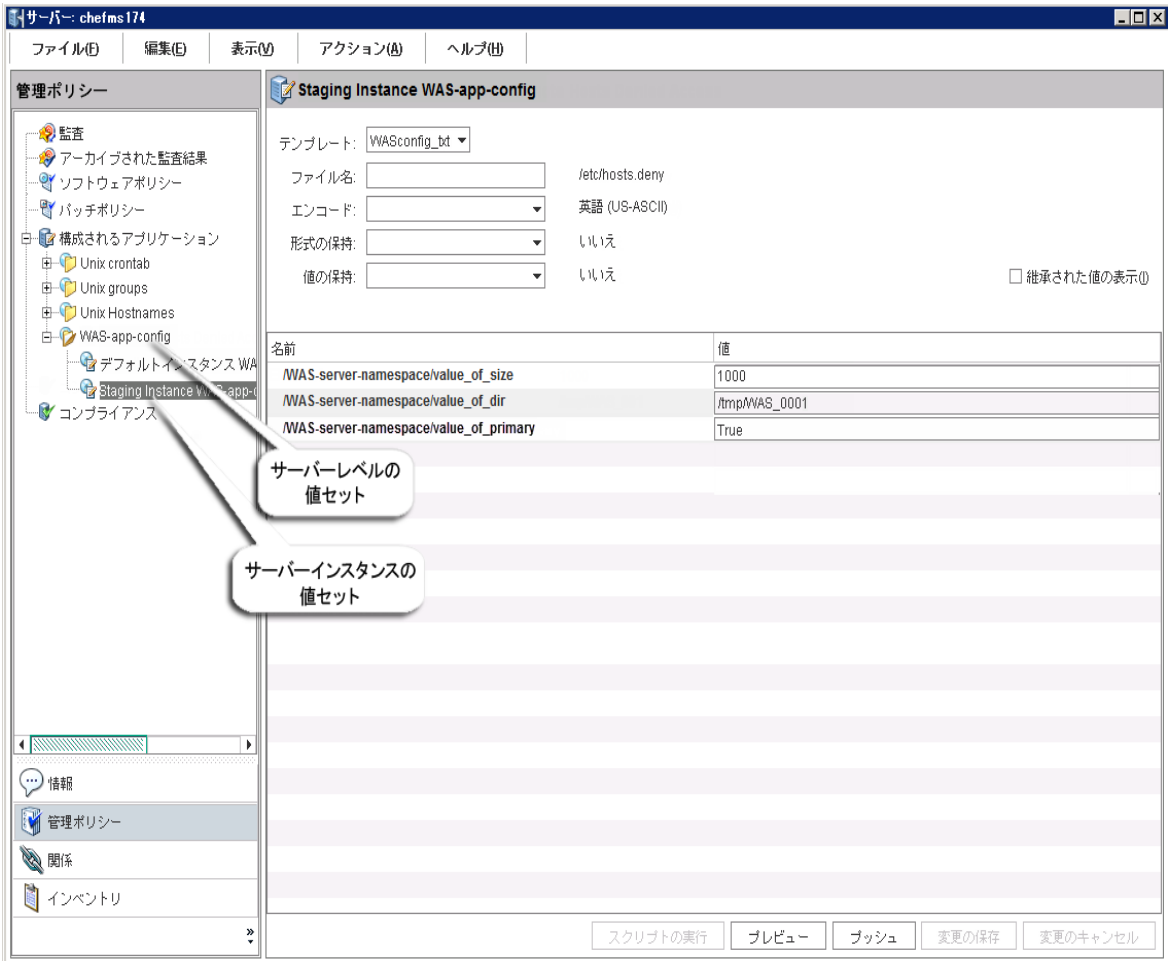
サーバーレベルの値セットエディター

サーバーレベルは、サーバーの値を定義します。これは、下のレベルによってオーバーライドされない限り、指定されたサーバーのアプリケーション構成のすべてのインスタンスに適用されます。

「サーバーレベルおよびサーバーインスタンスレベルの値セットエディター」(245ページ)は、サーバーおよびサーバーインスタンスレベルの値セットエディターを示します。このビューが使用できるのは、アプリケーション構成がアタッチされているサーバーを開いたときだけです。

- “WAS-app-config” という名前のアプリケーション構成を選択すると、サーバーレベルで値セットエディターが表示されます。
- “Production Instance WAS-appconfig” または “Staging Instance WAS-appconfig” という名前のインスタンスを選択すると、サーバーインスタンスレベルで値セットエディターが表示されます。

サーバーレベルおよびサーバーインスタンスレベルの値セットエディター



サーバーレベルでの値の設定

サーバーレベルは、サーバーの値を定義します。これは、下のレベルによってオーバーライドされない限り、指定されたサーバーのアプリケーション構成のすべてのインスタンスに適用されます。

サーバーレベルで値を設定するには、アプリケーション構成をサーバーにアタッチする必要があります。サーバーレベルとサーバーインスタンスレベルの値セットエディターの例については、「サーバーレベルおよびサーバーインスタンスレベルの値セットエディター」(245ページ)を参照してください。

サーバーレベルで値を設定するには、次の手順を実行します。

1. SAクライアントでサーバーを開きます。アプリケーション構成がサーバーにアタッチされている必要があります。詳細については、「サーバーまたはデバイスグループへのアプリケーション構成のアタッチ」(286ページ)を参照してください。
2. 左側の[管理ポリシー]タブを選択します。
3. 左側の[管理ポリシー]ペインで、[構成されるアプリケーション]ノードを開きます。
4. [構成されるアプリケーション]ノードの下で、目的のアプリケーション構成ノードを選択します。
5. [テンプレート]ドロップダウンリストで目的のテンプレートファイルを選択します。
6. テキストボックスに値を入力します。その他の編集機能を利用するには、右クリックメニュー項目を使用します。
7. [変更の保存]を選択します。

サーバーインスタンスレベルでの値の設定

サーバーインスタンスレベルは、サーバー上のアプリケーション構成の特定の1つのインスタンスの値を定義します。これは、指定されたサーバーの指定されたアプリケーション構成インスタンスだけに適用され、上のすべてのレベルをオーバーライドします。

サーバーインスタンスレベルで値を設定するには、アプリケーション構成をサーバーにアタッチする必要があります。サーバーレベルとサーバーインスタンスレベルの値セットエディターの例については、「サーバーレベルおよびサーバーインスタンスレベルの値セットエディター」(245ページ)を参照してください。

サーバーインスタンスレベルで値を設定するには、次の手順を実行します。

1. SAクライアントでサーバーを開きます。アプリケーション構成がサーバーにアタッチされている必要があります。詳細については、「サーバーまたはデバイスグループへのアプリケーション構成のアタッチ」(286

ページ)を参照してください。

2. 左側の[**管理ポリシー**]タブを選択します。
3. 左側の[管理ポリシー]ペインで、[構成されるアプリケーション]ノードを開きます。
4. [構成されるアプリケーション]ノードの下で、目的のアプリケーション構成ノードを開きます。
5. 目的のアプリケーション構成ノードの下で、目的のインスタンスを選択します。
6. [**テンプレート**]ドロップダウンリストで目的のテンプレートファイルを選択します。
7. テキストボックスに値を入力します。その他の編集機能を利用するには、右クリックメニュー項目を使用します。
8. [**変更の保存**]を選択します。

アプリケーション構成でのスクリプトの実行

構成値がターゲットサーバーにコピーされる前または後に実行されるスクリプトをアプリケーション構成に追加できます。

たとえば、アプリケーションを停止する**インストール前スクリプト**と、構成変更後にアプリケーションを再起動する**インストール後スクリプト**を追加できます。プッシュまたはインストール後スクリプトでエラーが発生した場合、**エラー後スクリプト**を実行できます。

あるいは、テキスト以外の構成データを処理するための**データ操作スクリプト**が必要な場合もあります。IISサーバーを構成する場合、データ操作スクリプトを使用して、メタベース情報をフラットファイルに読み取ることができます。フラットファイル内の情報が構成テンプレートによって解析されたら、インストール後スクリプトを実行して、更新された情報をメタベース情報に書き戻すことができます。

> **注意:** JScriptまたはVBScriptのインストール前、インストール後、またはエラー後スクリプトを含むアプリケーション構成をプッシュする場合、スクリプトが失敗してもプッシュが成功する可能性があります。このような場合、プッシュはスクリプトのエラーを無視します。アプリケーション構成はスクリプトの失敗を検出せず、プッシュはエラーなしで完了します。これらのタイプのスクリプトを使用する場合は、スクリプトにエラーがないことと、スクリプトがWScript.Quit(<ステータス>)を呼び出して0以外の終了ステータスを返すことを確認する必要があります。

アプリケーション構成スクリプトのタイプ

次の表に、アプリケーション構成オブジェクトで使用可能なスクリプトのタイプを示します。スクリプトタイプは、スクリプトが呼び出されるタイミングを指定します。各タイプのスクリプトは1つまでしか定義できません。

スクリプトを定義して、これらのタイプのうち1つを指定しない場合、スクリプトは構成テンプレートとして扱われます。すなわち、サーバーにプッシュされますが、実行はされません。

スクリプトのタイプと実行のタイミング

スクリプトタイプ	説明
データ操作	インストール前スクリプトより前に実行され、テキスト以外の構成ファイルを解析して、CMLテンプレートで解析可能にする役割を果たします。データ操作スクリプトは、アプリケーション構成によって管理される既存のファイルを単にスキャンしてインポートする場合にも使用できます。 このスクリプトが失敗した場合、アプリケーション構成はサーバーにプッシュされません。
インストール前	実際のプッシュの前に実行されます。たとえば、インストール前スクリプトはアプリケーションやサービスを停止することができます。 このスクリプトが失敗した場合、アプリケーション構成はサーバーにプッシュされません。
インストール後	実際のプッシュの後に実行されます。たとえば、インストール後スクリプトはプッシュ後にサービスを再起動できます。
エラー後	プッシュが失敗するか、インストール後スクリプトが失敗した場合のみ実行されます。たとえば、エラー後スクリプトはバックアップファイルを復元できます。

スクリプトタイプを指定するには、次の手順を実行します。

1. SAクライアントで、テンプレートを含むアプリケーション構成オブジェクトを開きます。
2. [構成される値]ビューを選択して、アプリケーション構成オブジェクトに含まれるテンプレートを表示します。
3. テンプレートを選択し、右クリックしてメニューを表示します。
4. 上の表に示されているスクリプトタイプを選択します。
5. [**ファイル**] > [**保存**]メニューを選択します。

詳細については、「スクリプトからのテンプレートの作成」(283ページ)も参照してください。

上の表に、アプリケーション構成オブジェクトで使用可能なスクリプトのタイプを示します。このタイプは、スクリプトの構文と実行環境を指定します。

スクリプトソースのタイプ

スクリプトソースタイプ	説明
Windows .BATスクリプト	Windowsバッチコマンドファイル。
Windows .JSスクリプト	Windowsで動作するJavascriptファイル。

スクリプトソースのタイプ (続き)

スクリプトソースタイプ	説明
Windows .CMDスクリプト	Windowsバッチコマンドファイル。
Windows .VBSスクリプト	Windows Visual Basicスクリプト。
Windows .WSFスクリプト	Windowsスクリプトファイル。
Windows .PYスクリプト	Windowsで動作するPythonファイル。
Unix .SHスクリプト	Unixシェルスクリプト。
その他のUnixスクリプト	Unixで動作するその他のスクリプト。

スクリプトタイプを指定するには、次の手順を実行します。

1. SAクライアントで、テンプレートを開きます。
2. [プロパティ] ビューを選択します。
3. [タイプ] フィールドで、上の表に示されているスクリプトタイプを選択します。
4. [**ファイル**] > [**保存**] メニューを選択します。

詳細については、「スクリプトからのテンプレートの作成」(283ページ)も参照してください。

アプリケーション構成のサーバーへのプッシュ

値セットの値を変更した場合、その変更をターゲットサーバー上の構成ファイルにマージするには、アプリケーション構成をサーバーにプッシュする必要があります。アプリケーション構成をプッシュすると、値セットのすべての値が、ターゲット管理対象サーバー上の構成ファイルの値を置き換えます。ターゲットサーバー上に構成ファイルが存在しない場合は、プッシュしたときにサーバー上に新規ファイルが作成されます。さらに、アプリケーション構成内のすべてのスクリプトが、スクリプトタイプに基づいて実行されます。

アプリケーション構成をプッシュすると、次の動作が発生します。

- SAはデータ操作スクリプト (指定されている場合) を実行します。
- SAはテンプレートと値セットからターゲット構成ファイルを生成します。
- SAは既存の構成ファイルをバックアップします。
- SAはインストール前スクリプト (指定されている場合) を実行します。
- SAは生成した構成ファイルをサーバーにコピーします。
- SAはインストール後スクリプト (指定されている場合) を実行します。

- インストール前スクリプトが失敗するか、インストール後スクリプトが失敗するか、コピー操作が失敗した場合、SAはエラー後スクリプト (指定されている場合) を実行します。

アプリケーション構成のプッシュとソフトウェアポリシーおよび監査の関連については、「ソフトウェアポリシーでのアプリケーション構成」(258ページ)を参照してください。

アプリケーション構成でのスクリプトの使用に関する詳細については、「スクリプトからのテンプレートの作成」(283ページ)と「データ操作スクリプトの実行による非テキスト構成の管理」(284ページ)を参照してください。

アプリケーション構成のプッシュ方法については、「アプリケーション構成のプッシュ」(290ページ)を参照してください。

> **注:** プッシュの際にシーケンス (リストとスカラーの) がマージされる方法は、アプリケーション構成の継承階層での値の設定方法と、アプリケーション構成のCMLテンプレートで構成されているシーケンスマージモードによって異なります。シーケンスのマージの詳細については、「シーケンスの集約」(417ページ)を参照してください。

アプリケーション構成コンプライアンス

アプリケーション構成コンプライアンスを使用すると、サーバー (またはサーバーのグループ) にアタッチされたアプリケーション構成の値が、ターゲットサーバー上の構成ファイルの値と一致するかどうかを判定できます。

ターゲット構成ファイルの値が、アプリケーション構成に定義された値と一致する場合、サーバーはコンプライアンス状態にあると見なされます。ターゲット構成がアプリケーション構成に定義された値と一致しない場合、サーバーは非コンプライアンス状態にあると見なされます。

SAクライアントは、アプリケーション構成に対して次のコンプライアンスステータスを表示します。

- **コンプライアンス**: サーバーまたはデバイスグループ (または複数のサーバーおよびグループ) にアタッチされたアプリケーション構成のすべての値が、ターゲットサーバー上の構成値と一致します。 アイコンで表されます。

  デバイスグループの場合、アプリケーション構成コンプライアンスは、グループに属するすべてのサーバー (およびすべてのサブグループのサーバー) のコンプライアンスステータスに基づきます。デフォルトでは、グループのコンプライアンスはデフォルトのしきい値で決定されます。グループ内のすべてのサーバーの5%より多くが非コンプライアンスステータスにある場合、グループ全体が非コンプライアンスステータスと見なされます。このデフォルト設定を変更するには、HPE SSOポータルで『Server Automationユーザーガイ

ド』の「デバイスグループのコンプライアンス設定の変更」を参照してください。

- **非コンプライアンス**: アプリケーション構成で定義された値のうち少なくとも1つが、ターゲットサーバー上の構成ファイルの値に一致しません。 アイコンで表されます。

  デバイスグループの場合、非コンプライアンスは、グループに属するすべてのサーバー(およびすべてのサブグループのサーバー)のコンプライアンスステータスに基づきます。デフォルトでは、グループの非コンプライアンスはデフォルトのしきい値で決定されます。グループ内のすべてのサーバーの5%より多くが非コンプライアンスステータスにある場合、グループ全体が非コンプライアンスステータスと見なされます。このデフォルト設定を変更するには、HPE SSOポータルで『Server Automationユーザーガイド』の「デバイスグループのコンプライアンス設定の変更」を参照してください。

- **スキャン開始済み**: アプリケーション構成コンプライアンス情報は現在計算中です。アイコンで表されます。
- **スキャンが必要**: アプリケーション構成コンプライアンス情報は未定義です。コンプライアンススキャンが実行されていないか(新規インストールの場合など)、最後にSAクライアントに情報が報告された後にサーバー(またはデバイスグループ内のサーバー)の構成が変更されている可能性があります。アイコンで表されます。
- **該当しない**: アプリケーション構成コンプライアンス情報は該当せず、ダッシュ(—)で表されます。これは、サーバーにアタッチされているアプリケーション構成が存在しない場合に表示されます。

アプリケーション構成コンプライアンスは、個々のサーバーまたはサーバーのグループに対して表示できます。

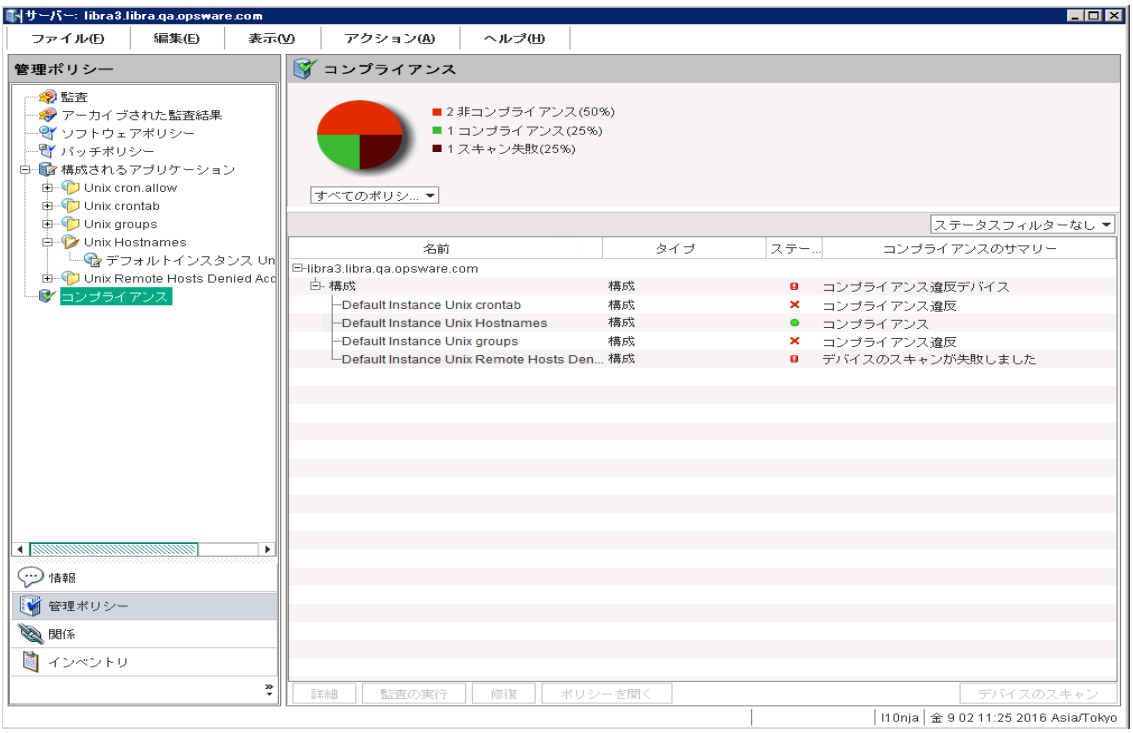
- 「1つのサーバーのアプリケーション構成コンプライアンス」(251ページ)
- 「複数のサーバーのアプリケーション構成コンプライアンス」(252ページ)

> **注:** 値セットエディターでの値の編集など、アプリケーション構成に対する何らかの変更を行った場合、その構成がアタッチされているサーバーまたはサーバーグループのコンプライアンスステータスは[スキャンが必要]になります。

1つのサーバーのアプリケーション構成コンプライアンス

1つのサーバーの場合、コンプライアンスビューには、そのサーバーにアタッチされているすべてのアプリケーション構成の総合的なコンプライアンスが表示されます。複数のアプリケーション構成がサーバーにアタッチされている場合、すべてのアプリケーション構成の総合コンプライアンスステータスの他に、各構成の個別のコンプライアンスステータスも表示できます。

次の図に、1つのサーバーのアプリケーション構成コンプライアンスを示します。



アプリケーション構成と、ターゲットサーバー上の実際の構成ファイルとの間に、何らかの違いが見つかった場合、下のペインに非コンプライアンス状態のカテゴリが表示されます。複数のアプリケーション構成がサーバーにアタッチされており、アプリケーション構成のターゲットとなる構成ファイルのどれか1つでもアプリケーション構成と異なる場合は、サーバーのステータスは非コンプライアンスになります。

アプリケーション構成コンプライアンススキャンの実行方法については、「サーバーのアプリケーション構成コンプライアンスのスキャン」(256ページ)を参照してください。

複数のサーバーのアプリケーション構成コンプライアンス

複数のサーバーに対してアプリケーション構成コンプライアンスステータスを表示できます。SAクライアントのナビゲーションペインで、[デバイス] を選択し、[デバイスグループ] または [サーバー] を選択します。デバイスグループまたはサーバーのセットを選択し、[表示] メニューから [コンプライアンス] を選択します。選択したサーバーの総合コンプライアンスステータスが表示されます。

サーバーのグループにアタッチされたアプリケーション構成がコンプライアンス状態と見なされるのは、グループ内のサーバーのうち非コンプライアンス状態のものの割合が5%未満の場合です。非コンプライアンス状態のものが5%を超える場合、総合コンプライアンスは非コンプライアンスと見なされます。この割合を変更するには、SAクライアントで [管理] タブを選択し、[コンプライアンス設定] を選択します。

コンプライアンスビューのサーバーグループの詳細ペインには、すべてのアプリケーション構成がコンプライアンス状態であるかどうかが示されますが、展開して個々のサーバーおよびアプリケーション構成の内訳を表示することはできません。

サーバーグループのアプリケーション構成コンプライアンスステータスを表示するには、次の方法を使用します。

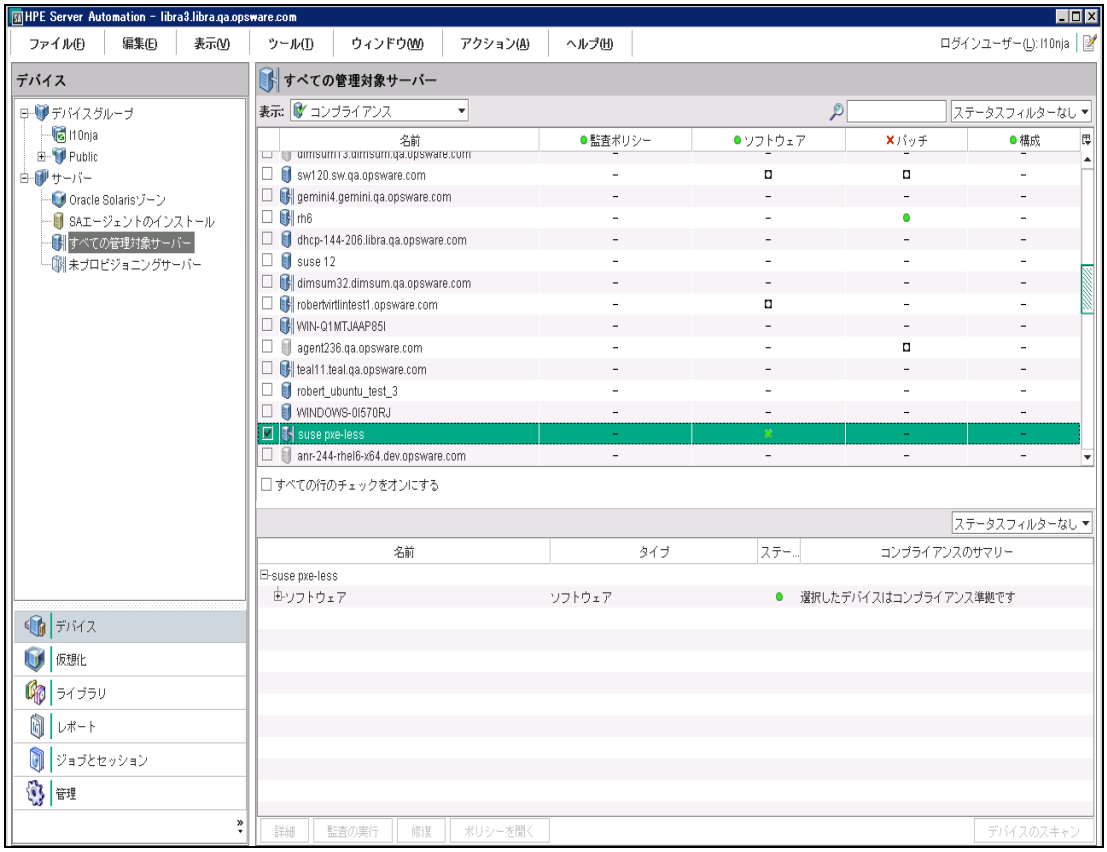
- 「複数のサーバーのアプリケーション構成コンプライアンスの表示」(253ページ)
- 「複数のデバイスグループのアプリケーション構成コンプライアンスの表示」(254ページ)
- 「1つのデバイスグループのアプリケーション構成コンプライアンスの表示」(255ページ)

複数のサーバーのアプリケーション構成コンプライアンスの表示

複数のサーバーのアプリケーション構成コンプライアンスを表示するには、次の手順を実行します。

1. SAクライアントのナビゲーションペインで [**デバイス**] > [**サーバー**] > [**すべての管理対象サーバー**] を選択します。
2. [表示] ドロップダウンリストから、[**コンプライアンス**] を選択します。
3. 複数のサーバーのコンプライアンスレベルを表示するには、サーバーの隣のチェックボックスを選択すると、選択したサーバーのコンプライアンスの集計が、次の図で示すように下の詳細ペインに表示され

ます。

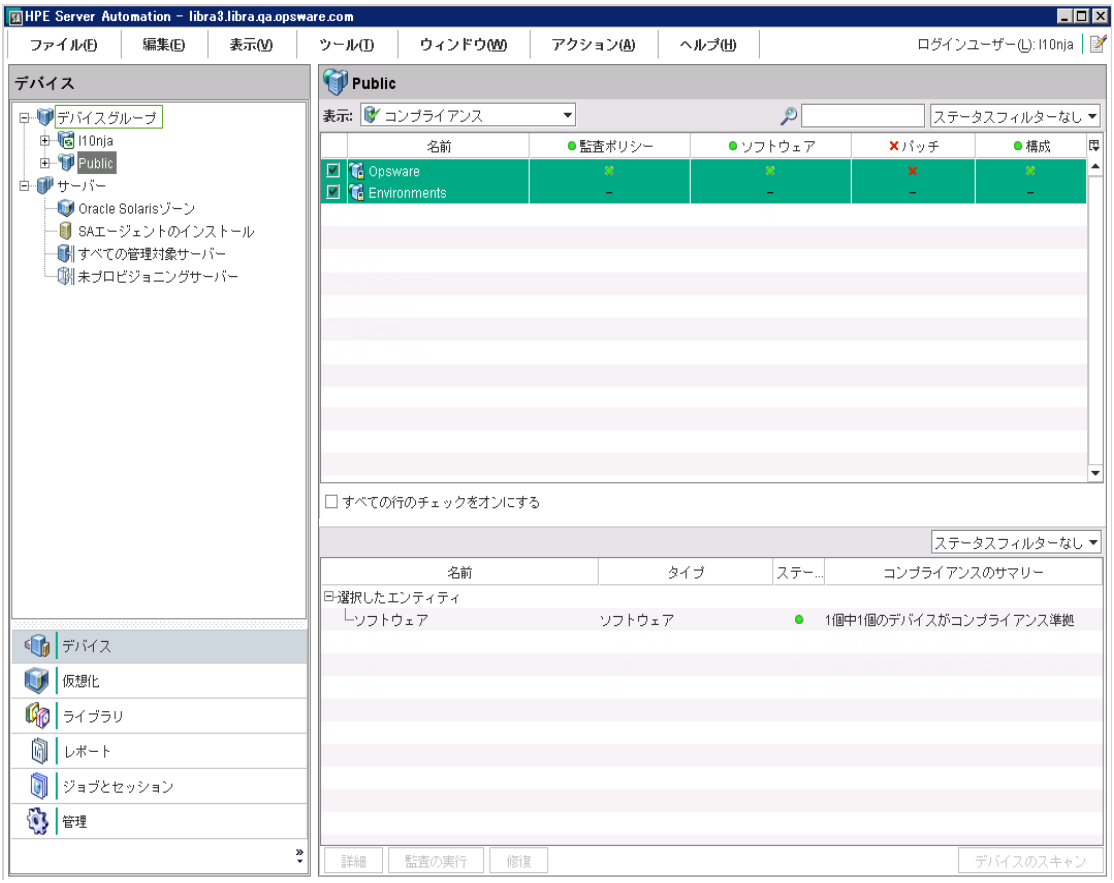


複数のデバイスグループのアプリケーション構成コンプライアンスの表示

複数のデバイスグループのアプリケーション構成コンプライアンスを表示するには、次の手順を実行します。

1. SAクライアントのナビゲーションペインで [**デバイス**] > [**デバイスグループ**] を選択します。
2. デバイスグループまたは、デバイスグループを含むフォルダーを選択します。
3. [表示] ドロップダウンリストから、[コンプライアンス] を選択します。すべてのグループのコンプライアンスステータスが表示されます。
4. 複数のグループのコンプライアンスレベルを表示するには、サーバーの隣のチェックボックスを選択すると、選択したグループのコンプライアンスのサマリーが、次の図で示すように下の詳細ペインに表示さ

れます。

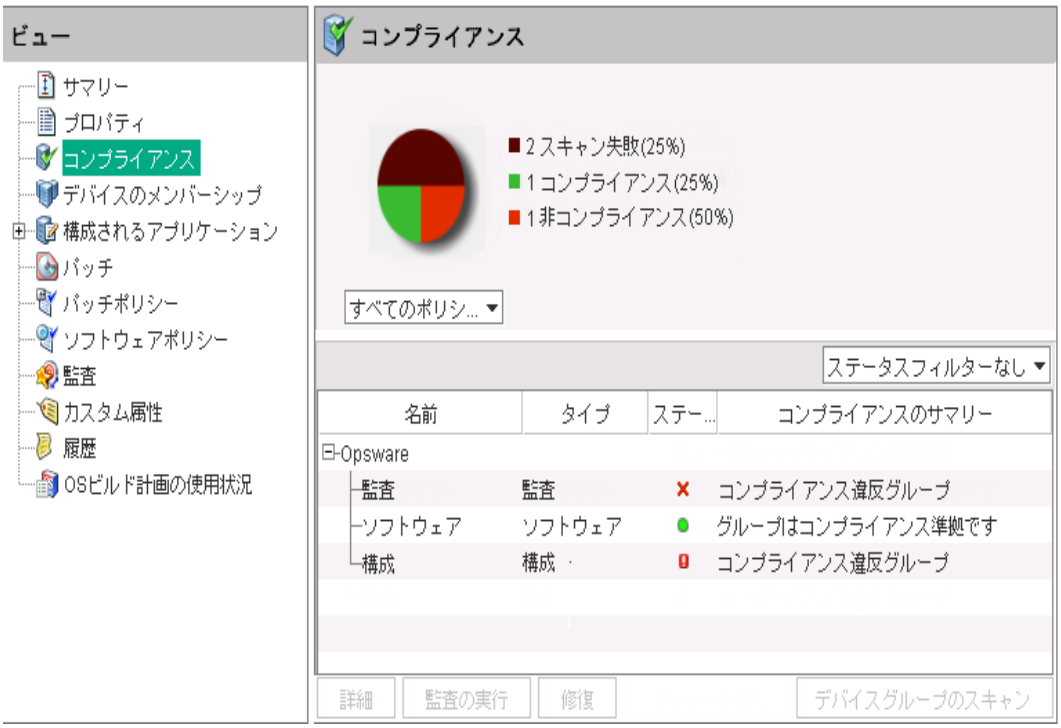


1つのデバイスグループのアプリケーション構成コンプライアンスの表示

1つのデバイスグループのアプリケーション構成コンプライアンスを表示するには、次の手順を実行します。

1. SAクライアントのナビゲーションペインで [**デバイス**] > [**デバイスグループ**] を選択します。
2. 目的のデバイスグループに移動して選択します。
3. 右クリックして [**開く**] を選択するか、[**アクション**] > [**開く**] を選択します。デバイスグループが表示されます。
4. [ビュー] ペインで、[コンプライアンス] を選択します。次の図に示すように、各ポリシータイプに関して、すべてのメンバーを含むグループ全体の総合コンプライアンスが表示されます。個々のサーバーのコンプライアンスステータスは表示されません。

1つのデバイスグループのアプリケーション構成コンプライアンス



サーバーのアプリケーション構成コンプライアンスのスキャン

アプリケーション構成をサーバーにプッシュした後で、サーバー上の構成ファイルが、意図的に、あるいは誤って変更される可能性があります。あるいは、アプリケーション構成に定義されている値が変更される可能性もあります。ターゲットサーバー上の構成ファイルの値がアプリケーション構成に定義されている値と一致しない場合、構成ファイルは非コンプライアンス状態と見なされます。

サーバーの構成コンプライアンスをスキャンすることにより、サーバー上の構成ファイルの中に、構成テンプレートに記録されている値に対して非コンプライアンス状態のものがないかどうかを確認できます。スキャンは定期的に実行するようにスケジュールできます。

1つまたは複数のサーバーの構成コンプライアンスをスキャンするには、次の手順を実行します。

1. SAクライアントのナビゲーションペインで、[デバイス]を選択します。
2. [デバイスグループ]または[すべての管理対象サーバー]を選択します。[デバイスグループ]を選択した場合、デバイスグループを選択して、それに属するサーバーを表示します。

3. 内容ペインでサーバーを選択します。複数のサーバーまたはデバイスグループを選択して、それらをすべてスキャンすることもできます。
4. [**アクション**] メニューで、[**スキャン**] > [**構成コンプライアンス**] を選択するか、[**スケジュール**] > [**構成コンプライアンススキャン**] を選択します。
   - [**スキャン**] > [**構成コンプライアンス**] を選択した場合、SAはデバイスをスキャンしてコンプライアンスを判定し、[構成コンプライアンスのスキャン] 画面にステータスを表示します。
   - [**スケジュール**] > [**構成コンプライアンススキャン**] を選択した場合、[ジョブのスケジュール] 画面が表示され、ジョブを実行する時刻とその他のジョブパラメーターを指定できます。

アプリケーション構成の監査

SAでは、サーバー上の構成ファイルを監査して、ファイルが組織の構成標準を満たすかどうかを判定できます。監査ルールを作成することにより、サーバー上の構成ファイルの正しい定義方法を指定し、定期的にサーバーを監査して、構成ファイルが正しく構成されているかどうかを確認できます。監査ルールの定義とターゲット構成ファイルの値に不一致が見つかった場合、サーバーを修復して問題を修正できます。

たとえば、管理対象サーバーの/etc/hostsファイルで特定のIPアドレスに対して特定のホスト名だけが定義されていることを確認するには、許容されるホスト名とIPアドレスのペアを指定する監査ルールを定義します。監査を実行したときに、ルールに指定されていない値がホストファイルに含まれていると、監査結果にエラーが表示され、問題を修復できます。

アプリケーション構成の監査の一般的なプロセスは、次の手順に従います。

1. **監査と監査ルールの作成**: サーバー上の構成ファイルを監査するには、まず監査を作成します。監査を作成するには、構成ルールの基になるソースサーバー (あるいはスナップショットまたはスナップショット仕様) を指定します。次に、アプリケーション構成テンプレートを選択してルールを作成します。ルールは、ターゲット構成ファイルでチェックする値を定義します。各監査ルールに対して、ターゲットサーバー上の構成ファイルの場所を指定します。
2. **ターゲットサーバーの選択**: 監査で、監査のターゲットサーバーを選択します。1つのサーバー、複数のサーバー、またはサーバーのグループを選択できます。
3. **監査の実行またはスケジュール**: 監査をスケジュールして、1回だけ実行するか、定期的に実行できます。また、監査結果を送信する電子メールアドレスを指定できます。
4. **監査結果のチェック**: 監査結果をチェックして、ターゲットサーバー上の構成ファイルが監査ルールに定義されている値と一致するかどうかを確認します。不一致がある場合、ルールとターゲットファイルを比較して差異を確認し、サーバーを修復する方法を決定できます。

5. **サーバーの修復**: 監査結果に見つかった差異を修正するには、サーバーまたはルールの一部または全部を修復して、ターゲット構成とルールが一致するようにします。

監査とスナップショットの使用法の詳細については、『SA 10.50管理ガイド』の「監査と修復」の項を参照してください。

ソフトウェアポリシーでのアプリケーション構成

アプリケーション構成は、ソフトウェアポリシー内部で使用すると強力なツールとなります。ソフトウェアポリシーは、アプリケーションの理想的な状態を定義します。これには、すべてのパッケージ、パッチ、スクリプト、およびサーバー上にインストールする必要があるその他のオブジェクトに加えて、アプリケーションの構成ファイルをサーバー上で設定する方法も含まれます。管理対象サーバーにソフトウェアポリシーをインストールすると、SAは、アプリケーション構成に定義されたすべての値を含め、すべての内容をポリシーのターゲットのサーバーに適用します。

SAクライアントのコンプライアンスビューを使用すると、ポリシーからインストールされたソフトウェアのコンプライアンスステータスを表示できます。たとえば、ソフトウェアポリシーからパッチが削除されたり、サーバー上に新しいパッケージがインストールされたり、ポリシーに定義された構成ファイルが変更されたりした場合、ポリシーはコンプライアンスビューに非コンプライアンスとして表示されます。アプリケーションが正しくインストールされ、構成されるようにするには、サーバーを修復します。

ソフトウェアポリシーの使用と作成の詳細については、『SA 10.50ユーザーガイド』の「ソフトウェア管理」を参照してください。

ソフトウェアポリシーでアプリケーション構成を使用するには、次の手順を実行します。

1. **アプリケーションの定義**: ソフトウェアポリシーを作成する前に、アプリケーションエキスパートが、アプリケーションを構成する必要なパッケージとパッチをすべて収集します。また、アプリケーションに関連する構成ファイルの定義と管理のための構成テンプレートを収集します。
2. **パッケージとパッチのSAへのインポート**: ソフトウェアポリシーのコンポーネントが定義されたら、アプリケーションを構成するパッケージとパッチをすべてSAライブラリにインポートし、ソフトウェアポリシーに配置できるようにします。
3. **アプリケーション構成の作成と値の設定**: 構成ファイルの生成に用いられる構成値を定義します。たとえば、Apache Webサーバーをデプロイするためのソフトウェアポリシーを作成する場合、アプリケーションエキスパートは、値セットエディターを使用して、httpd.confファイルのデフォルト値を定義します。必要な場合、インストール前またはインストール後スクリプトをアプリケーション構成に追加します。たとえば、ソフトウェアポリシーの修復中にアプリケーション構成がプッシュされた後でApacheサービスを再起動するスクリプトです。

4. **アプリケーション構成のテスト**: アプリケーション構成をソフトウェアポリシーに追加して、アプリケーションをサーバーにデプロイする前に、アプリケーション構成をサーバーにアタッチして、ソフトウェアポリシーを作成する前にアプリケーションが正しく動作することを確認するとよいでしょう。サーバーへの構成のプッシュをプレビューして、正しいかどうかを確認できます。
5. **ソフトウェアポリシーの作成**: ソフトウェアポリシーのすべてのコンポーネントを定義し、作成して、SAにインポートしたら、アプリケーションエキスパートはソフトウェアポリシーを作成して、インストールするソフトウェアと、そのコンポーネント (すべてのパッチ、パッケージ、アプリケーション構成を含む) のインストール順序を指定します。SAライブラリに保存されたソフトウェアポリシーは、アプリケーションのデプロイ、テスト、管理を行うシステム管理者から使用できるようになります。
6. **サーバーまたはサーバーグループへのポリシーのアタッチ**: ソフトウェアポリシーを作成して保存したら、システム管理者は、ポリシーをデバイスグループ内のサーバーまたはサーバーグループにアタッチします。
7. **サーバーの修復によるソフトウェアのインストール**: システム管理者は、ソフトウェアポリシーをサーバー上で修復して、1つ以上のサーバーにソフトウェアをデプロイします。修復により、ポリシーに定義されたすべてのものが、ポリシーに指定された順序でターゲットサーバー上にデプロイされます。管理者は、修復の前にアプリケーション構成をプレビューできます。プレビューステップでは、サーバーインスタンスレベルでテンプレート変数のappconfigの値を定義することができます。
8. **アプリケーションのテストと変更の反復**: システム管理者がソフトウェアポリシーの修復を通じてアプリケーションをインストールした後で、アプリケーションを運用環境に投入する前に、アプリケーションが正しく動作し、正しいコンポーネントを含むことをテストする必要があります。さらに、構成ファイルに影響されるアプリケーションの各部分をチェックして、正しく構成されていることを確認する必要があります。
9. **アプリケーションのロールアウト**: アプリケーションをデプロイして使用を開始した後で、システム管理者は継続的な管理およびメンテナンス作業を行います。たとえば、ソフトウェアコンプライアンススキャンを実行してアプリケーションがデプロイされているサーバーのコンプライアンスステータスを判定したり、非コンプライアンス状態のサーバーを修復したり、ソフトウェアコンプライアンスレポートを生成したりします。

ソフトウェアポリシーの使用に関する詳細については、『SA 10.50ユーザーガイド』を参照してください。

アプリケーション構成のタスク

この項では、アプリケーション構成に関するタスクを実行する手順について説明します。

アプリケーション構成とテンプレートの概要:

- 「アプリケーション構成の作成」(261ページ)
- 「構成テンプレートの作成」(262ページ)
- 「ビジュアルエディターによる構成テンプレートの作成」(264ページ)

アプリケーション構成テンプレートの編集と管理:

- 「ビジュアルエディターによる構成テンプレートの編集」(271ページ)
- 「テンプレート内のCMLまたはXMLの編集」(279ページ)
- 「テンプレートファイルのインポートと検証」(280ページ)
- 「構成テンプレートソースの表示」(281ページ)
- 「アプリケーション構成に対するテンプレートの追加または削除」(282ページ)
- 「アプリケーション構成でのテンプレート順序の指定」(282ページ)

アプリケーション構成でのスクリプトの使用:

- 「スクリプトからのテンプレートの作成」(283ページ)
- 「データ操作スクリプトの実行による非テキスト構成の管理」(284ページ)
- 「データ操作スクリプトの手動での実行」(285ページ)

アプリケーション構成のアタッチとデタッチ:

- 「サーバーまたはデバイスグループへのアプリケーション構成のアタッチ」(286ページ)
- 「サーバーまたはデバイスグループからのアプリケーション構成のデタッチ」(288ページ)

アプリケーション構成のプッシュと管理:

- 「アプリケーション構成のプッシュ」(290ページ)
- 「アプリケーション構成プッシュのスケジュール設定」(292ページ)
- 「構成ファイルの過去の状態への復元」(293ページ)
- 「ジョブ結果の検索とフィルター処理」(295ページ)
- 「構成テンプレートとターゲット構成ファイルの比較 - プレビュー」(296ページ)
- 「構成テンプレートの比較」(298ページ)
- 「構成ファイルの要素名のローカライズ」(298ページ)

アプリケーション構成の作成

アプリケーション構成は、構成テンプレートファイルのコンテナーであり、アプリケーション構成がサーバーにプッシュされる際に実行されるスクリプトを含む場合もあります。詳細については、「アプリケーション構成オブジェクト」(229ページ)を参照してください。

アプリケーション構成を作成する方法

1. SAクライアントナビゲーションペインで、[**ライブラリ**] を選択し、[**タイプ別**] タブを選択します。
2. [アプリケーション構成] ノードを見つけて開きます。
   - [構成] ノードを開きます。
   - アプリケーション構成に関連するオペレーティングシステムグループを開きます。
   - オペレーティングシステムを選択します。

   > **注:**
   > アプリケーション構成は複数のオペレーティングシステムに適用できます。これは後の手順で指定します。
3. メニューの [**アクション**] > [**新規**] を選択します。構成の新規作成ウィンドウが開いたら、構成のプロパティと内容を指定します。
4. プロパティビューで、構成の名前と説明を指定します。さらに、次の情報を指定します。
   - **場所: SAライブラリのどこにアプリケーション構成を保存するかを指定します。**
   - **バージョン**: バージョンは任意の文字列で、アプリケーション構成への変更を追跡するために使用します。バージョンは自動的には増加しません。
   - **OS**: このアプリケーション構成が適用されるオペレーティングシステムを指定します。
     - 指定したオペレーティングシステムを搭載したサーバーだけがこのアプリケーション構成を使用できます。
     - アプリケーション構成に含めることができるのは、ここで定義したすべてのオペレーティングシステムに適用可能なテンプレートに限定されます。つまり、構成対象のオペレーティングシステムは、当該テンプレートのサブセットである必要があります。
   - **可用性**: この設定は、テスト済みで使用可能なアプリケーション構成と、まだテストされていないか、非推奨になったアプリケーション構成を区別するために使用します。アプリケーション構成に対して実行可能な操作は、この設定によって変わりません。

- **ローカリゼーションファイル**: [+] ボタンを選択して、ローカル言語の構成ファイルを生成するためのテンプレートを追加します。詳細については、「構成ファイルの要素名のローカライズ」(298ページ)を参照してください。

5. [構成される値] ビューで、**[アクション] > [追加]** を選択するか [+] ボタンをクリックし、テンプレートをアプリケーション構成に追加します。
   - [-] ボタンを使用すると、選択した構成テンプレートを削除できます。
   - 上下の矢印を使用すると、構成テンプレートがインストールされる順序を変更できます。
   - テンプレートのプレビュービューを選択すると、選択したテンプレートファイルの内容が表示されます。
   - ファイルの値ビューを選択すると、構成ファイルを生成する際に選択したテンプレートに挿入される値を確認できます。
   - ファイルのプレビュービューを選択すると、選択した構成ファイルが現在の値セットでどのようになるかを確認できます。
6. **[ファイル] > [保存]** を選択すると、アプリケーション構成を保存できます。

構成テンプレートの作成

構成テンプレートは、ネイティブアプリケーションの構成ファイルと似ていますが、変数部分が構成モデリング言語 (CML) によってテンプレート化され、構成ファイルと値セットの間で値を移動するための命令が含まれています。「構成テンプレートとスクリプトテンプレート」(230ページ)を参照してください。

アプリケーション構成にともに実行するスクリプトも、CMLテンプレート形式で作成する必要があります。詳細については、「スクリプトからのテンプレートの作成」(283ページ)を参照してください。

構成テンプレートを作成するには、次の手順を実行します。

1. SAクライアントナビゲーションペインで、[ライブラリ] を選択し、[タイプ別] タブを選択します。
2. [アプリケーション構成] ノードを見つけて開きます。[テンプレート] ノードを開きます。オペレーティングシステムグループを開き、テンプレートファイルが使用されるオペレーティングシステムを選択します。テンプレートは複数のオペレーティングシステムに適用することもできます。これは後の手順で指定します。
3. [**アクション**] > [**新規**] メニューを選択します。[テンプレート] 画面が表示され、テンプレートのプロパティを定義できます。

4. [プロパティ] ビューを選択し、テンプレートファイルの名前および説明と、次の情報を入力します。
   - **場所: SA**ライブラリのどこにテンプレートを保存するかを指定します。
   - **バージョン**: バージョンは任意の文字列で、テンプレートへの変更を追跡するために使用します。バージョンは自動的には増加しません。
   - **タイプ**: これがテンプレートファイル、スクリプト、ローカリゼーションファイルのどれなのかを指定します。
     - テンプレートファイルは、構成ファイルのモデルです。
     - スクリプトは、構成ファイルをサーバーにプッシュする前または後に実行されます。詳細については、「スクリプトからのテンプレートの作成」(283ページ)を参照してください。
     - ローカリゼーションファイルは、別のロケール向けにカスタマイズされた構成ファイル用に使用されます。詳細については、「構成ファイルの要素名のローカライズ」(298ページ)を参照してください。
   - **パーサー構文**: テンプレートで使用する構文のタイプを次の中から選択します。
     - CML構文は、XMLファイル以外のすべてのテキスト構成ファイルと、すべてのスクリプトファイルに使用します。
     - XML構文は、XMLで作成された構成ファイルに使用します。
     - XML DTD構文は、DTDを使用するXMLで作成された構成ファイルに使用します。
     - OSプロビジョニング用のカスタム属性構文
   - **OS**: このアプリケーション構成テンプレートが適用されるオペレーティングシステムを指定します。
     - 指定したオペレーティングシステムを搭載したサーバーだけがこのテンプレートを使用できます。
     - アプリケーション構成で使用できるテンプレートは、ここで列挙されている1つまたは複数のオペレーシングシステムに適用できるものに限定されます。
   - **可用性**: この設定は、テスト済みで使用可能なテンプレートと、まだテストされていないか、非推奨になったテンプレートを区別するために使用します。テンプレートに対して実行可能な操作は、この設定によって変わりません。このフィールドの値は検索基準として使用できます。
   - **監査可能**: これを設定すると、このテンプレートファイルの監査が可能になります。詳細については、「アプリケーション構成の監査」(257ページ)を参照してください。
5. [内容] ビューを選択します。
6. CMLまたはXMLまたはXML DTDテキストをテンプレートエディターに直接入力します。編集操作と構文の強調表示については、「テンプレート内のCMLまたはXMLの編集」(279ページ)を参照してください。CMLとXMLの詳細については、「CMLリファレンス」(375ページ)および「XML構成ファイルの管理」(301ページ)を参照してください。
7. [**検証**] を選択して、CMLまたはXMLの構文を解析し、エラーをチェックします。
8. [**ファイル**] > [**保存**] を選択してテンプレートを保存します。

9. テンプレートをアプリケーション構成オブジェクトに追加します。「アプリケーション構成に対するテンプレートの追加または削除」(282ページ)を参照してください。

ビジュアルエディターによる構成テンプレートの作成

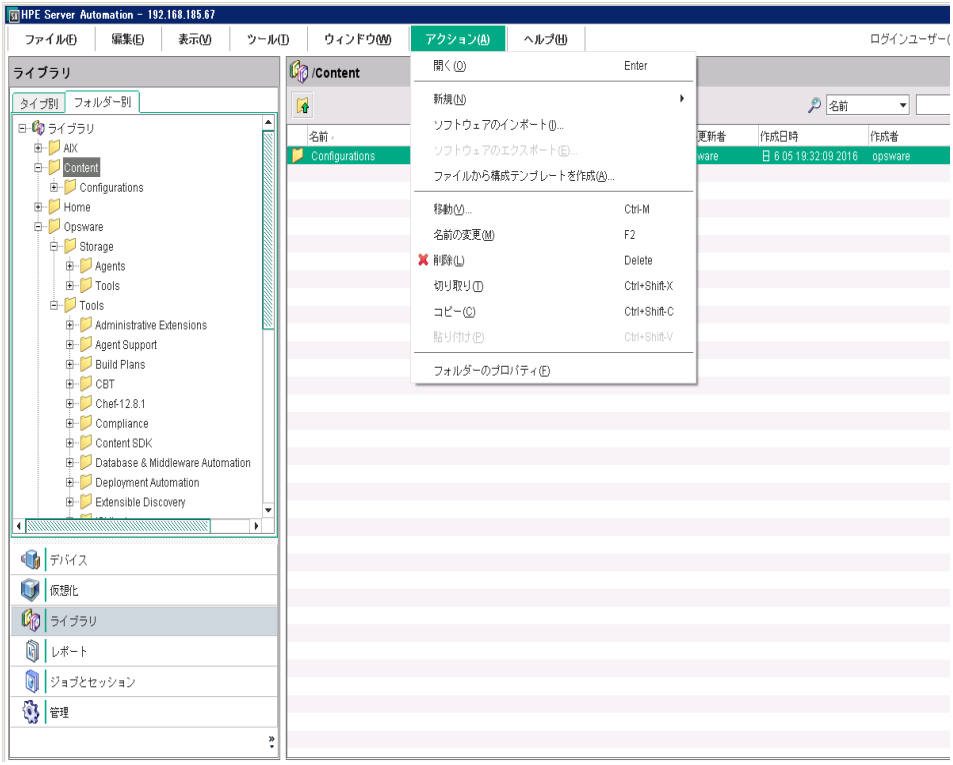
ビジュアルエディターモードでは、CMLを学んだり、構成ファイル全体を理解したりしなくても、簡単なアプリケーションテンプレートを作成できます。これは、構成ファイルのごく一部をパラメーター化して、変更をサーバーにプッシュしたい場合に便利です。

1. 初めに、既存のファイルに基づいて、ビジュアルエディターを使用してアプリケーション構成テンプレートを作成できます。次のようなファイルが使用できます。
   - SA管理対象サーバー上の使用中の構成ファイル
     「使用中の構成ファイルからのテンプレートの作成」(265ページ)を参照してください。
   - ローカルPCからインポートしたサンプル構成ファイル
     「テンプレートのインポートと作成」(265ページ)を参照してください。
2. ビジュアルエディターモードには、CMLを使用せずに簡単なアプリケーション構成テンプレートを編集するためのユーザーインタフェースが用意されています。
   - 構成のプッシュ時に置換する構成オプションにマークを付けると、フォームビューが表示されます。マークの付いたオプションは、元になるテンプレートに基づいて、パラメーター名、表示名、データ型を入力します。フォームビューでは、必要に応じてパラメーターの詳細を変更できます。「ビジュアルエディターによる構成テンプレートの編集」(271ページ)を参照してください。
3. テンプレートを作成した後で、ただちにアプリケーション構成インスタンスを生成することもできます。
   - 管理対象サーバー上の使用中の構成ファイルから、互換性のあるテンプレートのリストを表示したり、テンプレートを開いたり、アプリケーション構成インスタンスをただちに生成してサーバーにアタッチしたりできます。構成インスタンスが値セットエディターで開き、値を変更して変更をサーバーにプッシュできます。詳細については「構成ファイルの管理」(275ページ)を参照してください

> **注:** SAで特定のアクションを実行できるかどうかは、アクセス権の設定で決まります。さらに、サーバーのファイルシステムを通じてビジュアルエディターにアクセスするには、ファイルシステムを読み取るためのOGFSアクセス権が必要です。追加のアクセス権の取得については、SA管理者にお問い合わせください。詳細については、『SA 10.5管理ガイド』を参照してください。

テンプレートのインポートと作成

1. SAクライアントナビゲーションペインで、[**ライブラリ**] > [**フォルダー別**] または [**タイプ別**] を選択して目的のフォルダーに移動して、PCから構成ファイルをインポートするフォルダーにアクセスします。
2. [**アクション**] メニューの [**ファイルから構成テンプレートを作成**] を選択します。



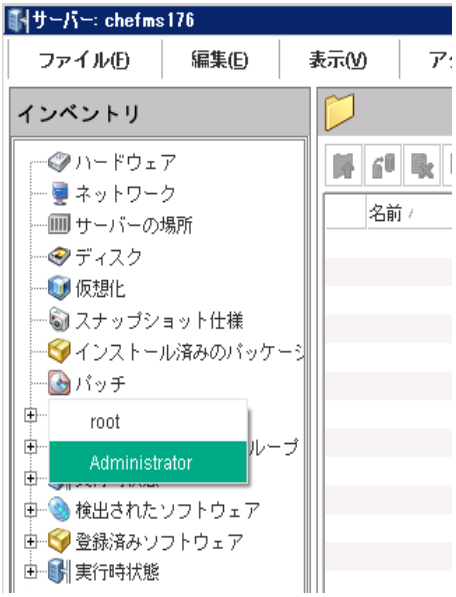
3. インポートする構成ファイルを見つけて選択し、[**開く**] をクリックします。
4. 選択したファイルのテンプレートがビジュアルエディターモードで開きます。

「ビジュアルエディターのインタフェース」(267ページ)および「ビジュアルエディターによる構成テンプレートの編集」(271ページ)を参照してください。

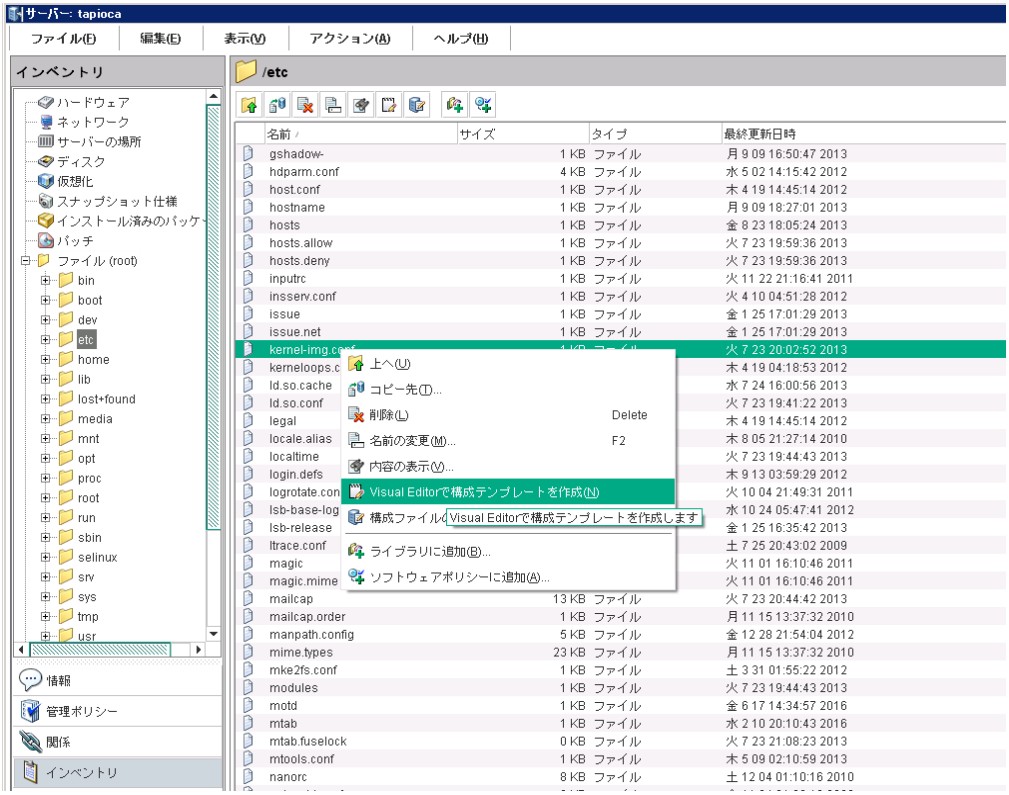
使用中の構成ファイルからのテンプレートの作成

使用中の構成ファイルからビジュアルエディターで構成テンプレートを作成するには、次の手順を実行します。

1. サーバーブラウザーを開きます。
   a. SAクライアントのナビゲーションペインで、管理対象サーバーまたはデバイスグループのリストにアクセスします。
      i. [**デバイス**] > [**サーバー**] > [**すべての管理対象サーバー**] を選択して、サーバーリストを表示します。
      ii. [**デバイス**] > [**デバイスグループ**] を選択して、デバイスグループリストを表示します。
   b. 内容ペインで、開くサーバーまたはデバイスグループを選択します。
   c. [**アクション**] メニューから [**開く**] を選択します。
2. [**サーバーブラウザー**] > [**インベントリ**] > [**ファイル**] に移動します。
3. 指示に従って、サーバーファイルシステムのルートパスを選択します (通常、Unixではroot、WindowsではAdministrator)。



4. 構成ファイルを右クリックして、[**Visual Editorで構成テンプレートを作成**]を選択します。

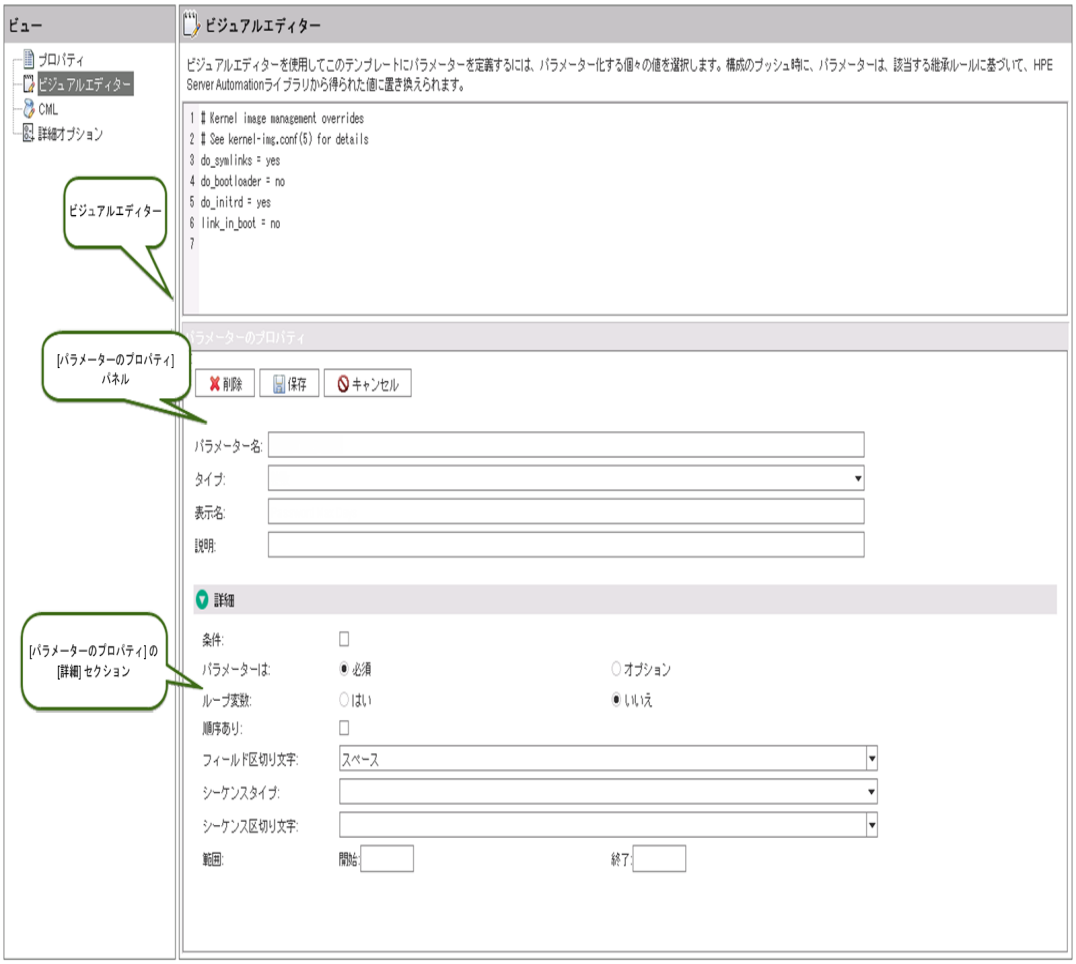


選択した構成ファイルのテンプレートがビジュアルエディターモードで開きます。「ビジュアルエディターのインタフェース」(267ページ)を参照してください。

ビジュアルエディターのインタフェース

ビジュアルエディターインタフェースでは、元になったアプリケーション構成テンプレートの詳細が、2つのパネルに分かれて表示されます。下のパネルには、選択したパラメーターの詳細なプロパティが、編集可能なフォームビューで表示されます。[パラメーターのプロパティ]パネルは、ビジュアルエディターウィンドウに初めて入ったときに表示されます。

ビジュアルエディターのインタフェース

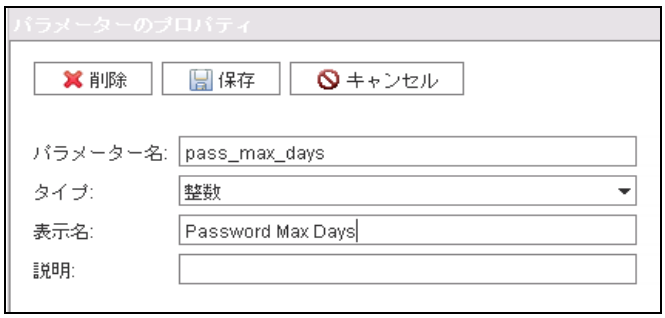


[パラメーターのプロパティ] パネル

ビジュアルエディターで値を選択してから、[パラメーターのプロパティ] パネルの [**パラメーター名**] フィールドの中をクリックすると、[パラメーターのプロパティ] パネルのフィールドが、元になる構成ファイルの値に基づいて設定されます。

パラメーターのプロパティフィールドは、使いやすいフォームビューで編集できます。

[パラメーターのプロパティ] フォーム



パラメーターのプロパティの [詳細] セクション: [詳細] セクションでは、パラメーターの追加オプションを指定できます。追加オプションには、必須 (デフォルト) かオプションか、ループ変数、順序付きリスト、シーケンス (タイプと区切り文字を指定)、または範囲かどうかなどがあります。範囲は、整数、小数、ポートなどの数値タイプに対して指定します(数値タイプはシーケンスに含めることができます)。

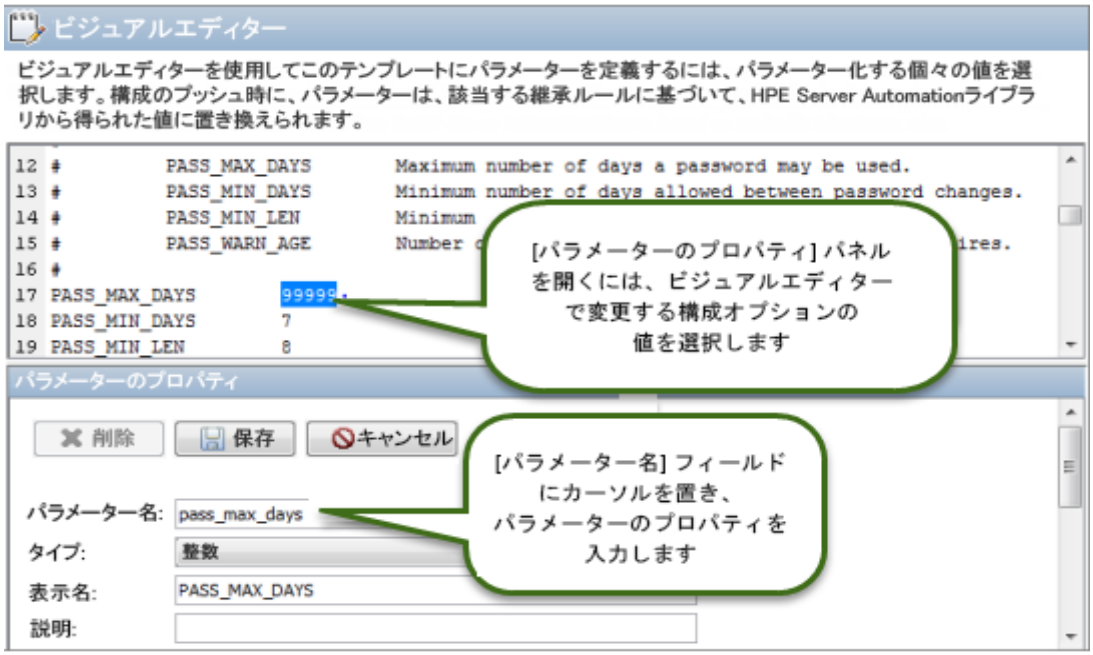
[パラメーターのプロパティ] フォームの [詳細] セクション

ビジュアルエディターインタフェースの仕組み

ビジュアルエディターで、構成をプッシュする際に置き換える構成オプションの値を選択すると、フォームビューが表示されます。

マークの付いたオプションでは、[パラメーターのプロパティ] パネルの [パラメーター名]、[表示名]、[タイプ] (データ型) フィールドが、元になる構成ファイルのデータに基づいて設定されます。フォームビューでは、必要に応じてパラメーターの詳細を変更できます。

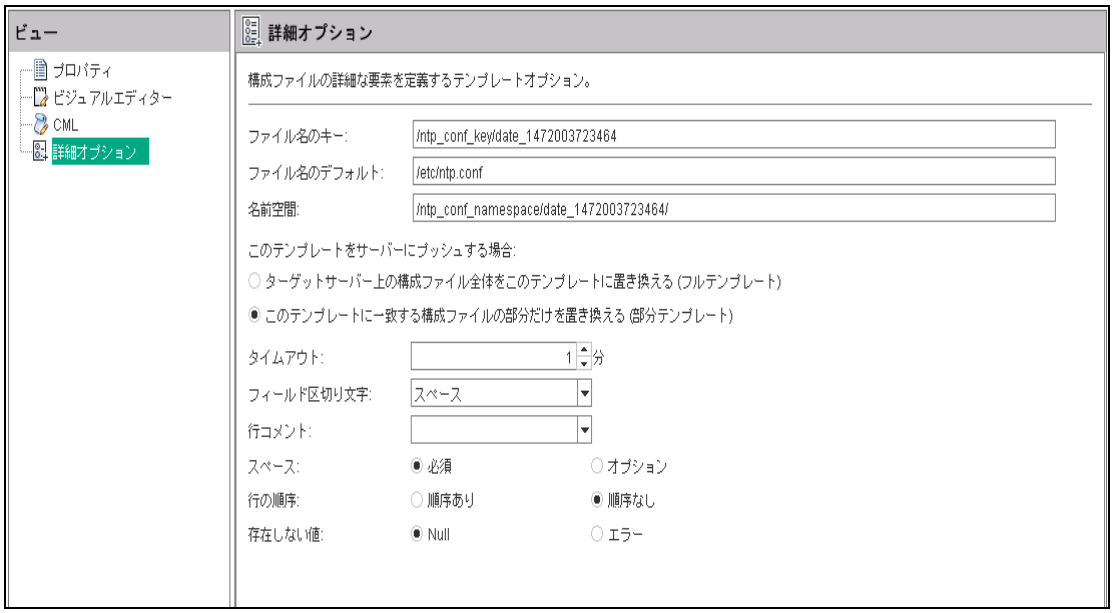
ビジュアルエディターでの構成テンプレートの編集



ビジュアルエディターによる構成テンプレートの編集方法については、「ビジュアルエディターによる構成テンプレートの編集」(271ページ)を参照してください。

ビジュアルエディターの詳細オプション

[詳細オプション] ビューでは、テンプレート内のグローバルCMLオプションを変更できます。このビューにアクセスするには、テンプレートのビューペインで [**詳細オプション**] を選択します。



これらの値はデフォルトに設定され、ユーザーが入力する必要はありませんが、次のような編集は可能です。

- filename-key、filename-default、namespaceの3つの必須フィールドの定義
- 生成するテンプレートが部分テンプレートかフルテンプレートかの指定 (デフォルトは部分テンプレート、すなわち指定した値だけが置換されます)
- プッシュタイムアウト値とその他のCML形式指定および解析オプション

> **注:** ビジュアルエディターの制限: ビジュアルエディターは、CML内容の一部しかサポートしません。CMLが直接編集された結果、ビジュアルエディターでサポートされない形式になった場合には、互換性がないことを示す警告メッセージが表示されます。この場合、変更を保持してビジュアルエディターを使用不可にするか、変更を破棄してビジュアルエディターを使用し続けるかを選択できます。

ビジュアルエディターによる構成テンプレートの編集

この項では、ビジュアルエディターのフローに関する基本的な説明と、ビジュアルエディターで構成テンプレートを編集するためのヒントを紹介します。ビジュアルエディターを開く手順については、「ビジュアルエディターによる構成テンプレートの作成」(264ページ)を参照してください。

構成テンプレートを編集するには、次の手順を実行します。

1. ビジュアルエディターを開きます。
   「テンプレートのインポートと作成」(265ページ)または「使用中の構成ファイルからのテンプレートの作

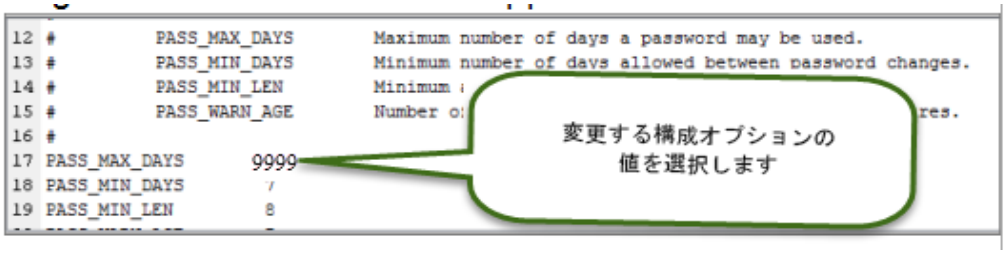
成」(265ページ)を参照してください。

2. プッシュするパラメーターを定義します。
   例については、「単純なパラメーターの作成」(272ページ)または「シーケンスパラメーターの作成」(274ページ)を参照してください。
3. [**プロパティ**] タブを確認して、テンプレートの名前と保存する場所を指定します。
4. [**保存**] をクリックしてテンプレートを保存します。

単純なパラメーターの作成

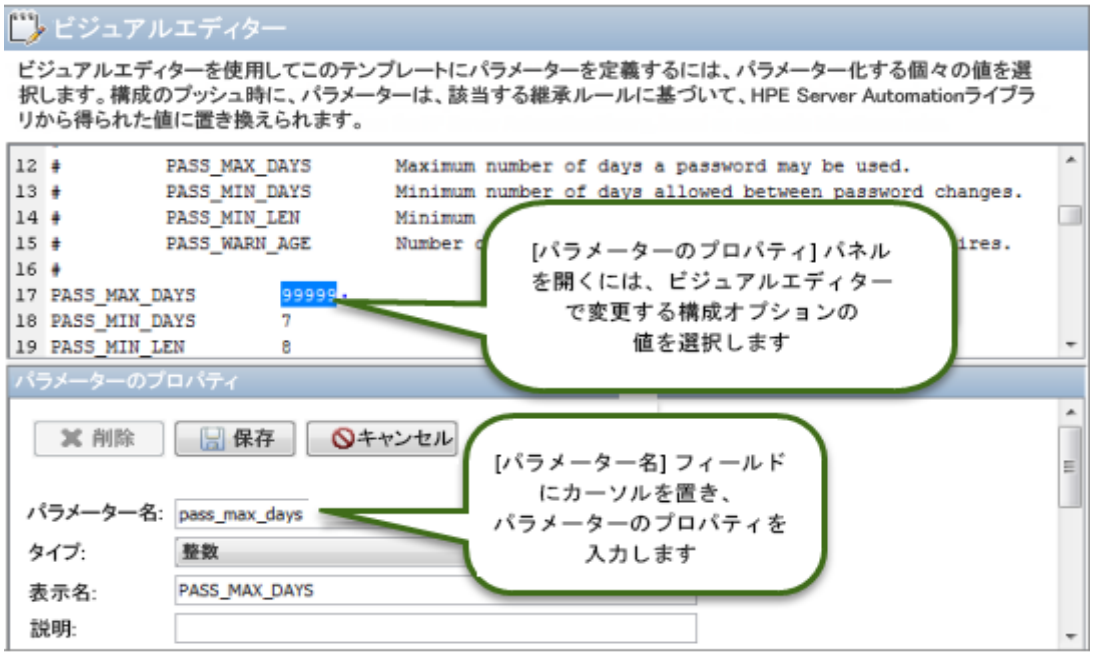
単純なパラメーターを作成するには、次の手順を実行します。

1. 構成をプッシュする際に置き換える構成オプションを選択すると、フォームビューが表示されます。

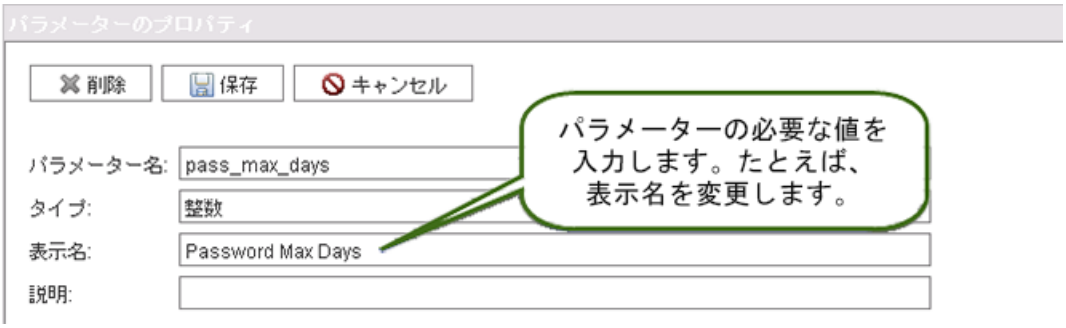


2. [パラメーターのプロパティ] パネルで、[**パラメーター名**] フィールドをクリックすると、マークの付いたオプションで、[**パラメーター名**]、[**表示名**]、[**データ型**] の各フィールドが、元になるテンプレートに基づいて

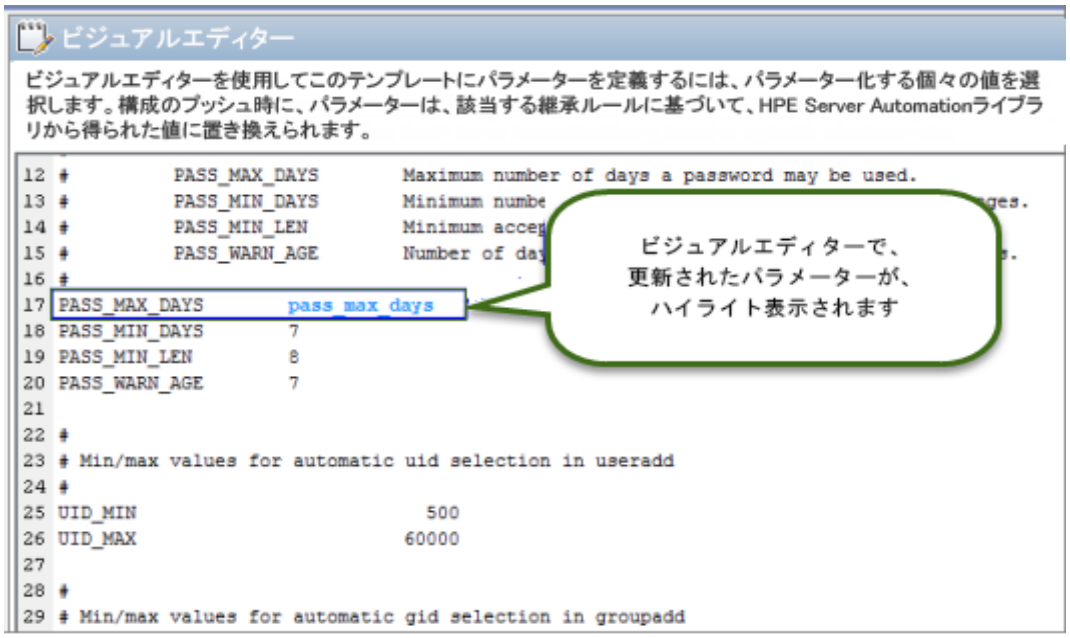
設定されます。



3. [**パラメーター名**] と [**表示名**] の値を入力し、[**保存**] をクリックしてパラメーターを保存します。



4. 新しく作成したパラメーターは、ビジュアルエディターで青で強調表示されます。



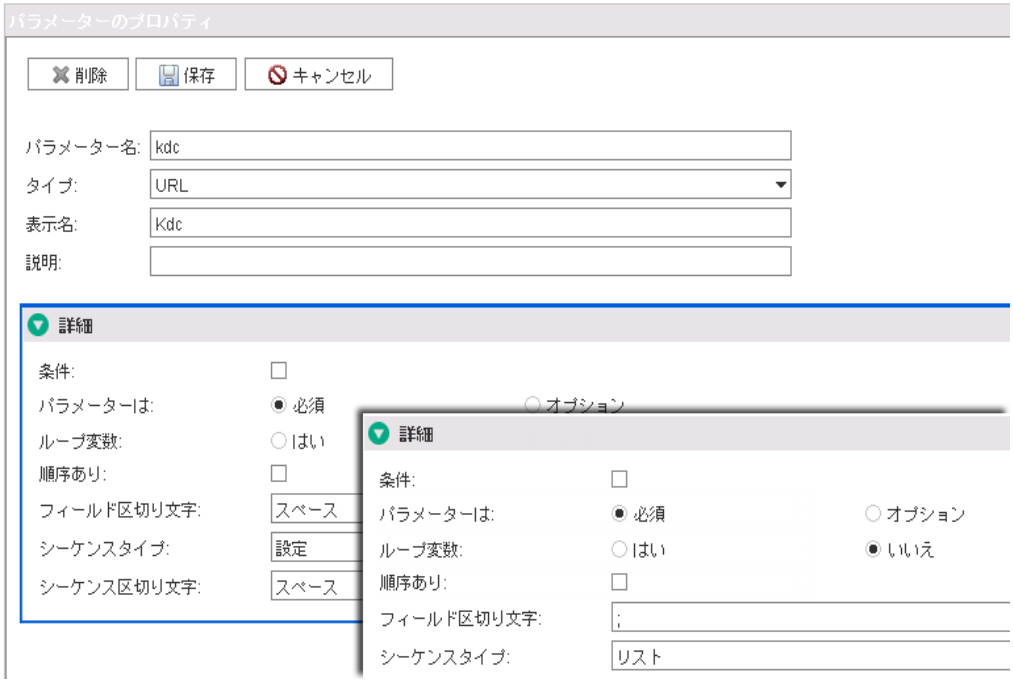
シーケンスパラメーターの作成

シーケンスパラメーターを作成するには、次の手順を実行します。

1. 単純なパラメーターを作成したときと同じ手順を実行します(「単純なパラメーターの作成」(272ページ)を参照してください)。
2. [詳細] セクションで、[**シーケンスタイプ**] フィールドに値を指定します。値はセットまたはリストです。

   リストは要素が特定の順序で並んでいる必要があり、セットは重複する要素を持つことができませ

ん。



3. シーケンス区切り文字を指定します。これは、リストの値を区切るために使用する文字のタイプです。

ビジュアルエディターパネルでシーケンスパラメーターを作成する場合、構成ファイル内でパラメーターに一致するすべての要素が同じパラメーター名で強調表示されます。

構成ファイルの管理

構成ファイルの管理機能では、既存のテンプレートを使用して使用中の構成を簡単にモデル化できます。

> **注:** この操作を開始するには、ファイルシステムのルートディレクトリにアクセスするためのOGFSアクセス権が必要です。また、この操作を完了するには、自分のホームフォルダーへの書き込みアクセス権も必要です。

既存のテンプレートを使用して使用中の構成をモデル化するには、次の手順を実行します。

1. サーバーブラウザーを開きます。
   a. SAクライアントのナビゲーションペインで、管理対象サーバーまたはデバイスグループのリストにアクセスします。

i. [**デバイス**] > [**サーバー**] > [**すべての管理対象サーバー**] を選択して、サーバーリストを表示します。

ii. [**デバイス**] > [**デバイスグループ**] を選択して、デバイスグループリストを表示します。

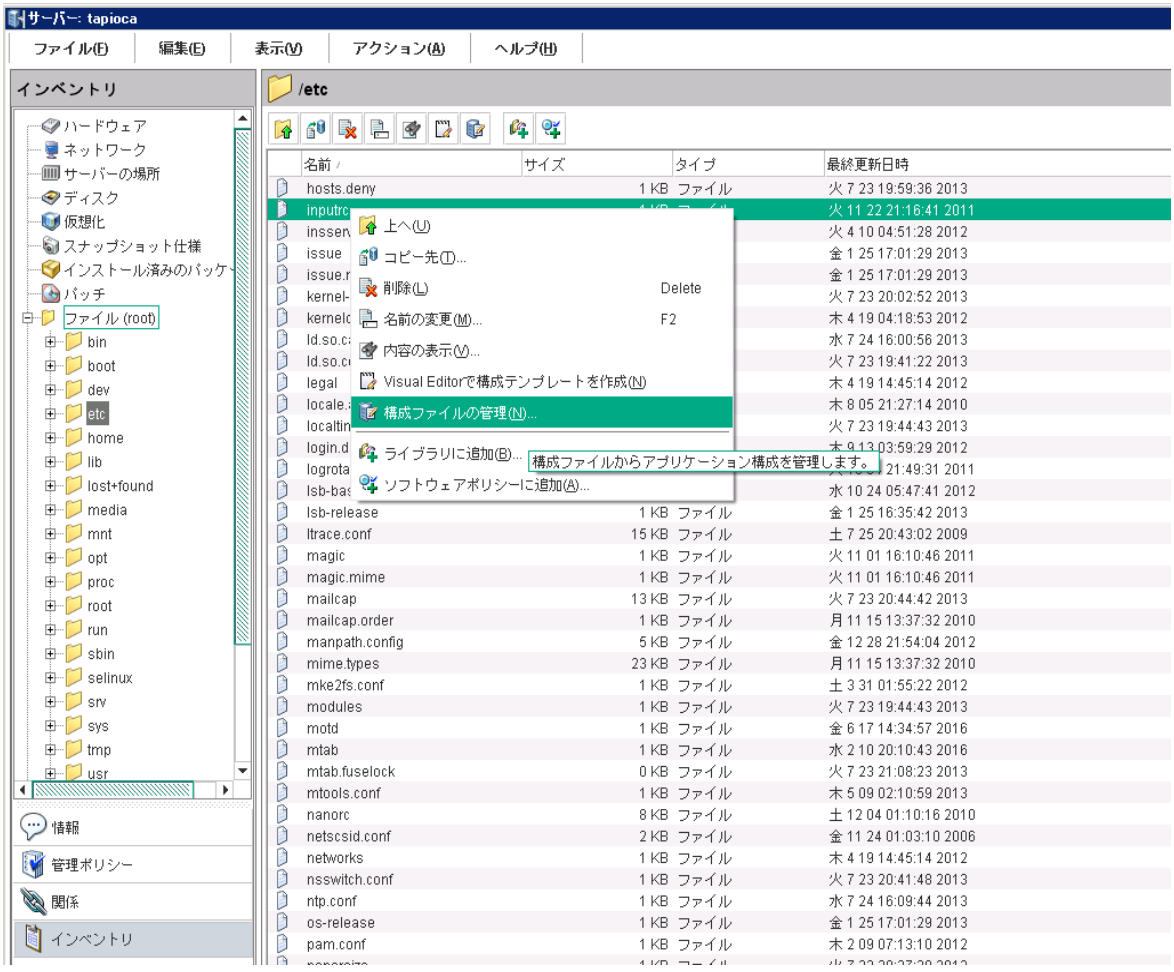
b. 内容ペインで、開くサーバーまたはデバイスグループを選択します。

c. [**アクション**] メニューから [**開く**] を選択します。

2. [**サーバーブラウザー**] > [**インベントリ**] > [**ファイル**] に移動します。(この手順では、OGFSアクセス権を持っていることが必要です)。

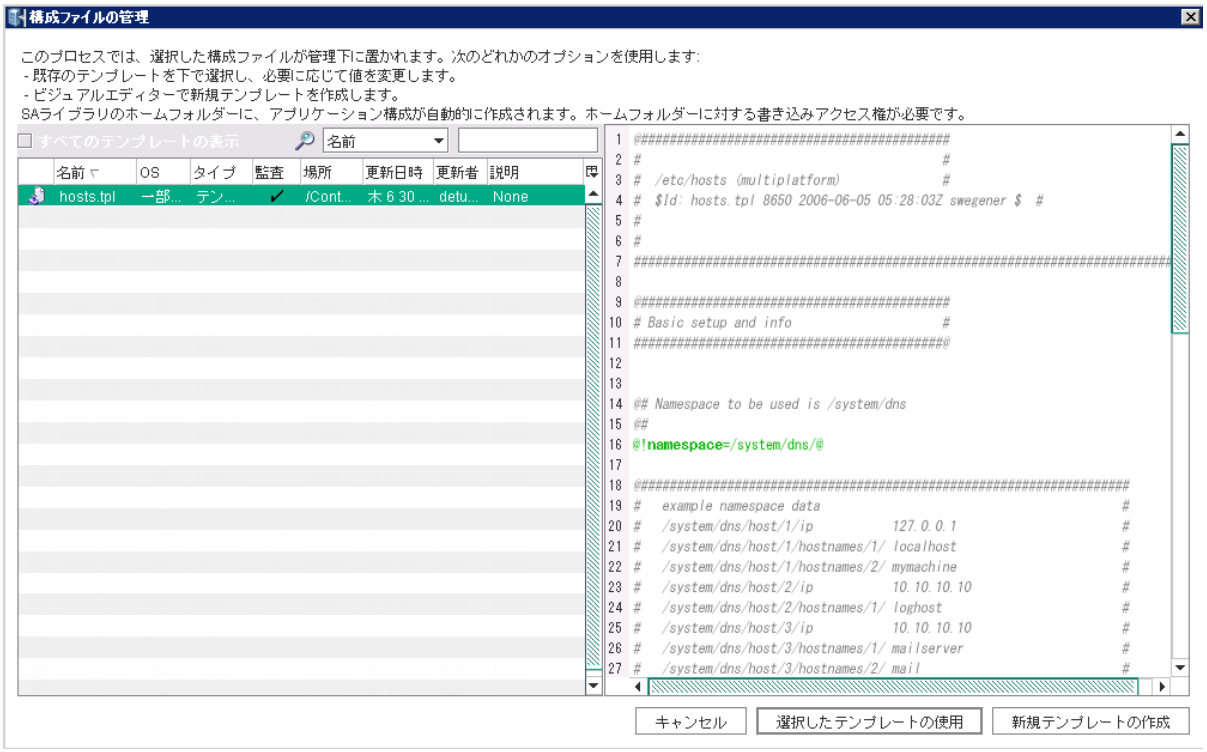
   指示に従って、サーバーファイルシステムのルートパスを選択します (通常、Unixではroot、WindowsではAdministrator)。

3. 構成ファイルを右クリックして、[**構成ファイルの管理**] を選択します。



構成ファイル名とパスに一致するSAの既存のテンプレートが候補としてリストされます。サーバープラットフォームに一致するテンプレートだけが候補になります。

4. テンプレートを選択して、内容をプレビューペインに表示します。



代替オプション:

a. [すべてのテンプレートの表示] チェックボックスを選択すると、リストが拡張され、サーバーのプラットフォームに一致するすべてのテンプレートが含められます。

b. [**キャンセル**] をクリックすると、このウィンドウが閉じ、サーバーファイルシステムのディレクトリに戻ります。

c. [**新規テンプレートの作成**] をクリックすると、選択した構成ファイルに基づいて新しいテンプレートをビジュアルエディターで作成できます (「使用中の構成ファイルからのテンプレートの作成」(265ページ)または「ビジュアルエディターによる構成テンプレートの編集」(271ページ)を参照してください)。

   i. [新規テンプレートの作成] を選択した場合、テンプレートを保存して閉じると、アプリケーション構成を生成してインスタンスをサーバーにアタッチするかどうかを尋ねられます。

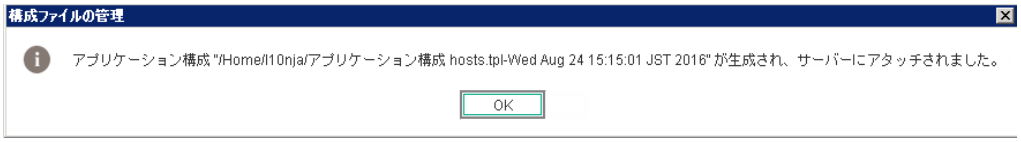
   ii. オプション: 単にテンプレートを保存して後でまた開きたい場合は、[**いいえ**] をクリックします。テンプレートは [プロパティ] タブに指定したディレクトリに保存されます。

   iii. [**はい**] をクリックすると、アプリケーション構成インスタンスがただちに生成され、サーバーにアタッチされます。構成インスタンスは、値セットエディターに表示されます。

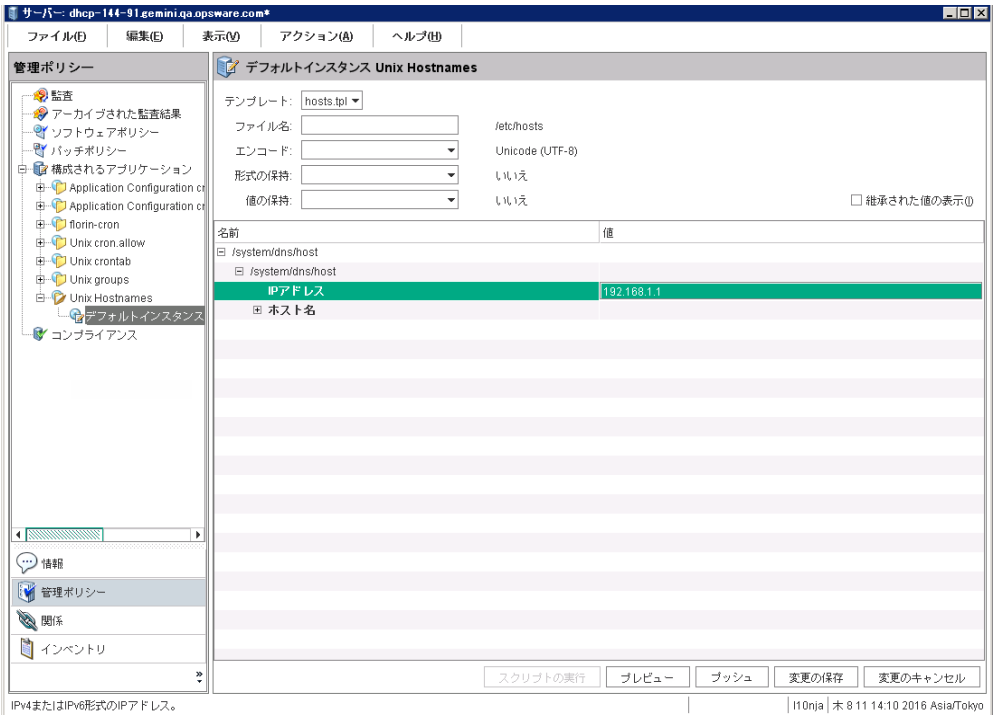
5. [**選択したテンプレートの使用**] をクリックして、選択したテンプレートを使用して新しいアプリケーション構成を作成します(この手順では、自分のホームフォルダーへのアクセス権が必要です)。

新しく作成した構成ファイルがSAライブラリのホームフォルダーに格納され、そのインスタンスが管理対象サーバーにアタッチされます。

構成ファイルが作成され、サーバーにアタッチされたことを示す確認メッセージが表示されます。

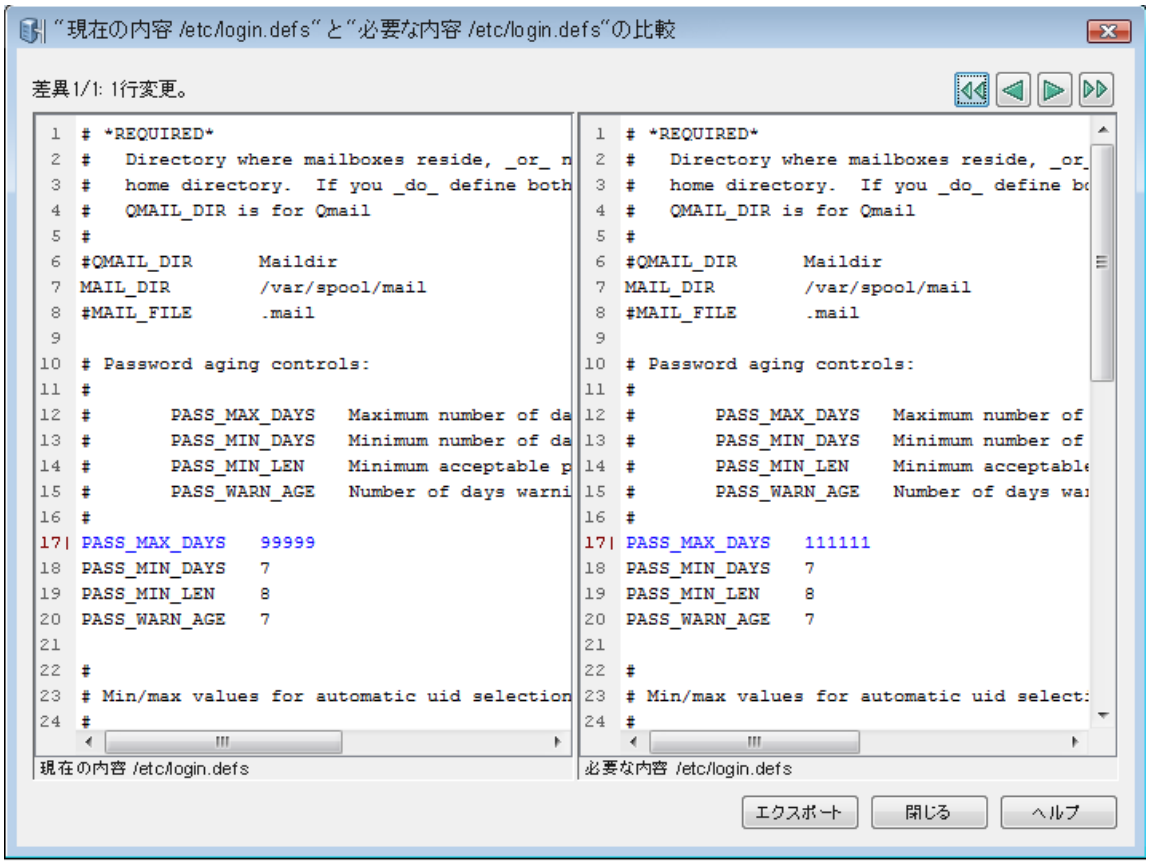


6. [**OK**] をクリックします。構成インスタンスは、値セットエディターに表示されます。



7. このインスタンスの値セットデータを入力します (「値セットエディターでの値の設定」(235ページ)を参照してください)。

8. (オプション) [**プレビュー**] をクリックして、変更結果を現在の値と並べてプレビューします。[**閉じる**] をクリックして、アプリケーション構成インスタンスビューに戻ります。



9. [**プッシュ**] をクリックして、変更をサーバーにプッシュします。

テンプレート内のCMLまたはXMLの編集

テンプレートのCMLまたはXMLは、[内容] ビューで編集できます。テンプレートエディターでは、次に示す編集操作と構文の強調表示が使用できます。

テンプレートのCMLまたはXMLを編集するには、次の操作を実行します。

1. SAクライアントナビゲーションペインで、[**ライブラリ**] を選択し、[**タイプ別**] タブを選択します。
2. [アプリケーション構成] ノードを見つけて開きます。[テンプレート] ノードを開きます。オペレーティングシステムグループを開き、テンプレートファイルが存在するオペレーティングシステムを選択します。テンプレートは複数のオペレーティングシステムに適用することもできます。
3. テンプレートを選択します。

4. [**アクション**] > [**開く**] メニューを選択するか、右クリックして [開く] メニューを選択するか、Enterキーを押します。[テンプレート] 画面が表示され、選択したテンプレートが表示されます。
5. [内容] ビューを選択します。テンプレートの内容が表示されます。
6. CMLまたはXMLテキストをテンプレートエディターに直接入力します。

   CML構文は、読みやすいように次の色で強調表示されます。
   - 緑: CML命令は緑で表示されます。CMLキーワードは太字の緑です。
   - 青: 値セットから実際の値に置き換えられる変数は青で表示されます。
   - 黒: 固定テキストは黒で表示されます。
   - グレー: コメントはグレーで表示されます。

   CMLエディターでは、次の編集操作も実行できます。
   - CMLテキストを右クリックし、切り取り、コピー、貼り付け操作を使用できます。
   - [アクション] メニューには、検索、置換、元に戻す、やり直しの機能があります。
   - [検証] ボタンと [**アクション**] > [**検証**] メニューは、テンプレートの構文をチェックし、エラーを報告します。

CMLとXMLの詳細については、「CMLリファレンス」(375ページ)および「XML構成ファイルの管理」(301ページ)を参照してください。

テンプレートファイルのインポートと検証

CMLテンプレートまたはXMLテンプレートをテキストエディターで作成してから、SAライブラリにインポートして、アプリケーション構成で使用することができます。また、インポート前にテンプレートをSAで検証することもできます (「構成テンプレートとスクリプトテンプレート」(230ページ)を参照してください)。

Windowsサーバー上の構成ファイルのうち、UTF-8でエンコードされているものは、構成ファイルの最初の3文字にバイト順序マーク (BOM) が含まれている可能性があります。このファイルをアプリケーション構成テンプレートにインポートした場合、ファイルのインポート後にBOMがテンプレートに現れます。このBOMをアプリケーション構成テンプレートに含めたくない場合は、構成ファイルをテンプレートにアップロードした後で削除します。

SAクライアントではUTF-16エンコードはサポートされていません。

テンプレートファイルを検証してインポートするには、次の手順を実行します。

1. SAクライアントで、ナビゲーションペインを選択し、[ライブラリ] を選択し、[タイプ別] タブを選択します。
2. [アプリケーション構成] ノードを見つけて開きます。[テンプレート] ノードを開きます。オペレーティングシステムグループを開き、テンプレートファイルが使用されるオペレーティングシステムを選択します。テンプレートは複数のオペレーティングシステムに適用することもできます。これは後の手順で指定します。
3. [**アクション**] > [**テンプレートの検証...**] メニューを選択します。
4. テンプレートファイルを見つけて選択し、適切なエンコードを選択して、[開く] を選択します。SAはテンプレートの構文をチェックし、結果を報告します。ファイルにエラーがある場合は、修正してから再検証します。
5. [**アクション**] > [**テンプレートのインポート...**] メニューを選択します。
6. テンプレートファイルを見つけて選択し、適切なエンコードを選択して、[開く] を選択します。なお、SAクライアントではUTF-16エンコードはサポートされていません。SAはテンプレートをインポートし、[テンプレート] 画面を表示します。
7. 「構成テンプレートの作成」(262ページ)の手順の手順4以降を実行します。

構成テンプレートソースの表示

構成テンプレートの内容を表示し、そのCMLまたはXMLソースを表示できます。これは、アプリケーション構成をサーバーにプッシュする前にテンプレートに設定されていたリストマージモードを知るために役立ちます。アプリケーション構成のシーケンスマージモードの詳細については、「シーケンスの集約」(417ページ)を参照してください。

構成テンプレートのソースを表示するには、次の手順を実行します。

1. SAクライアントナビゲーションペインで、[ライブラリ] を選択し、[タイプ別] タブを選択します。
2. [アプリケーション構成] ノードを見つけて開きます。[テンプレート] ノードを開きます。オペレーティングシステムグループを開き、テンプレートファイルが存在するオペレーティングシステムに移動します。テンプレートは複数のオペレーティングシステムに適用することもできます。
3. 構成テンプレートを選択し、[**アクション**] > [**開く**] を選択します。
4. [内容] ビューを選択して、構成テンプレートのCMLまたはXML内容を表示します。

アプリケーション構成に対するテンプレートの追加または削除

アプリケーション構成には、1つ以上のテンプレートが含まれます。

アプリケーション構成に対してテンプレートを追加または削除するには、次の手順を実行します。

1. SAクライアントナビゲーションペインで、[ライブラリ] を選択し、[**タイプ別**] タブを選択します。
2. [アプリケーション構成] ノードを見つけて開きます。[構成] ノードを開きます。オペレーティングシステムグループを開き、アプリケーション構成が存在するオペレーティングシステムに移動します。アプリケーション構成は複数のオペレーティングシステムに適用することもできます。
3. アプリケーション構成を選択し、[**アクション**] > [**開く**] を選択します。
4. [構成される値] ビューを選択します。
5. テンプレートをアプリケーション構成に追加するには、[**アクション**] > [**追加**] を選択するか、[+] ボタンを選択します。目的のテンプレートを選択して、[**OK**] を選択します。

   テンプレートは、構成に該当するすべてのプラットフォームに適用できる必要があります。つまり、構成対象のオペレーティングシステムは、当該テンプレートのサブセットである必要があります。

   テンプレートを含むフォルダーのカスタマー設定には、アプリケーション構成オブジェクトのカスタマー設定が含まれる必要があります。そうでないと、テンプレートは利用可能なテンプレートのリストに含められません。フォルダー設定の詳細については、『SA管理ガイド』の「フォルダーのアクセス権」を参照してください。
6. テンプレートを削除するには、テンプレートを選択して、[**アクション**] > [**削除**] を選択するか、[-] ボタンを選択します。
7. 変更内容を保存する場合は、[**ファイル**] > [**保存**] を選択します。

アプリケーション構成でのテンプレート順序の指定

アプリケーション構成には、1つ以上の構成テンプレートと関連するスクリプトを含めることができます。アプリケーション構成でのテンプレートの順序を指定できます。テンプレートは、アプリケーション構成に現れる

順序で、管理対象サーバーにプッシュされます。たとえば、特定の構成ファイルへの変更を他のものより前に適用することが必要な場合があります。

アプリケーション構成内のスクリプトの実行順序は、スクリプトのタイプ(データ操作、インストール前、インストール後、エラー後)によって決まります。アプリケーション構成内のスクリプトの順序は無関係です。詳細については、「アプリケーション構成スクリプトのタイプ」(247ページ)を参照してください。

アプリケーション構成でのテンプレートの順序を指定するには、次の手順を実行します。

1. SAクライアントナビゲーションペインで、[ライブラリ]を選択し、[タイプ別]タブを選択します。
2. アプリケーション構成ノードを見つけて開きます。[構成]ノードを開きます。オペレーティングシステムグループを開き、アプリケーション構成が存在するオペレーティングシステムに移動します。アプリケーション構成は複数のオペレーティングシステムに適用することもできます。
3. アプリケーション構成を選択し、[**アクション**] > [**開く**]を選択します。
4. [**構成される値**]ビューを選択します。アプリケーション構成内のすべての構成テンプレートとスクリプト(存在する場合)が表示されます。テンプレートとスクリプトには順序を示す番号が付いています。
5. テンプレートまたはスクリプトの順序を変更するには、選択してから[**アクション**] > [**上に移動**]または[**アクション**] > [**下に移動**]を選択するか、上矢印または下矢印アイコンを選択します。
6. 変更内容を保存する場合は、[**ファイル**] > [**保存**]を選択します。

スクリプトからのテンプレートの作成

アプリケーション構成オブジェクトにスクリプトを含めるには、スクリプトをCMLテンプレートにコピーしてから、テンプレートをアプリケーション構成オブジェクトにインポートする必要があります。CMLの詳細については、「CMLリファレンス」(375ページ)を参照してください。「アプリケーション構成でのスクリプトの実行」(247ページ)を参照してください。

次の例は、touchコマンドとechoコマンドを実行する単純なUnixシェルスクリプトを示します。

```
#/bin/sh
touch abc.txt
echo abc >>abc.txt
```

次の例は、このUnixシェルスクリプトをCMLに変換したものです。

```
@#################################################################
# /tmp/simple-script/TouchABC.sh                                #
# Version 0.1                                                   #
# Author: <name>                                                #
#################################################################@
@!namespace=/simple-script-namespace/@
@!filename-key="/TouchABC"@
@!filename-default=/tmp/simple-script/TouchABC.sh@
```

```
#/bin/sh
touch abc.txt
echo abc >>abc.txt
```

このスクリプトからテンプレートを作成するには、次の手順を実行します。

1. 「構成テンプレートの作成」(262ページ)の手順に従ってテンプレートを作成します。スクリプトのCMLバージョンをテンプレートの内容として使用します。
2. [タイプ] フィールドを適切なスクリプトタイプに設定します。上記の例では、[タイプ] を [Unix .SHスクリプト] に設定します。サポートされる他のスクリプトタイプの一覧は「アプリケーション構成スクリプトのタイプ」(247ページ)にあります。
3. [パーサー構文] フィールドを [CML構文] に設定します。すべてのスクリプトはCML構文で作成する必要があるからです。
4. 残りのフィールドを「構成テンプレートの作成」(262ページ)の説明のように設定します。
5. **[ファイル] > [保存]** を選択します。
6. **[ファイル] > [閉じる]** を選択します。
7. 「アプリケーション構成に対するテンプレートの追加または削除」(282ページ)の手順に従って、テンプレートをアプリケーション構成オブジェクトに追加します。
8. テンプレートをアプリケーション構成オブジェクトに追加したら、アプリケーション構成オブジェクトを開きます。
9. [**構成される値**] ビューを選択します。
10. スクリプトテンプレートを選択し、右クリックしてメニューを表示します。
11. スクリプトタイプを選択します。これは、スクリプトが実行されるタイミングを指定します。選択できるのは、データ操作、インストール前、インストール後、エラー後です。
12. **[ファイル] > [保存]** を選択します。
13. **[ファイル] > [閉じる]** を選択します。

データ操作スクリプトの実行による非テキスト構成の管理

SAでは、非テキスト構成を管理するために、データ操作スクリプトを作成し、非テキストデータを抽出してテキストファイルに格納できます。結果のテキストファイルは、他のテキスト構成ファイルと同じ方法でSAで管理でき、元の形式に戻すこともできます。スクリプトタイプの説明については、「アプリケーション構成スクリプトのタイプ」(247ページ)を参照してください。

非テキスト構成データの例としては、次のものがあります。

- SQLデータベース。データ操作スクリプトでSQLクエリを実行し、データをテキストファイルに格納できます。
- IISサーバーの構成。データ操作スクリプトでメタベース情報を読み取って、テキストファイルに格納できます。
- バイナリファイル。データ操作スクリプトで値を抽出して、テキストファイルに格納できます。

データ操作スクリプトの手動での実行

[スクリプトの実行] ボタンを使用すると、アプリケーション構成に関連するデータ操作スクリプトを実行し、管理対象サーバー上にターゲット構成ファイルを準備して、その値を値セットにインポートできます。

データ操作スクリプトを手動で実行するには、次の手順を実行します。

1. SAクライアントナビゲーションペインで、[デバイス] タブを選択します。
2. [すべての管理対象サーバー] または [デバイスグループ] を選択します。目的のサーバーまたはデバイスグループに移動します。
3. サーバーまたはデバイスグループを選択し、[**アクション**] > [**開く**] メニューを選択します。
4. サーバーを選択した場合は、下記の手順を実行します。デバイスグループを選択した場合は、次のステップまでスキップします。
   a. [管理ポリシー] タブを選択します。
   b. ナビゲーションペインで、[構成されるアプリケーション] ノードを開きます。サーバーにアタッチされているすべてのアプリケーション構成が表示されます。
   c. プッシュするアプリケーション構成ノードを選択します。アプリケーション構成に対応する値セットが表示されます。
   d. プッシュするアプリケーション構成インスタンスを選択します。
5. デバイスグループを選択した場合は、次の手順を実行します。
   a. ナビゲーションペインで、[構成されるアプリケーション] ノードを開きます。サーバーにアタッチされているすべてのアプリケーション構成と、すべてのサーバーのリストが表示されます。
   b. [サーバー] ノードを開きます。デバイスグループのすべてのサーバーが表示されます。
   c. データ操作スクリプトを実行するサーバーノードを選択します。そのサーバーにアタッチされているアプリケーション構成が表示されます。
   d. プッシュするアプリケーション構成ノードを選択します。アプリケーション構成に対応する値セットが表示されます。

e. プッシュする値セットを選択します。サーバーインスタンスレベルの値セットを選択する必要があります。

6. サーバーインスタンスレベルの値セットが選択された状態で、[スクリプトの実行] ボタンを選択します。確認ダイアログが表示されます。
7. [はい] を選択して、データ操作スクリプトをサーバー上で実行します。
8. データ操作スクリプトが動作し、構成データを抽出してテキストファイルに格納したら、テキストファイルを他の構成ファイルと同じ方法で管理し、データをテキストファイルから元の形式に戻すことができます。

サーバーまたはデバイスグループへのアプリケーション構成のアタッチ

アプリケーション構成を作成し、必要なすべての構成テンプレートとスクリプトを追加して、デフォルト値を編集したら、サーバーまたはパブリックデバイスグループにアタッチする必要があります。アプリケーション構成をサーバーまたはサーバーグループにアタッチしたら、「アプリケーション構成のプッシュ」(290ページ)の手順に従って、アプリケーション構成をサーバーにプッシュします。

> - アプリケーション構成を含むフォルダーのカスタマー設定には、アプリケーション構成をプッシュする管理対象サーバーのカスタマー設定が含まれる必要があります。フォルダー設定の詳細については、『SA管理ガイド』の「フォルダーのアクセス権」を参照してください。サーバーとカスタマーの詳細については、『ユーザーガイド』の「カスタマーアカウント」を参照してください。
> - アプリケーション構成は、個々のサーバーまたはパブリックデバイスグループのみにアタッチできます。プライベートデバイスグループにはアタッチできません。

1つのサーバーへのアプリケーション構成のアタッチ

ショートカット: [**デバイス**] タブ > [**サーバー**] を選択 > [**すべての管理対象サーバー**] > 右クリック > [**アタッチ**] > [**アプリケーション構成**] > 構成ファイルを選択 > [**OK**] をクリック。

アプリケーション構成**を1つのサーバーにアタッチするには、次の手順を実行します。**

1. SAクライアントナビゲーションペインで、[**デバイス**] タブを選択します。
2. [**サーバー**] > [**すべての管理対象サーバー**] を選択します。
3. 内容ペインでサーバーを選択します。

4. [**アクション**] > [**開く**] メニューを選択します。
5. [**管理ポリシー**] タブを選択し、[**構成されるアプリケーション**] を選択します。
6. [インストール済み構成] タブを選択します。
7. [**アクション**] > [**構成の追加...**] メニューを選択します。
8. [アプリケーション構成の選択] 画面で、管理対象サーバーにアタッチするアプリケーション構成を選択します。
   検索ツールを使用すると、名前、最終更新日などの基準によって検索できます。

   > **注:**
   > アプリケーション構成を含むフォルダーのカスタマー設定には、アプリケーション構成をプッシュする管理対象サーバーのカスタマー設定が含まれる必要があります。フォルダー設定の詳細については、『SA 10.50管理ガイド』の「フォルダーのアクセス権」を参照してください。サーバーとカスタマーの詳細については、『SA 10.50ユーザーガイド』の「カスタマーアカウント」を参照してください。

9. [**OK**] を選択して、アプリケーション構成をサーバーにアタッチします。
10. [変更の保存] ボタンを選択します。
11. その後、このサーバーに対するアプリケーション構成の値を設定できます。アプリケーション構成の値の設定方法の詳細については、「値セット」(232ページ)を参照してください。

デバイスグループへのアプリケーション構成のアタッチ

アプリケーション構成**をデバイスグループにアタッチするには、次の手順を実行します。**

アプリケーション構成はパブリックデバイスグループのみにアタッチできます。アプリケーション構成をプライベートデバイスグループにアタッチすることはできません。

1. SAクライアントナビゲーションペインで、[デバイス] タブを選択します。
2. [デバイスグループ] ノードを開き、パブリックデバイスグループに移動します。
3. 内容ペインで、デバイスグループを選択します。
4. [**アクション**] > [**開く**] メニューを選択します。
5. デバイスグループ画面で、[構成されるアプリケーション] ビューを選択します。
6. [**アクション**] > [**構成の追加**] メニューを選択します。
7. [アプリケーション構成の選択] ダイアログボックスで、アプリケーション構成を選択します。
   検索ツールを使用すると、名前、最終更新日などの基準によって検索できます。

8. インスタンス名を入力します。これは、グループインスタンスレベルの値セットの名前です。詳細については、「グループインスタンスレベルでの値の設定」(244ページ)を参照してください。
9. [**OK**] を選択して、アプリケーション構成をデバイスグループにアタッチします。指定した構成ファイルをグループ内のすべてのサーバーにプッシュできます。
10. サーバーにプッシュする値を設定します。値の設定の詳細については、「値セット」(232ページ)を参照してください。

サーバーまたはデバイスグループからのアプリケーション構成のデタッチ

サーバーからアプリケーション構成をデタッチするには、サーバーを開き、サーバーインスタンスレベルのアプリケーション構成のインスタンスをすべて削除する必要があります。デバイスグループからアプリケーション構成をデタッチするには、デバイスグループを開き、グループインスタンスレベルのインスタンスをすべて削除する必要があります。以下に詳細を示します。

サーバーからのアプリケーション構成のデタッチ

サーバーからアプリケーション構成をデタッチするには、下記の手順でサーバーインスタンスレベルのインスタンスをすべて削除する必要があります。詳細については、「サーバーレベルでの値の設定」(246ページ)を参照してください。

サーバーからアプリケーション構成をデタッチするには、次の手順を実行します。

1. SAクライアントでサーバーを開きます。アプリケーション構成がサーバーにアタッチされている必要があります。「サーバーまたはデバイスグループへのアプリケーション構成のアタッチ」(286ページ)を参照してください。
2. 左側の[管理ポリシー] タブを選択します。
3. 左側の[管理ポリシー] ペインで、[構成されるアプリケーション] ノードを開きます。
4. [構成されるアプリケーション] ノードの下で、目的のアプリケーション構成ノードを開きます。サーバーインスタンスレベルにあるアプリケーション構成のすべてのインスタンスが表示されます。
5. アプリケーション構成ノードの下で、インスタンスを1つ選択します。
6. [**アクション**] > [**構成の削除**] メニューを選択するか、右クリックして [**構成の削除**] メニューを選択します。これにより、選択したインスタンスが削除されます。

7. 各インスタンスに対して、「アプリケーション構成ノードの下で、インスタンスを1つ選択します。」(288ページ)と「[アクション] > [構成の削除] メニューを選択するか、右クリックして [構成の削除] メニューを選択します。これにより、選択したインスタンスが削除されます。」(288ページ)を繰り返します。最後のインスタンスを削除すると、アプリケーション構成がサーバーからデタッチされます。
8. [**変更の保存**] をクリックします。

デバイスグループからのアプリケーション構成のデタッチ

デバイスグループからアプリケーション構成をデタッチするには、下記の手順でグループインスタンスレベルのアプリケーション構成のインスタンスをすべて削除する必要があります。詳細については、「グループレベルでの値の設定」(243ページ)を参照してください。

デバイスグループからアプリケーション構成をデタッチするには、次の手順を実行します。

1. SAクライアントナビゲーションペインで、[デバイス] タブを選択します。
2. [デバイスグループ] ノードを開き、パブリックデバイスグループに移動します。
3. 内容ペインで、デバイスグループを選択します。
4. [**アクション**] > [**開く**] メニューを選択します。
5. デバイスグループ画面で、[構成されるアプリケーション] ビューを選択し、[構成されるアプリケーション] ノードを開きます。デバイスグループにアタッチされているアプリケーション構成が表示されます。
6. [構成されるアプリケーション] ノードの下で、目的のアプリケーション構成ノードを開きます。グループインスタンスレベルの値セットが表示されます。
7. 目的のアプリケーション構成ノードの下で、グループインスタンスレベルのアプリケーション構成インスタンスを1つ選択します。
8. [**アクション**] > [**構成の削除**] メニューを選択するか、右クリックして [**構成の削除**] メニューを選択します。これにより、選択したインスタンスが削除されます。
9. サーバーインスタンスレベルの各インスタンスに対して、上記の手順7と手順8を繰り返します。最後のインスタンスを削除すると、アプリケーション構成がサーバーからデタッチされます。
10. [**変更の保存**] をクリックします。

アプリケーション構成のプッシュ

値セットの値を変更した場合、その変更をターゲットサーバー上の構成ファイルにマージするには、アプリケーション構成をサーバーにプッシュする必要があります。詳細については、「アプリケーション構成のサーバーへのプッシュ」(249ページ)を参照してください。

アプリケーション構成の変更をサーバーまたはサーバーグループにプッシュするには、次の手順を実行します。

1. SAクライアントナビゲーションペインで、[**デバイス**] タブを選択します。
2. [すべての管理対象サーバー] または [デバイスグループ] を選択します。目的のサーバーまたはデバイスグループに移動します。
3. サーバーまたはデバイスグループを選択し、[**アクション**] > [**開く**] メニューを選択します。
   - サーバーを選択した場合、[管理ポリシー] タブを選択します。
   - デバイスグループを選択した場合は、次のステップまでスキップします。
4. ナビゲーションペインで [構成されるアプリケーション] ノードを開き、プッシュするアプリケーション構成インスタンスを選択します。

   オプションで、[**プレビュー**] ボタンを選択することにより、個々のサーバーに対して行われる変更をプレビューできます。比較画面に差異が表示されます。終わったら、[**閉じる**] を選択します。
5. サーバーに変更を適用する準備ができたら、[**プッシュ**] を選択します。
6. [構成のプッシュ] 画面で、プッシュされるアプリケーション構成と値セットを確認します。
   残りの [スケジュール設定]、[通知]、[ジョブステータス] をデフォルトのままにする場合は、[**ジョブの開始**] をクリックします。変更する場合は、[**次へ**] をクリックしてウィザードオプションの確認を続行します。
7. [スケジュール設定] ペインで、アプリケーション構成をプッシュする日時を指定します。[スケジュール設定] ペインでは、ジョブを将来のある時点で実行するように設定するか、毎週、毎月などの一定の間隔で定期的に実行するように設定できます。
   残りの [通知]、[ジョブステータス] をデフォルトのままにする場合は、[**ジョブの開始**] をクリックします。変更する場合は、[**次へ**] をクリックしてウィザードオプションの確認を続行します。
8. [通知] ペインで、1つ以上の電子メールアドレスとチケットIDをオプションで指定します。各受信者に対して、電子メール通知の送信条件のオプションを選択します。
   - 成功時: ジョブが成功した場合に電子メールを送信します。
   - 失敗時: ジョブが失敗した場合に電子メールを送信します。

- 終了時: ジョブが終了した場合に電子メールを送信します。
  - ジョブの終了とは、ジョブの終了アクションを使って実行中のジョブを停止した状態を指します。
  - ジョブを開始する前にキャンセルした場合には、電子メールは送信されません。

残りの[ジョブステータス]をデフォルトのままにする場合は、[**ジョブの開始**]をクリックします。変更する場合は、[**次へ**]をクリックしてウィザードオプションの確認を続行します。

9. [**ジョブの開始**]をクリックします。

   ジョブが開始された後でそのステータスを確認するには、メインSAクライアント画面で[ジョブとセッション]タブを選択し、[ジョブログ]を選択します。

   また、オプションで次のアクションを実行することもできます。
   - [**エクスポート**]をクリックすると、ジョブステータス結果をテキストファイルにエクスポートできます。
   - [**ジョブの終了**]をクリックして、ジョブを停止します。「構成のプッシュジョブの停止」(291ページ)を参照してください。
   - ウィンドウを閉じるには、[**閉じる**]をクリックします。後でジョブステータスを確認するには、SAクライアントナビゲーションペインで[**ジョブステータス**]をクリックし、ジョブをダブルクリックして詳細を表示します。

構成のプッシュジョブの停止

アクティブに実行中の構成のプッシュジョブを終了させることができます。ジョブの終了が必要になるケースとしては、たとえば、ジョブの実行結果に誤りがある場合や、予定していたメンテナンス時間枠を超えてしまう場合があります。

ジョブの整合性を維持するため、構成のプッシュジョブのいくつかのステップはキャンセルできません。ジョブを停止すると、[ジョブステータス]ウィンドウに完了したステップとスキップされたステップが表示されます。

アクティブなアプリケーション構成プッシュジョブを停止するには、次の手順を実行します。

1. [ジョブステータス]ウィンドウで[**ジョブの終了**]をクリックします(このボタンは、ジョブが実行中のときだけ表示されます)。
2. [ジョブの終了]警告ダイアログが表示され、ジョブの終了に関する注意事項が示されます。
   - その後のサーバーに対してはジョブの作業は開始されません。
   - すでに作業が開始されているサーバーに対しては、ジョブのステップのうち安全にキャンセルできるものだけがスキップされます。

- [ジョブステータス] に、完了したステップとスキップされたステップが示されます。
- ジョブが正常に終了した場合、最終的なジョブステータスは「終了済み」になります。

3. [**OK**] をクリックして、ジョブの終了を確認します。[ジョブステータス] ウィンドウに、終了処理の進行状況が表示されます。

   ジョブステータスは終了済みになります。サーバーステータスはキャンセルになります。タスクステータスは成功またはスキップ済みになります。

4. 終了が完了したら、SAクライアントジョブログでもジョブを確認できます。

   SAクライアントのナビゲーションペインで、[**ジョブとセッション**] をクリックします。[ジョブログ] ビューが開き、ステータスが [終了済み] のジョブが表示されます。

プッシュタイムアウト値の変更

デフォルトでは、アプリケーション構成をプッシュするときのタイムアウト値は、10分にテンプレート内のアプリケーション構成1つにつき1分を加算した値です。そのアプリケーション構成に含まれるテンプレートのそれぞれが、アプリケーション構成のベースタイムアウトに自分のタイムアウトを加算します。

たとえば、3つのテンプレートを含むアプリケーション構成の場合、アプリケーション構成全体のデフォルトのタイムアウト値は13分です。テンプレートをプッシュしたときに、プッシュ全体にかかる時間が13分を超えた場合、プッシュはタイムアウトし、すでに行われたすべての変更を含めて操作はキャンセルされます。

テンプレートのタイムアウト時間を長くするには、アプリケーション構成内の個々のテンプレートでCMLのtimeoutタグを使用します。CMLのtimeoutタグの構文は次のとおりです。

```
@!timeout=1@
```

有効な値は0～999 (分) です。

プッシュの途中でアプリケーション構成がタイムアウトした場合、プッシュのターゲットファイルに対するすべての変更はバックアウトされ、操作はキャンセルされます。

CMLのtimeoutタグの詳細については、「CMLリファレンス」(375ページ)を参照してください。

アプリケーション構成プッシュのスケジュール設定

アプリケーション構成のプッシュは、ただちに実行するか、将来のある時点で実行するか、毎日、毎週、毎月といった定期的なスケジュールで実行するように設定できます。アプリケーション構成プッシュのスケ

ジュールを設定するには、「アプリケーション構成のプッシュ」(290ページ)の手順を実行して、[スケジュール設定] のステップに達したときに、プッシュを実行する頻度と時刻を入力します。

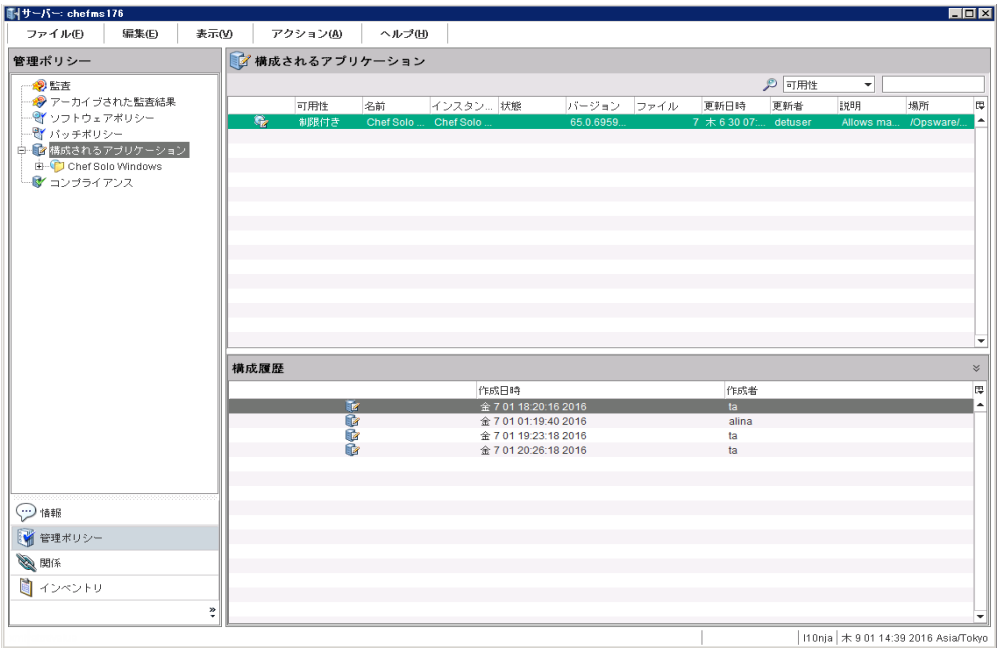
ジョブのスケジュールを設定した後でそのステータスを確認するには、メインSAクライアント画面で [ジョブとセッション] タブを選択し、[定期的スケジュール] を選択します。

構成ファイルの過去の状態への復元

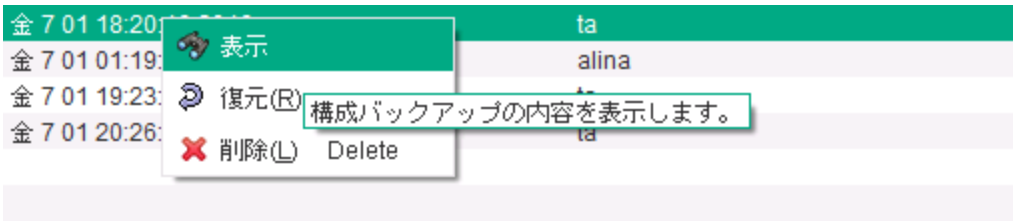
アプリケーション構成をサーバーにプッシュするたびに、構成ファイルは構成プッシュ履歴リストに保存されます。アプリケーション構成はいつでも履歴リスト内の過去の状態に復元できます。

アプリケーション構成を過去の状態に復元するには、次の手順を実行します。

1. SAクライアントナビゲーションペインで、[デバイス] タブを選択します。
2. [すべての管理対象サーバー] を選択し、目的のサーバーを見つけます。
3. サーバーを選択し、[**アクション**] > [**開く**] メニューを選択します。
4. [管理ポリシー] タブを選択します。
5. ナビゲーションペインで、[構成されるアプリケーション] ノードを選択します。サーバーにインストールされているすべてのアプリケーション構成が表示されます。
6. 構成を選択すると、詳細ペインに構成履歴が表示されます。サーバー上で実行されたすべてのプッシュジョブが表示されます。

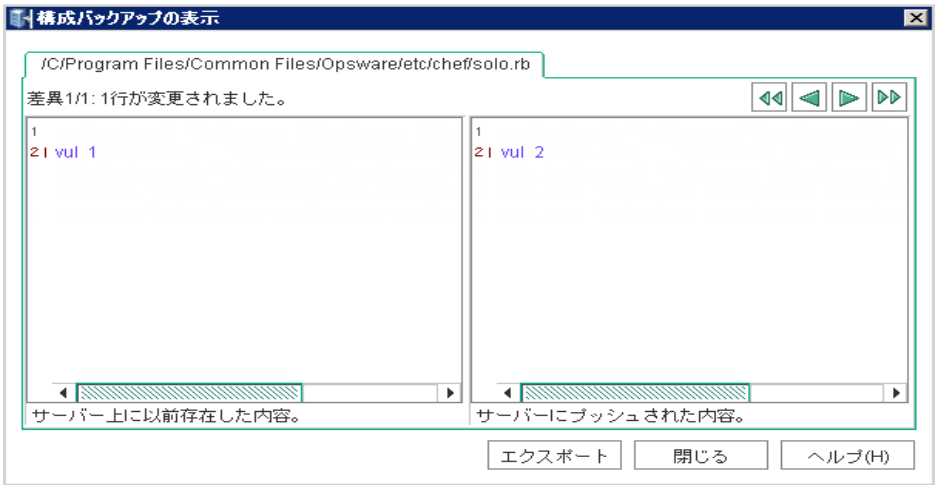


7. 履歴のインスタンスを右クリックして [**表示**] を選択すると、スナップショットが表示されます。



構成バックアップの表示ウィンドウが開きます。このウィンドウは、内容を比較できるように2つに分かれています。

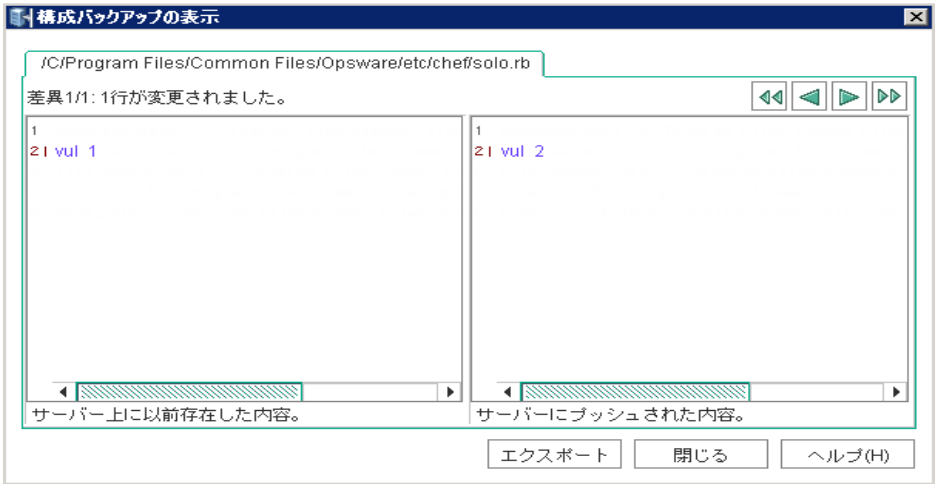
- 左のペインには、スナップショットの前にプッシュしたファイルの内容が表示されます。これは、既存ファイルのバックアップです (存在する場合)。
- 右のペインには、選択したスナップショットにプッシュしたファイルの内容が表示されます。



8. 復元したい行を選択し、[**復元**] をクリックします。
[構成の復元] ウィザードが表示されます。
9. 復元するサーバーと履歴インスタンスを確認して、[**次へ**] ボタンを選択します。
10. 復元タイプを選択します。
    - [選択したプッシュより前 (元に戻す)] を選択すると、選択した履歴インスタンスの直前の値に構成ファイルが復元されます。つまり、選択したインスタンスが元に戻されます。
    - [選択したプッシュより後 (やり直し)] を選択すると、選択した履歴インスタンスの値に構成ファイルが復元されます。つまり、選択したインスタンスが再プッシュされます。
11. [**次へ**] をクリックします。
12. [プレビュー] ステップで [**プレビュー**] をクリックすると、サーバー上にある現在のインスタンスに対する変更内容を比較できます。
構成バックアップの表示ウィンドウが開きます。このウィンドウは、内容を比較できるように2つに分か

れています。

- 左には、サーバー上にある所定での場所の現在の状態が表示されます。
- 右には、ジョブで選択した復元タイプに基づいて、その場所に復元される内容が表示されます。



このウィンドウでは、実際に復元を行う前に、実行結果を確認することができます。

- キャンセルするには、ウィンドウを閉じてからジョブウィンドウで [**キャンセル**] をクリックします。
- 復元を行うには、ウィンドウを閉じてから次のステップに進みます。

13. [**ジョブの開始**] をクリックします。サーバー上の構成が、選択した履歴インスタンスに復元されます。
14. ジョブが開始された後でそのステータスを確認するには、メインSAクライアント画面で [ジョブとセッション] タブを選択し、[ジョブログ] を選択します。

ジョブ結果の検索とフィルター処理

プッシュまたは復元ジョブの結果を検索し、フィルター処理することができます。これは、多数のサーバーに対してジョブを実行する場合に有効です。なお、検索とフィルター処理は、プッシュジョブと復元ジョブだけでなく、すべての種類のSAジョブに対して実行できます。

結果の検索とフィルター処理を行うには、次の手順を実行します。

1. SAクライアントナビゲーションペインで、[**ジョブとセッション**] タブを選択します。
2. [**ジョブログ**] を選択します。サーバー上で実行されたジョブのリストが表示されます。
3. ジョブリストからジョブを選択します。
4. [**アクション**] > [**開く**] メニューを選択します。選択したジョブの詳細が表示されます。
5. ジョブ画面で、[ジョブステータス] を選択します。ジョブの結果が表示されます。

6. [アクション] 列でどれかのステップを選択します。
7. キーボードの [Ctrl] ＋ [F] を押します。ジョブのステップを対象とする検索ツールが表示されます。検索するテキストをテキストボックスに入力し、ボタンを使用して検索と強調表示を行います。
8. 下のボックスで詳細テキストを選択します。
9. キーボードの [Ctrl] ＋ [F] を押します。ジョブ結果の詳細を対象とする検索ツールが表示されます。検索するテキストをテキストボックスに入力し、ボタンを使用して検索と強調表示を行います。
10. 検索ツールを消去するには、テキストボックスを選択して、キーボードの [Esc] キーを押します。

構成テンプレートとターゲット構成ファイルの比較 - プレビュー

アプリケーション構成をサーバーにプッシュする前に、[プレビュー] ボタンを使用して、提示されたアプリケーション構成をサーバー上のターゲット構成ファイルと比較することができます。サーバーは、管理対象サーバーの中から選択するか、デバイスグループから選択します。

管理対象サーバーからのサーバーの選択

構成テンプレートの値をサーバー上の実際の構成ファイルと比較するには、次の手順を実行します。

1. SAクライアントナビゲーションペインで、[**デバイス**] タブを選択します。
2. [**サーバー**] > [**すべての管理対象サーバー**] を選択します。
3. 内容ペインでサーバーを選択します。
4. [**アクション**] > [**開く**] メニューを選択します。
5. [**管理ポリシー**] タブを選択します。
6. [**構成されるアプリケーション**] ノードを開き、サーバーにアタッチされているすべてのアプリケーション構成を表示します。
7. 目的のアプリケーション構成ノードを開き、そのアプリケーション構成のすべてのインスタンスを表示します。
8. アプリケーション構成のインスタンスを選択します。
9. 内容ペイン、ドロップダウンリストからテンプレートを選択します。

10. [**プレビュー**] をクリックします。テンプレートと値セットから生成された構成ファイルとサーバー上の実際の構成ファイルが比較され、2つのファイルが見やすいように色分けされて並べて表示されます。
    - **緑**: 新しく追加された内容です。
    - **青**: 情報が変更されたことを示します。
    - **赤**: 削除された内容です。
    - **黒**: 変更されていない情報です。

デバイスグループからのサーバーの選択

構成テンプレートの値をデバイスグループ内のサーバー上の実際の構成ファイルと比較するには、次の手順を実行します。

1. SAクライアントナビゲーションペインで、[**デバイス**] タブを選択します。
2. [**デバイスグループ**] > [**パブリック**] を選択して、パブリックデバイスグループを表示します。
3. 目的のデバイスグループに移動し、内容ペインパブリックデバイスグループを選択します。
4. [**アクション**] > [**開く**] メニューを選択します。
5. [**構成されるアプリケーション**] ノードを開き、デバイスグループおよびデバイスグループ内のサーバーにアタッチされているすべてのアプリケーション構成を表示します。
6. [サーバー] ノードを開き、デバイスグループ内のすべてのサーバーを表示します。
7. 目的のサーバーノードを開き、そのサーバーにアタッチされているアプリケーション構成を表示します。
8. 目的のアプリケーション構成ノードを開き、アプリケーション構成のすべてのインスタンスを表示します。
9. アプリケーション構成のインスタンスを選択します。
10. 内容ペイン、ドロップダウンリストからテンプレートを選択します。
11. [**プレビュー**] をクリックします。テンプレートと値セットから生成された構成ファイルとサーバー上の実際の構成ファイルが比較され、2つのファイルが見やすいように色分けされて並べて表示されます。
    - **緑**: 新しく追加された内容です。
    - **青**: 情報が変更されたことを示します。
    - **赤**: 削除された内容です。
    - **黒**: 変更されていない情報です。

構成テンプレートの比較

2つの構成テンプレートを比較するには、次の手順を実行します。

1. SAクライアントナビゲーションペインで、[ライブラリ] を選択し、[タイプ別] タブを選択します。
2. [アプリケーション構成] ノードを見つけて開きます。[テンプレート] ノードを開きます。オペレーティングシステムグループを開き、テンプレートファイルが存在するオペレーティングシステムに移動します。テンプレートは複数のオペレーティングシステムに適用することもできます。
3. 構成テンプレートを選択します。
4. キーボードのCtrlキーを押しながら、もう1つのテンプレートを選択します。
5. 右クリックして [比較] メニューを選択します。[比較] 画面に2つのファイルの違いが表示されます。画面右上の矢印を使用すると、2つのファイル内を移動できます。差異は次の色で表されます。
   - **青**: 2つのテンプレートの間で異なる情報です。
   - **赤**: 削除された情報です。
   - **緑**: 追加された情報です。
   - **黒**: は、同一のテキストです。
6. 差異の確認が済んだら、[**閉じる**] を選択します。

構成ファイルの要素名のローカライズ

SAクライアントの値セットエディターで、構成ファイル要素の名前をローカル言語で表示できます。これは、値セットを指定するシステム管理者が別の言語を使用する場合に役立ちます。

ローカリゼーションファイルは、構成テンプレート要素に対応するローカライズされた文字列が定義されるロケール固有のリソースファイルを表します。ローカライズされた文字列は、値セットエディターに表示されます。

ローカリゼーションテンプレートは、SAクライアントの値セットエディターのみで用いられ、プッシュの際には無視されます。

ローカリゼーションファイルの作成

SAクライアントで構成ファイル要素の名前をローカライズするには、最初にSAライブラリでローカリゼーションテンプレートを作成する必要があります。

ローカリゼーションファイルを作成するには、次の手順を実行します。

1. SAクライアントナビゲーションペインで、[ライブラリ] を選択し、[タイプ別] タブを選択します。
2. [アプリケーション構成] ノードを見つけて開きます。[テンプレート] ノードを開きます。オペレーティングシステムグループを開き、テンプレートファイルが使用されるオペレーティングシステムを選択します。テンプレートは複数のオペレーティングシステムに適用することもできます。これは後の手順で指定します。
3. [**アクション**] > [**新規**] メニューを選択します。[テンプレート] 画面が表示され、テンプレートのプロパティを定義できます。
4. [プロパティ] ビューを選択し、ローカリゼーションテンプレートファイルの説明と、次の情報を入力します。
   - **名前**: ローカリゼーションファイルの名前を指定します。ローカリゼーションテンプレートファイルの命名規則では、ローカリゼーションファイル名の最後は.<ロケール>であることが要求されています。<ロケール>の値はISO-639で定義されており、言語名を表す2文字の英小文字コードです。ISO 639の詳細については、http://www.loc.gov/standards/iso639-2/php/code_list.php (英語サイト) を参照してください。
   - たとえば、.esはスペイン語、.enは英語、.frはフランス語、.zhは中国語、.hiはヒンディー語を表します。
   - **場所**: **SA**ライブラリのどこにローカリゼーションテンプレートを保存するかを指定します。
   - **バージョン**: バージョンは任意の文字列で、テンプレートへの変更を追跡するために使用します。バージョンは自動的には増加しません。
   - **タイプ**: タイプとして [ローカリゼーションファイル] を指定します。
   - **可用性**: この設定は、テスト済みで使用可能なローカリゼーションファイルと、まだテストされていないか、非推奨になったローカリゼーションファイルを区別するために使用します。ローカリゼーションファイルに対して実行可能な操作は、この設定によって変わりません。このフィールドの値は検索基準として使用できます。
5. [内容] ビューを選択します。

6. ローカリゼーション命令をテンプレートエディターに直接入力します。
   - 表示テキストローカリゼーション命令の形式は次のとおりです。

     `printable/<名前空間>/<変数> <ローカライズされた文字列>`

     ここで、`printable`は、この行が表示テキストのローカリゼーション命令であることを示すキーワードです。

     <名前空間>は、目的の変数がデータベースに格納される名前空間です。

     <変数>は、構成テンプレートで定義されている変数です。

     <ローカライズされた文字列>は、名前空間と変数名全体の代わりに値セットエディターに表示されるテキストです。

   - ツールヒントローカリゼーション命令の形式は次のとおりです。

     `description/<名前空間>/<変数> <ローカライズされた文字列>`

     ここで、`description`は、この行がツールヒントテキストのローカリゼーション命令であることを示すキーワードです。

     <名前空間>と<変数>は上と同じです。

     <ローカライズされた文字列>は、値セットエディターで項目の上にマウスポインターを置いたときに表示されるテキストです。

   編集操作と構文の強調表示については、「テンプレート内のCMLまたはXMLの編集」(279ページ)を参照してください。

7. [**検証**]を選択して、構文を解析し、エラーをチェックします。
8. [**ファイル**] > [**保存**]を選択してテンプレートを保存します。
9. 「ローカリゼーションテンプレートの適用」(300ページ)の手順に従って、ローカリゼーションテンプレートをアプリケーション構成オブジェクトに追加します。

ローカリゼーションテンプレートの適用

ローカリゼーションテンプレートを作成したら、アプリケーション構成に適用して、構成ファイル要素がローカル言語で表示されるようにします。

ローカリゼーションテンプレートをアプリケーション構成に適用するには、次の手順を実行します。

1. アプリケーション構成オブジェクトを次のように開きます。
   a. SAクライアントナビゲーションペインで、[ライブラリ] を選択し、[タイプ別] タブを選択します。
   b. [アプリケーション構成] ノードを見つけて開きます。[構成] ノードを開きます。オペレーティングシステムグループを開き、アプリケーション構成が存在するオペレーティングシステムに移動します。アプリケーション構成は複数のオペレーティングシステムに適用することもできます。
   c. アプリケーション構成を選択し、[**アクション**] > [**開く**] を選択します。
2. [プロパティ] ビューを選択します。
3. [内容] ペインの [ローカリゼーションファイル] の下で、[+] ボタンを選択します。
4. [ローカリゼーションテンプレートの選択] 画面で、ローカリゼーションテンプレートを選択します。
5. [**OK**] を選択します。
6. [**ファイル**] > [**保存**] を選択します。これによりローカリゼーションテンプレートが適用され、構成ファイルの要素が、ローカリゼーションテンプレートで指定された言語で表示されます。このアプリケーション構成に対して値セットエディターを使用すると、元の名前空間および変数名の代わりに、ローカライズされた文字列が表示されます。

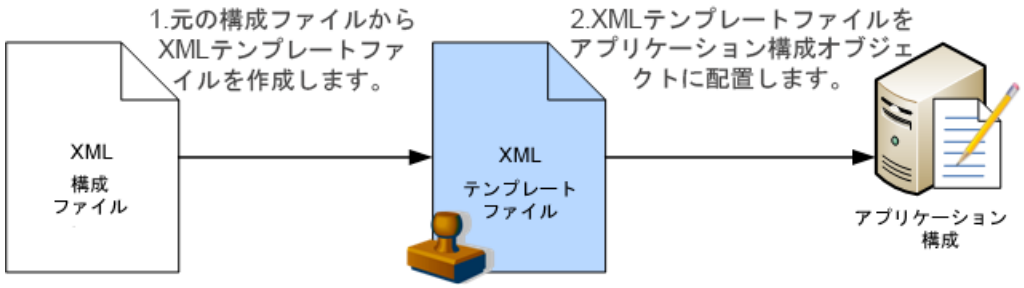
XML構成ファイルの管理

SAでは、XML構成ファイルを1か所でまとめて管理し、データセンターで複数のサーバー間の変更を反映することができます。管理対象サーバー上のXML構成ファイルが正しくなるように、構成ファイルの値の作成、編集、記録を行うことができます。DTDを使用するXMLファイルだけでなく、DTDを使用しないものも管理できます。

この項では、XML構成テンプレートの構造と、一般的な (DTDを使用しない) XMLファイルおよびDTDを参照するXMLファイルの管理方法を説明します。XMLは構造が明確なので、SAは最小限の情報だけでXMLベースの構成ファイルをモデル化し、管理することができます。

XML構成ファイルを管理するには、まずXML構成ファイル用の**テンプレート**ファイルを作成する必要があります。テンプレートを作成したら、**アプリケーション構成**オブジェクトに追加して、管理対象サーバー上のネイティブ構成ファイルの管理、編集、変更を可能にします。

構成ファイル



次の項では、単純なXMLファイルを示し、非DTDベースのXMLファイル用のアプリケーション構成と、DTDベースのXMLファイル用のアプリケーション構成を作成する方法を説明します。

次の例も参照してください。

- 「XMLチュートリアル1 - 非DTD XML構成テンプレートの作成」(312ページ)
- 「XMLチュートリアル2 - XML-DTD構成テンプレートの作成」(319ページ)

例: Travel ManagerアプリケーションとXML構成ファイル

この項では、単純なXMLファイルによって構成を制御するWebアプリケーションの例を示し、このファイルを管理するためのアプリケーション構成を作成する方法を説明します。

Travel Managerは、ユーザーの旅行を支援するWebアプリケーションで、ホテルの予約、レンタカーの手配、出費の記録といった機能を備えています。Travel Managerは、MySQLリレーショナルデータベース管理システム(RDMS)を、ユーザーデータとアプリケーションの一部の構成データのリポジトリとして使用します。

Travel Managerは、それぞれ異なるデータベースサーバーを持つさまざまなネットワーク上で動作するように設計されているため、MySQLサーバーへの接続に使用する情報の柔軟性が重要です。アプリケーションは、接続情報を`mysql.xml`というXML構成ファイルから取得します。

アプリケーション構成により、ローカルMySQLデータベースにアクセスするために必要な構成ファイルの値を設定できます。たとえば、データベースへの接続を開くために使用するユーザー名とパスワードは、Travel Managerアプリケーションのインストールごとに異なる可能性があります。これらの値は構成ファイルで変更することができ、Travel Managerアプリケーションのコードを再コンパイルする必要はありません。

Travel ManagerがローカルMySQLデータベースに接続するために必要なのは、`mysql.xml`内の4つの値だけであり、これらはそれぞれアプリケーションのXMLファイル内の要素として表現されています。

- **ホスト**: MySQL RDMSがインストールされているサーバーのホスト名。
- **名前**: ホストサーバー上のデータベースの名前。
- **ユーザー**: データベースへの接続を開くために使用するユーザー名資格情報。
- **パスワード**: データベースへの接続を開くために必要なパスワード。

Travel Managerのmysql.xmlファイルの内容

次に示すのは、Travel Managerの`mysql.xml`構成ファイルの例です。

```
<?xml version="1.0" ?>
<db-config>
  <db-host>localhost</db-host>
  <db-name>wrightevents</db-name>
  <db-user>root</db-user>
  <db-password>hp-pass</db-password>
</db-config>
```

Travel Managerのmysql.xml DTDベースXMLファイルの内容

次に示すのは、Travel ManagerのDTDを参照する`mysql.xml`構成ファイルの例です。

```
<?xml version="1.0"?>
<!DOCTYPE db-config PUBLIC "-//Williams Events//Travel Manager//EN" "mysql2.dtd">
<db-config>
       <db-host>localhost</db-host>
       <db-name>wrightevents</db-name>
       <db-user>root</db-user>
       <db-password>hp-pass</db-password>
</db-config>
```

非DTD XML構成テンプレート

非DTDベースのXML構成テンプレートを作成するには、ターゲットXMLファイルから値を抽出して記録するために必要な次の3つの情報を、1つのXMLコメントで指定します。

- ACM-NAMESPACE: 管理対象サーバー上のターゲットXMLファイルから読み取った値がデータベースで記録される場所を定義します。名前空間は一意で、パスの先頭はスラッシュ (/) である必要があります。
- ACM-FILENAME-DEFAULT: 管理対象サーバー上のターゲットXML構成ファイルのデフォルトの絶対パスを定義します。
- ACM-FILENAME-KEY: ターゲットXML構成ファイル名が記録される名前空間内の場所を定義します。

構成テンプレートのプロパティをXMLを使用するように設定した場合、値セットエディターに表示されるラベルは、XMLファイル内部の対応する各要素のタグ名に一致します。

XMLテンプレートのテンプレート設定の一覧については、「XML構成テンプレートの設定」(310ページ)を参照してください。

構成テンプレートでパーサー構文をXMLに設定する方法については、「構成テンプレートの作成」(262ページ)を参照してください。

mysql.xmlに対する非DTD XML構成テンプレート

次の例は、mysql.xmlファイルに基づくXML構成テンプレートを示します。ファイル名はmysql.tplで、テンプレートファイルであることを表しています。

```
<!--
ACM-NAMESPACE = /TravelManager/
ACM-FILENAME-KEY = /files/TravelManager
ACM-FILENAME-DEFAULT = /var/www/html/we/mysql.xml
ACM-TIMEOUT = 1
-->
```

この例は、XML構成テンプレートがターゲットXMLファイル (/var/www/html/we/mysql.xml) を参照することによって、アプリケーション構成パーサーによるファイルの解析と、その値の読み取りとSAライブラリへの記録を可能にしていることを示しています。

mysql.tpl構成テンプレートには、次の必須情報が含まれています。

- ACM-NAMESPACE: 管理対象サーバー上のmysql.xmlファイルから読み取った値がデータベースで記録される場所を定義します。名前空間は一意で、パスの先頭はスラッシュ (/) である必要があります。
- ACM-FILENAME-DEFAULT: 管理対象サーバー上のmysql.xmlファイルのデフォルトの絶対パスを定義します。
- ACM-FILENAME-KEY: mysql.xmlファイル名が記録される名前空間内の場所を定義します。
- ACM-TIMEOUT: (オプション) プッシュ時に構成テンプレートのデフォルトのタイムアウト値 (10分) に加算される時間 (分) を表します。

アプリケーション構成全体のデフォルトのタイムアウト値は、10分に、アプリケーション構成内部のすべての構成テンプレートのタイムアウトを加算したものです。したがって、このテンプレートがアプリケーション構成内部のただ1つのテンプレートであり、この値が1に設定されている場合は、プッシュ時のアプリケーション構成全体のタイムアウト値は11分になります。

DTDベースのXML構成テンプレート

XML-DTD構成テンプレートは、実際には単にコメントにアプリケーション構成オプションが定義されたXML DTDです。DTD標準ではXMLファイルの構文とレイアウトが定義されているので、この構文を別の言語で再定義する必要はありません。

DTDベースのXMLファイルの場合、XML-DTD構成テンプレートには、一般的なXMLファイルの場合の3つの必須基本属性、すなわちACM-NAMESPACE、ACM-FILENAME-DEFAULT、ACM-FILENAME-KEYに加えて、次の3つの属性が必要です。

- ACM-DOCTYPE: XMLファイル内のルート要素の名前を定義します。ルート要素は、ターゲットXML構成ファイルの開始<!DOCTYPE宣言の後にあります。
- ACM-DOCTYPE-SYSTEM-ID: 管理対象サーバー上の対応するDTDファイルの名前を定義します。この値は通常、XML構成ファイル内のDOCTYPE要素のSYSTEM属性に記述されています。
- ACM-DOCTYPE-PUBLIC-ID: XMLドキュメントのパブリックIDを表す文字列を定義します。この値は通常、XML構成ファイル内のDOCTYPE要素のPUBLICID属性に記述されています。

XML構成ファイルの属性の一覧については、「XML構成テンプレートの設定」(310ページ)を参照してください。

mysql.xmlに対するXML-DTD構成テンプレート

次に示すのは、Travel ManagerのDTDベースXMLファイル用に作成された構成テンプレートの例です。

```
<!--
    ACM-FILENAME-KEY = /files/TravelManager
    ACM-FILENAME-DEFAULT = /var/www/html/we/mysql.xml
    ACM-NAMESPACE = /TravelManager/
    ACM-TIMEOUT = 1
    ACM-DOCTYPE = db-config
    ACM-DOCTYPE-SYSTEM-ID = mysql.dtd
    ACM-DOCTYPE-PUBLIC-ID = -//Williams Events//Travel Manager//EN
    -->
    <!ELEMENT db-config (db-host,db-name,db-user,db-password)>
    <!ELEMENT db-host (#PCDATA)>
    <!ELEMENT db-name (#PCDATA)>
    <!ELEMENT db-user (#PCDATA)>
    <!ELEMENT db-password (#PCDATA)>
```

この例で、DOCTYPE属性は、DTDファイルと参照されるXMLファイルの両方からパーサーが情報を抽出するために必要な特定のXMLおよびDTD情報を参照します。

具体的には、DTDベースのXML構成テンプレートには次の情報が必要です。

- ACM-DOCTYPE: ターゲットXMLファイルのルートノード。mysql.xmlの場合、ルートノードはdbconfigです。
- ACM-DOCTYPE-SYSTEM-ID: 構成テンプレートのターゲットとなるDTDファイルの名前。mysql.xmlの場合、使用されるDTDの名前はmysql.dtdです。
- ACM-DOCTYPE-SYSTEM-ID: XMLファイルのパブリックID。

XML DTD要素表示のカスタマイズ

XML-DTD構成テンプレートに2つのオプションの設定を追加することで、ターゲットXML-DTD構成ファイルの要素がSAクライアントの値セットエディターに表示される方法をカスタマイズすることができます。ACM-PRINTABLEおよびACM-DESCRIPTIONオプション設定を使用すると、SAクライアントに表示される要素の名前を制御できます。

- ACM-PRINTABLE: XML-DTDテンプレートをSAクライアントに表示したときに、値セットエディターに表示されるXMLファイルの各要素のラベルを定義します。
- ACM-DESCRIPTION: SAクライアントの値セットエディターで、ACM-PRINTABLEに定義されたフィールドにマウスポインターを移動したときに表示されるテキストを定義します。

明示的表示設定と位置による表示設定

XML-DTD構成テンプレート内部の属性と要素のACM-PRINTABLEとACM-DESCRIPTIONの値を設定するには、位置による方法と明示的な方法の2つがあります。

- 位置による定義では、ACM-PRINTABLEおよびACM-DESCRIPTIONを、XML-DTD構成テンプレートで対象となる要素または属性の直後に挿入します。
- 明示的な定義では、ACM-PRINTABLEとACM-DESCRIPTIONはテンプレートの任意の場所で定義できます。

```
<!ELEMENT db-config (db-host,db-name,db-user,db-password)>
<!--
ACM-PRINTABLE = database configuration
ACM-DESCRIPTION = The db-config element specifies the data structure that contains
the information needed to connect to a database.
-->

<!ELEMENT db-host (#PCDATA)>
<!--
ACM-PRINTABLE = database hostname
ACM-DESCRIPTION = The db-host element specifies the name of the host computer (the
server) on which the database engine is running.
-->

<!ELEMENT db-name (#PCDATA)>
<!--
ACM-PRINTABLE = database name
ACM-DESCRIPTION = The db-name element specifies the name of the database.
-->

<!ELEMENT db-user (#PCDATA)>
<!--
ACM-PRINTABLE = database user
ACM-DESCRIPTION = The db-user element specifies the user identification used to
connect to the database.
-->

<!ELEMENT db-password (#PCDATA)>
<!--
ACM-PRINTABLE = database password
ACM-DESCRIPTION = The db-password element specifies the password used to connect to
the database.
-->
```

位置によるカスタム表示設定の追加

XMLテンプレートに要素テーブルとマウスオーバーテキストを位置によって追加するには、対象の要素または属性定義の直後にコメントを追加して、その中でACM-PRINTABLEとACM-DESCRIPTIONの値を設定します。言い換えれば、XML要素または属性に対して、ラベルとラベルにマウスポインターを置いたときに表示されるテキストを直接指定できます。

次の例では、mysql.xmlの各XML要素に対して、XML-DTDテンプレート内の各要素の直後にACM-PRINTABLEとACM-DESCRIPTIONの設定が定義されています。

明示的なカスタム表示設定の追加

XML-DTDテンプレートに明示的に設定を追加する方法では、ACM-PRINTABLEとACM-DESCRIPTIONの値を構成テンプレートの任意の場所で定義できます。このためには、ACM-ELEMENTタグによって要素名を指定し、オプションでACM-ATTRIBUTEタグによって属性名を指定します。

この方法では、属性に対してACM-PRINTABLEとACM-DESCRIPTIONを定義する場合でも、ACM-ELEMENTタグは必須です。属性は常に特定の要素と関連付けられるからです。

ACM-ELEMENTタグとACM-ATTRIBUTEタグを設定したら、同じコメントブロックの中でACM-DESCRIPTIONタグとACM-PRINTABLEタグも設定できます。1つのコメントブロックには1つの定義だけを置きます。すなわち、1つの要素に対してACM-PRINTABLEとACM-DESCRIPTIONを定義したら、次の要素に対しては新しいコメントブロックを開始します。

ACM-ELEMENTタグとACM-ATTRIBUTEタグ(該当する場合)は、ACM-PRINTABLEタグおよびACM-DESCRIPTIONタグの前で定義する必要があります。

たとえば、mysql.tplテンプレートをカスタマイズするには、次のようなテンプレートを作成します。

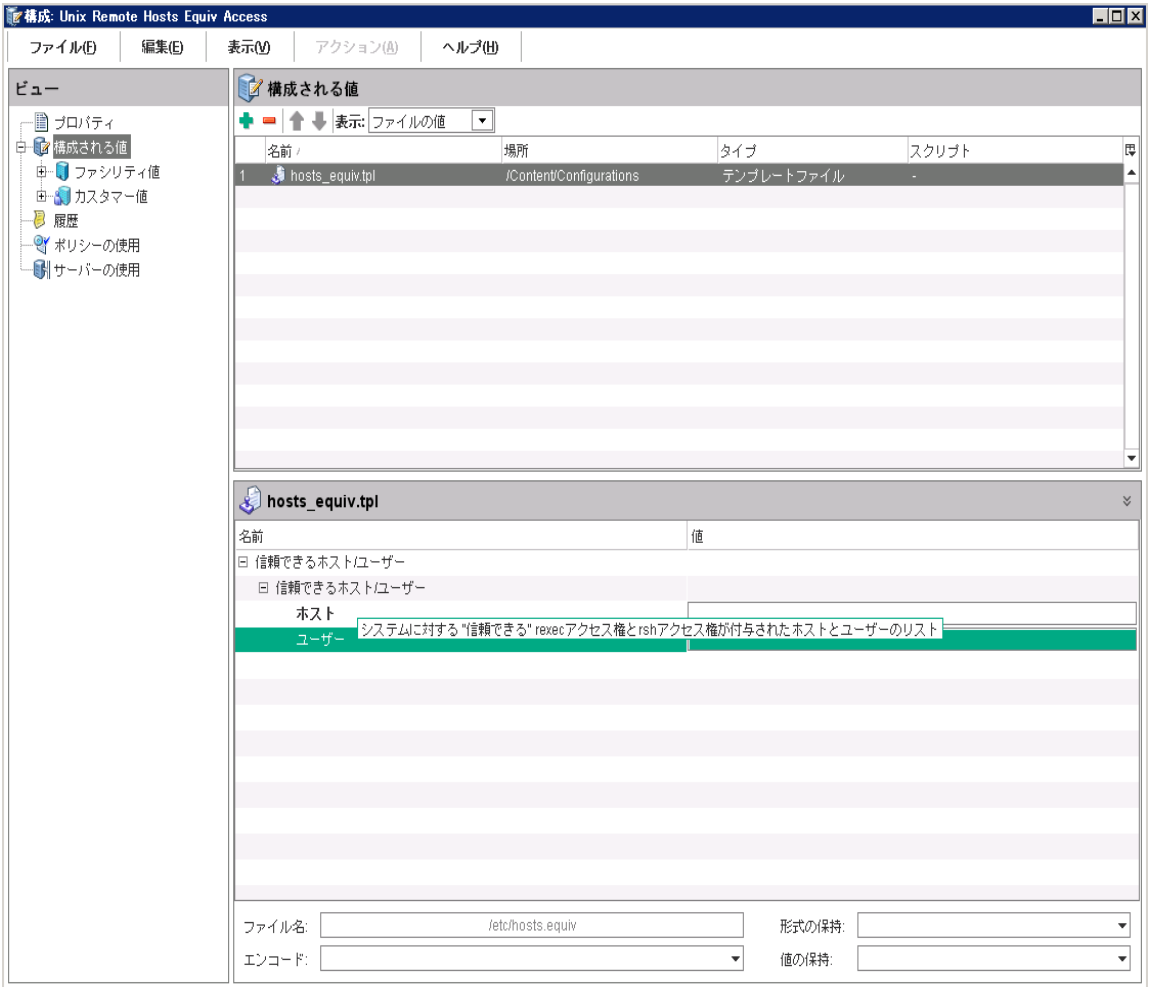
```
<!--
ACM-TIMEOUT = 1
ACM-FILENAME-KEY = /files/TravelManager
ACM-FILENAME-DEFAULT = /var/www/html/we/mysql2.xml
ACM-NAMESPACE = /TravelManager/
ACM-DOCTYPE = db-config
ACM-DOCTYPE-SYSTEM-ID = mysql.dtd
ACM-DOCTYPE-PUBLIC-ID = -//Williams Events//Travel Manager//EN
-->

<!ELEMENT db-config (db-host,db-name,db-user,db-password)>
<!ELEMENT db-host     (#PCDATA)>
<!ELEMENT db-name     (#PCDATA)>
<!ELEMENT db-user     (#PCDATA)>
<!ELEMENT db-password (#PCDATA)>
```

```
<!--
ACM-ELEMENT = db-config
ACM-PRINTABLE = database configuration
ACM-DESCRIPTION = The db-config element specifies the data structure that contains
the information needed to connect to a database.
-->

<!--
ACM-ELEMENT = db-host
ACM-PRINTABLE = database hostname
ACM-DESCRIPTION = The db-host element specifies the name of the host computer (the
server) on which the database engine is running.
-->

<!--
ACM-ELEMENT = db-name
ACM-PRINTABLE = database name
ACM-DESCRIPTION = The db-name element specifies the name of the database.
-->

<!--
ACM-ELEMENT = db-user
ACM-PRINTABLE = database user
ACM-DESCRIPTION = The db-user element specifies the user identification used to
connect to the database.
-->

<!--
ACM-ELEMENT = db-password
ACM-PRINTABLE = database password
ACM-DESCRIPTION = The db-password element specifies the password used to connect to
the database.
-->
```

SAクライアントでの要素の表示方法のカスタマイズ

これらの属性を位置によって追加しても明示的に追加しても、どちらの方法でも結果は同じです。SAクライアントの値セットエディターには、次の図に示すように、要素名 (ACM-PRINTABLEで定義) とマウスオーバーテキスト (ACM-DESCRIPTIONで定義) が表示されます。

カスタム要素名とマウスオーバーテキスト



XML構成テンプレートの設定

次の表に、一般あるいはDTDベースのXML構成テンプレートを作成する際に使用できるXML設定の一覧を示します。このリストは、設定が必須かオプションかと、XML-DTDテンプレートだけに適用されるかどうかを示します。

XMLおよびXML-DTDテンプレート設定

属性	説明
ACM-FILENAME-KEY=<キー> 必須。デフォルト値なし。	`filename-key`は、生成するファイルの名前を含む値セットのキーのパスを指定します。
ACM-FILENAME-DEFAULT=<ファイル名>	`filename-default`は、値セットにファイル名がない場合に返されるデフォルトのファイル名を指定します。

XMLおよびXML-DTDテンプレート設定 (続き)

属性	説明
必須。デフォルト値なし。	
ACM-NAMESPACE=<文字列> 必須。デフォルト値なし。	namespaceは、XML要素がデータベースに記録される場所を指定します。
ACM-TIMEOUT=<整数> オプション (デフォルト値は0)。	timeoutは、アプリケーション構成の合計タイムアウトに加算する時間を分単位で指定します。 有効なタイムアウトは、0～999 (両端含む) の整数です。 アプリケーション構成内のすべての構成テンプレートのタイムアウトの合計に、構成のデフォルトのタイムアウトである10分を加算したものが、構成全体の最終的なタイムアウト値になります。 アプリケーション構成のインストール前または後のスクリプトの実行時間が10分を超えると、タイムアウトが発生し、プッシュジョブ全体がキャンセルされることに注意してください。
ACM-DOCTYPE = <文字列> 必須。デフォルト値なし。 XML-DTDテンプレートのみ。	doctypeは、XMLファイル内のルート要素の名前を表します。これはXMLファイルの先頭にあるDOCTYPEタグの中にあります。
ACM-DOCTYPE-SYSTEM-ID = <文字列> 必須。デフォルト値なし。 XML-DTDテンプレートのみ。	system-idは、構成テンプレートの元になるDTDファイルのシステムIDを表します。この値は、XMLファイルの先頭にあるDOCTYPEタグの中にあります。
ACM-DOCTYPE-PUBLIC-ID = <文字列> 必須。デフォルト値なし。 XML-DTDテンプレートのみ。	public-idは、構成テンプレートで解析されるXMLファイルのパブリックIDを表します。この値は、XMLファイルのDTDオプションの先頭にあるDOCTYPEタグの中にあります。
ACM-ELEMENT=<要素名> オプション XML-DTDテンプレートのみ。	elementは、現在のオプションが記述する要素を設定します。このオプションのデフォルトは、DTDファイル内でこのセクションの前にある要素または属性です。
ACM-ATTRIBUTE=<属性名> オプション XML-DTDテンプレートのみ。	attributeは、現在のオプションが記述する属性を設定します。属性が設定されていない場合、このオプションは無視されます。この属性のデフォルトは、ファイル内でこのセクションの前にある要素または属性です。
ACM-PRINTABLE=<ラベル>	printableは、SAクライアントでの要素または属性のラベルの

XMLおよびXML-DTDテンプレート設定 (続き)

属性	説明
オプション XML-DTDテンプレートのみ。	値を設定します。この値は、値セットエディターでフィールドの左に表示されます。これは通常、短くてわかりやすいテキストに設定します。
ACM-DESCRIPTION=<説明> オプション XML-DTDテンプレートのみ。	`description`は、SAクライアントに表示される現在の要素または属性の説明を設定します。この値は、値セットエディターで名前または値フィールドの上にマウスを置くと表示されます。これは、値セットエディターで、フィールドの目的や有効な値を示すために使用します。

XMLチュートリアル1 - 非DTD XML構成テンプレートの作成

このチュートリアルでは、非DTD XML構成ファイルから構成テンプレートを作成する方法を示します。ここでは、XML構文を使用して構成テンプレートを作成し、アプリケーション構成に追加して、アプリケーション構成を管理対象サーバーにアタッチする方法を示します。その後、管理対象サーバー上の`mysql.xml`構成ファイルから値をインポートし、値の一部を変更し、新しい構成ファイルを管理対象サーバーにプッシュします。

このチュートリアルは、「例: Travel ManagerアプリケーションとXML構成ファイル」(302ページ)で説明したTravel Managerの例に基づいています。

非DTD XMLのmysql.xmlファイルの例

次に示すのは、Travel Managerアプリケーション用のXML構成ファイルの内容です。

```
<?xml version="1.0" ?>
<db-config>
  <db-host>localhost</db-host>
  <db-name>wrightevents</db-name>
  <db-user>root</db-user>
  <db-password>hp-pass</db-password>
</db-config>
```

1. XML構成テンプレートの作成

SAクライアントを使用して、mysql.xml構成ファイルに基づいて構成テンプレートを作成します。

1. SAクライアントナビゲーションペインで、[ライブラリ] を選択し、[タイプ別] タブを選択します。
2. [アプリケーション構成] ノードを開き、[テンプレート] ノードを開きます。すべてのオペレーティングシステムグループが表示されます。
3. オペレーティングシステムノードを開き、1つのオペレーティングシステムノードの下で特定のオペレーティングシステムを選択します。この例では、このアプリケーション構成をインストールできる1つのサーバーのオペレーティングシステムを選択します。
4. [**アクション**] メニューから [**新規**] を選択します。
5. プロパティビューに次の情報を入力します。
   - **名前**: TM-MySql
   - **説明**: This is the template for the mysql.xml configuration file for the Travel Manager application.
   - **場所**: SAライブラリのデフォルトの場所である/をそのまま使用するか、テンプレートファイルを保存する別の場所を選択します。テンプレートを含むフォルダーのカスタマー設定には、アプリケーション構成オブジェクトのカスタマー設定が含まれる必要があります。そうでないと、テンプレートは利用可能なテンプレートのリストに含められません。フォルダー設定の詳細については、『SA 10.50管理ガイド』の「フォルダーのアクセス権」の項を参照してください。
   - **バージョン**: 0.1.
   - **タイプ**: テンプレートファイル
   - **パーサー構文**: XML構文
   - **OS**: 構成テンプレートがインストールできるすべてのオペレーティングシステムを選択します。
   - **[ファイル] > [保存]** を選択します。
6. 次の作業のために [テンプレート] ウィンドウを開いたままにしておきます。

2. XML設定の追加

XML構成ファイルmysql.xmlにはファイルの内容を解析するための構造設定の大部分が含まれているため、SAのXML構成テンプレートに必要なのは、XMLコメント内のACM-NAMESPACE、ACM-FILENAME-KEY、ACM-FILENAME-DEFAULTの3つの情報だけです。

1. ナビゲーションペインで [内容] ビューを選択します。

```
<!--
ACM-NAMESPACE = /TravelManager
ACM-FILENAME-KEY = /files/TravelManager
ACM-FILENAME-DEFAULT = /var/www/html/we/mysql.xml
-->
```

2. 次のXMLをコピーして内容ペインに貼り付けます。
3. [検証] ボタンを選択して、XMLが有効であることを確認します。
4. **[ファイル] > [保存]** メニューを選択してテンプレートを保存します。
5. **[ファイル] > [閉じる]** メニューを選択します。

これらのXML行は、次の内容を定義します。

- `ACM-NAMESPACE`: 各構成テンプレートに必要な固有の名前空間を指定します。この例では、Travel Managerアプリケーションの名前空間はすでに確立されているので、ルート名前空間を再使用して、サービス名を追加することができます。次に例を示します。
- `ACM-NAMESPACE = /TravelManager/web/mysql`
- `ACM-FILENAME-KEY`: 生成するファイルのファイル名を記録する名前空間内のキーのパスを指定します。
- `ACM-FILENAME-DEFAULT`: Travel Mangerアプリケーションの`mysql.xml`ファイルが保存されるターゲットサーバー上のパスを指定します。これは特定のサーバーまたはサーバーグループに対してオーバーライドできます。

3. テンプレートを含むアプリケーション構成の作成

このステップでは、構成テンプレートを含むアプリケーション構成オブジェクトを作成します。

1. SAクライアントのナビゲーションペインで、[ライブラリ] を選択し、[タイプ別] タブを選択します。
2. [アプリケーション構成] ノードを開き、[構成] ノードを開きます。すべてのオペレーティングシステムグループが表示されます。
3. オペレーティングシステムノードを開き、前のステップでテンプレートを作成したときに使用したのと同じオペレーティングシステムを選択します。アプリケーション構成に対して指定するOSは、テンプレートに対して指定したOSのサブセットである必要があります。

4. [**アクション**] メニューから [**新規**] を選択します。
5. ファイルの構成画面のプロパティビューで、次のプロパティを指定します。
   - **名前**: Tm-MySql-Config
   - **説明**: This is the application configuration for the mySQL configuration file for the Travel Manager application.
   - **場所**: SAライブラリのデフォルトのフォルダー場所である/をそのまま使用するか、アプリケーション構成を保存する別のフォルダーを選択します。アプリケーション構成を含むフォルダーのカスタマー設定には、アプリケーション構成をプッシュする管理対象サーバーのカスタマー設定が含まれる必要があります。フォルダー設定の詳細については、『SA 10.50管理ガイド』の「フォルダーのアクセス権」の項を参照してください。
   - **バージョン**: 0.1.
   - **OS**: アプリケーション構成をインストールできる管理対象サーバーのオペレーティングシステムを1つ以上選択します。
6. [**構成される値**] を選択します。
7. 追加ボタン [+] または **[アクション] > [追加]** メニューを選択して、テンプレートを追加します。
8. [構成テンプレートの選択] 画面で、TM-MySqlテンプレートを選択し、[**OK**] を選択します。これにより、テンプレートがアプリケーション構成オブジェクトに追加されます。
9. **[ファイル] > [保存]** を選択し、**[ファイル] > [閉じる]** を選択します。これで、アプリケーション構成とその中の構成テンプレートを、構成ファイルが保存されるサーバーにアタッチする準備ができました。

4. 管理対象サーバーへのアプリケーション構成のアタッチ

構成テンプレートとアプリケーション構成オブジェクトを作成したら、Travel Managerアプリケーションがインストールされているサーバーにアプリケーション構成をアタッチし、管理対象サーバー上で`mysql.xml`ファイルが保存される場所のパスを指定する必要があります。

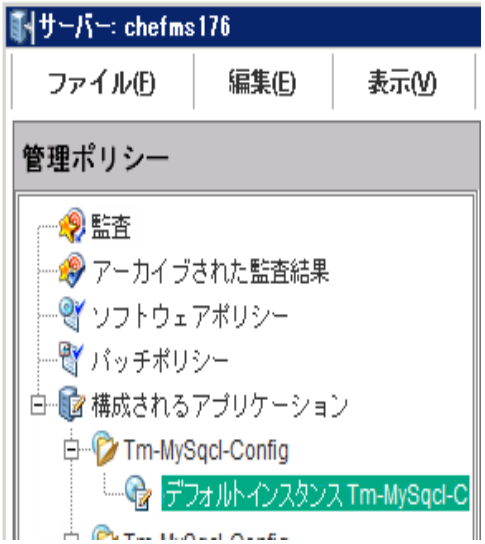
アプリケーション構成をサーバーにアタッチするには、次の手順を実行します。

1. SAクライアントのナビゲーションペインで、[デバイス] を選択し、[サーバー] > [すべての管理対象サーバー] を選択します。

2. Travel Managerアプリケーション構成のインストールをシミュレートできるサーバーを見つけます。このサーバーのオペレーティングシステムは、アプリケーション構成に指定されているオペレーティングシステムの1つに一致する必要があります。
3. サーバーを選択し、[**アクション**] メニューで、[**開く**] を選択します。
4. サーバー画面で、[管理ポリシー] タブを選択します。
5. ナビゲーションペインで [構成されるアプリケーション] を選択します。そのサーバーにアタッチされているアプリケーション構成が表示されます。
6. [インストール済み構成] タブを選択します。
7. [**アクション**] メニューで [**構成の追加**] を選択します。
8. [アプリケーション構成の選択] 画面で、Tm-MySql-Configアプリケーション構成を選択します。
9. [インスタンス名] フィールドに、「Default mysql config values」と入力します。これにより、サーバーインスタンスレベルの値セットが作成されます。詳細については、「サーバーレベルの値セットエディター」(245ページ)を参照してください。
10. [**OK**] を選択します。アプリケーション構成がサーバーにアタッチされます。
11. [変更の保存] ボタンを選択します。次のステップのためにサーバー画面を開いておきます。

    次の図は、サーバーにアタッチされたTm-MySql-Configアプリケーション構成と、“Default mysql config values” 値セットを示します。この値セットはサーバーインスタンスレベルです。

サーバーにアタッチされたアプリケーション構成とサーバーインスタンスレベルの値セット (強調表示)



> **注:**
> この時点で、アプリケーション構成を追加するサーバーがアプリケーションの複数のインスタンスをホストしているために、`mysql.xml`構成ファイルのインスタンスがサーバーに複数存在する場合、構成の“Default mysql config values”ノードを右クリックして、[**複製**]を選択できます。これにより別の値セットが作成され、そのファイル名パスをアプリケーションの別のインスタンスを指すように設定できます。値セットのさまざまなレベルの詳細については、「値セットのレベルと値セットの継承」(233ページ)を参照してください。

5. サーバーに対するアプリケーション構成設定の構成

アプリケーション構成を管理対象サーバーにアタッチしたら、サーバーに対して構成し、構成ファイルの値を設定する必要があります。

以下で説明するように構成ファイルから値をインポートするには、非DTD XMLのmysql.xmlファイルの例に記載されたXMLをコピーして、管理対象サーバー上のターゲットファイル/var/www/html/we/mysql.xmlに貼り付けます。これにより、次に示す値のインポートの手順が有効になります。

1. Tm-MySql-Configノードを展開して、サーバーインスタンスレベルの値セットを表示します。この値セットの名前は“Default mysql config values”です。
2. 内容ペインで、アプリケーション構成の値セットエディターで次の設定を構成します。
   - **ファイル名**: 管理対象サーバー上のターゲットXMLファイルの元のパスとファイル名が、[ファイル名]フィールドの右に表示されます。この値は、テンプレートに定義されたFILENAME-DEFAULTと同じ値です。このパス名がこのサーバーに対して適切な場合は、このフィールドは空にしておきます。このサーバーでは構成ファイルを別の場所に置きたい場合は、[ファイル名]フィールドでターゲットサーバー上のターゲットXMLファイルの正しいパスを設定します。
   - **エンコード**: 管理対象の構成ファイルの文字エンコードを選択します。デフォルトのエンコードは管理対象サーバーで使用されているエンコードです(UTF-16エンコードはサポートされていません)。
   - **形式の保持**: このオプションは、ターゲットサーバー上の元のXML構成ファイルのコメントを残し、順序とスペースをできる限り保存する場合に選択します。SAはターゲットファイルをできる限り元のままに保持します。詳細については、「値セットエディターでのフィールドの設定」(236ページ)を参照してください。
   - **値の保持**: 値セットに値がない場合にサーバー上の実際の構成ファイル内の値を保持するには、このオプションに対して[**はい**]を選択します。このオプションが[はい]に設定されている場合、継承階層のどれかのレベルの値によってオーバーライドされない限り、ターゲットファイルの値が使

用されます。このオプションが[いいえ]に設定されており、値セットに値が存在しない場合、構成ファイルにエントリは書き込まれません。詳細については、「値セットエディターでのフィールドの設定」(236ページ)を参照してください。

- **継承された値の表示**: このオプションを選択すると、値セットと継承レベルの値が表示されます。オフにした場合、現在の継承レベルで設定された値だけが表示されます。オンにした場合、現在のレベルで設定された値と継承された値を含めて、値セットのすべての値が表示されます。このビューは読み取り専用です。

3. 値セットエディター内で右クリックして、[**値のインポート**]を選択します。値をインポートすると、管理対象サーバー上のXMLファイルが読み込まれ、XMLファイルの内容がサーバーインスタンスレベルの値セットにコピーされます。
4. 変更を保存するには、[変更の保存]ボタンを選択します。次のステップのためにサーバー画面を開いておきます。

6. 値の編集と構成のプッシュ

最後のステップでは、値セットエディターで値を編集し、構成をサーバーにプッシュします。アプリケーション構成をプッシュすると、値セットのすべての値が、ターゲット管理対象サーバー上の構成ファイルの値を置き換えます。アプリケーション構成に含まれるスクリプトが、それぞれのタイプに基づいて実行されます。ターゲットサーバー上に構成ファイルが存在しない場合は、プッシュを実行したときにファイルが作成されます。

値を編集してアプリケーション構成をプッシュするには、次の手順を実行します。

1. [値]列を編集して、値セットの値を変更します。列の説明については、「値セットエディターの列」(237ページ)を参照してください。
2. アプリケーション構成の値を設定したら、[**プレビュー**]を選択して、サーバー上の既存のファイルと、サーバーにプッシュされるファイルを表示します。
3. [**プッシュ**]を選択して、新しいアプリケーション構成をサーバーにコピーします。
4. [構成のプッシュ]画面で、[ジョブの開始]を選択します。プッシュジョブのステータスと結果を確認します。

XMLチュートリアル2 - XML-DTD構成テンプレートの作成

この項では、XML-DTD構成テンプレートを作成して、DTDを参照するXML構成ファイルを管理する方法を、Travel Managerアプリケーションを例として説明します。Travel Managerの例の説明については、「例：Travel ManagerアプリケーションとXML構成ファイル」(302ページ)を参照してください。

最初にXML-DTDテンプレートをテキストファイルで作成し、テキストファイルをSAライブラリにインポートして、アプリケーション構成オブジェクトに追加します。次に、アプリケーション構成を管理対象サーバーにアタッチします。最後に、アプリケーション構成の値を編集し、管理対象サーバー上のターゲットXMLファイルに変更をプッシュします。

Travel ManagerのDTDベースXMLファイルmysql.xmlの例

Travel Managerアプリケーション用のXML構成ファイル`mysql.xml`を次に示します。これは`/var/www/html/we/mysql.xml`にあります。

```
<?xml version="1.0"?>
<!DOCTYPE db-config PUBLIC "-//Williams Events//Travel Manager//EN" "mysql.dtd">
<db-config>
       <db-host>localhost</db-host>
       <db-name>wrightevents</db-name>
       <db-user>root</db-user>
       <db-password>hp-pass</db-password>
</db-config>
```

Travel ManagerのXML DTDファイルmysql.dtdの例

Travel Managerアプリケーション用のDTD`mysql.dtd`を次に示します。これは`/var/www/html/we/mysql.dtd`にあります。

```
<!ELEMENT db-config (db-host,db-name,db-user,db-password)>
<!ELEMENT db-host     (#PCDATA)>
<!ELEMENT db-name     (#PCDATA)>
<!ELEMENT db-user     (#PCDATA)>
<!ELEMENT db-password (#PCDATA)>
```

1. テキストエディターでのXML-DTDテンプレートの作成

この作業では、XML-DTD構成テンプレートのソースをテキストエディターで作成します。

XML-DTD構成テンプレートをテキストエディターで作成するには、次の手順を実行します。

1. テキストエディターで、次の情報を入力します。

   ```
   <!--
   ACM-TIMEOUT = 1
   ACM-FILENAME-KEY = /files/TravelManager
   ACM-FILENAME-DEFAULT = /var/www/html/we/mysql.xml
   ACM-NAMESPACE = /TravelManager/
   ACM-DOCTYPE = db-config
   ACM-DOCTYPE-SYSTEM-ID = mysql.dtd
   ACM-DOCTYPE-PUBLIC-ID = -//Williams Events//Travel Manager//EN
   -->
   ```

   この情報は必須 (ACM-TIMEOUTを除く) であり、管理対象のXML-DTDとXMLファイルの両方を読み取るためにアプリケーション構成パーサーが使用します。

   - `ACM-TIMEOUT`: (オプション) プッシュ時に構成テンプレートのデフォルトのタイムアウト値 (10分) に加算される時間 (分) を表します。
   - `ACM-FILENAME-KEY`: `mysql.xml`ファイル名が記録される名前空間内の場所を定義します。
   - `ACM-FILENAME-DEFAULT`: 管理対象サーバー上の`mysql.xml`ファイルのデフォルトの場所 (絶対パス) を定義します。
   - `ACM-NAMESPACE`: この値は、管理対象サーバー上の`mysql.xml`ファイルから読み取った値がデータベースで記録される場所を定義します。この名前空間は一意で、パスの先頭はスラッシュ (/) である必要があります。
   - `ACM-DOCTYPE`: XMLファイル内のルート要素の名前を定義します。ルート要素は、ターゲットXML構成ファイルの開始`<!DOCTYPE`宣言の後にあります。

- ACM-DOCTYPE-SYSTEM-ID: 管理対象サーバー上の対応するDTDファイルの名前を定義します。この値は通常、XML構成ファイル内のDOCTYPE要素のSYSTEM属性に記述されています。
- ACM-DOCTYPE-PUBLIC-ID: XMLドキュメントのパブリックIDを表す文字列を定義します。この値は通常、XML構成ファイル内のDOCTYPE要素のPUBLIC-ID属性に記述されています。

2. ファイルをmysql-dtd.tplという名前で保存します。次の作業のためにファイルを開いたままにしておきます。

2. 値セットエディターでの要素記述へのカスタム設定の追加

この作業では、SAクライアントの値セットエディターでのターゲットXMLファイルの各要素の表示をカスタマイズするために、XML-DTDテンプレートに情報を追加します。

XML-DTD構成テンプレートに次の2つのオプションの設定を追加することで、ターゲットXML-DTD構成ファイルの要素がSAクライアントの値セットエディターに表示される方法をカスタマイズすることができます。

- ACM-PRINTABLE: XML-DTDテンプレートをSAクライアントに表示したときに、値セットエディターに表示されるXMLファイルの各要素のラベルを定義します。
- ACM-DESCRIPTION: SAクライアントの値セットエディターで、ACM-PRINTABLEに定義されたフィールドにマウスポインターを移動したときに表示されるテキストを定義します。

SAクライアントの値セットエディターでのこれらの要素の表示例については、XML-DTDテンプレートの図を参照してください。

この例では、明示的な方法でこれらのカスタム設定をXML-DTDテンプレート内部に配置しています。カスタム設定のこの配置方法の詳細については、「明示的表示設定と位置による表示設定」(307ページ)を参照してください。

XML-DTDテンプレートにカスタム設定を追加するには、次の手順を実行します。

1. mysql-dtd.tplファイルを開いているテキストエディターで、DTDから参照される各XML要素に次の情報を追加します。たとえば、テンプレートのメイン情報の後に、ソースXMLファイルに含まれるすべての要素のリストを追加し、その中の各要素に対して、以下の3つのACM設定タグを使用してXMLコメントを作成します。
   - ACM-ELEMENT: 次のACM-PRINTABLEおよびACM-DESCRIPTION設定が記述するXMLファイルの要素を宣言します。このオプションのデフォルトは、DTDファイル内でこのセクションの前にあった要素または属性です。

- ACM-PRINTABLE: 値セットエディターに表示する要素のわかりやすい短いラベルを設定します。
- ACM-DESCRIPTION: 要素のマウスオーバーテキストを設定します。

例のXML-DTDテンプレートファイルは次のようになります。

```
ACM-TIMEOUT = 1
ACM-FILENAME-KEY = /files/TravelManager
ACM-FILENAME-DEFAULT = /var/www/html/we/mysql.xml
ACM-NAMESPACE = /TravelManager/
ACM-DOCTYPE = db-config
ACM-DOCTYPE-SYSTEM-ID = mysql.dtd
ACM-DOCTYPE-PUBLIC-ID = -//Williams Events//Travel Manager//EN
-->
<!ELEMENT db-config (db-host,db-name,db-user,db-password)>
<!ELEMENT db-host     (#PCDATA)>
<!ELEMENT db-name     (#PCDATA)>
<!ELEMENT db-user     (#PCDATA)>
<!ELEMENT db-password (#PCDATA)>
<!--
ACM-ELEMENT = db-config
ACM-PRINTABLE = database configuration
ACM-DESCRIPTION = The db-config element specifies the data structure that
contains the information needed to connect to a database.
-->
<!--
ACM-ELEMENT = db-host
ACM-PRINTABLE = database hostname
ACM-DESCRIPTION = The db-host element specifies the name of the host computer
(the server) on which the database engine is running.
-->
<!--
ACM-ELEMENT = db-name
ACM-PRINTABLE = database name
ACM-DESCRIPTION = The db-name element specifies the name of the database.
-->
<!--

ACM-ELEMENT = db-user
ACM-PRINTABLE = database user
ACM-DESCRIPTION = The db-user element specifies the user identification used
to connect to the database.
-->
<!--
ACM-ELEMENT = db-password
ACM-PRINTABLE = database password
ACM-DESCRIPTION = The db-password element specifies the password used to
```

```
connect to the database.
-->
```

2. ファイルを保存して閉じます。

3. XML-DTD構成ファイルのインポート

この作業では、テンプレートファイルをインポートして、ターゲットXMLおよびDTDファイルを管理する新しい構成テンプレートを作成します。

XML-DTD構成ファイルをSAライブラリにインポートするには、次の手順を実行します。

1. SAクライアントナビゲーションペインで、[ライブラリ] を選択し、[タイプ別] タブを選択します。
2. [アプリケーション構成] ノードを見つけて開きます。[テンプレート] ノードを開きます。オペレーティングシステムグループを開き、テンプレートファイルが適用されるオペレーティングシステムに移動します。テンプレートは複数のオペレーティングシステムに適用することもできます。たとえば、Red Hatオペレーティングシステムバージョンの1つを選択します。
3. [**アクション**] メニューで [**テンプレートのインポート**] を選択します。
4. 前のステップで作成したファイルを見つけて選択します。エンコーディングがデフォルト以外の場合は、エンコーディングを選択します。
5. [開く] を選択します。テンプレートファイルがインポートされ、[テンプレート] 画面に表示されます。
6. プロパティビューを選択し、次の情報を入力します。
   - **名前:** mysql-dtd.tpl
   - **説明**: This is the template for the mysql.dtd (mysql.xml) file for the Travel Manager application.
   - **場所**: **SAライブラリのどこにテンプレートを保存するかを指定します。**
   - **バージョン**: 0.1.
   - **タイプ**: テンプレートファイル
   - **パーサー構文**: XML DTD構文。
   - **OS**: 適切なオペレーティングシステムを選択します。
7. [内容] ビューを選択し、インポートしたテンプレートファイルの内容を表示します。
8. 先に進む前に、[**検証**] ボタンを選択して、構文が有効であることを確認します。
9. [**ファイル**] > [**保存**] を選択します。
10. [**ファイル**] > [**閉じる**] を選択します。

4. アプリケーション構成オブジェクトの作成

アプリケーション構成とは、構成テンプレートファイルを格納するコンテナーです。このステップでは、アプリケーション構成を作成し、テンプレートをインポートします。

1. SAクライアントナビゲーションペインで、[ライブラリ] を選択し、[タイプ別] タブを選択します。
2. [アプリケーション構成] ノードを見つけて開きます。[構成] ノードを開きます。オペレーティングシステムグループを開き、アプリケーション構成が適用されるオペレーティングシステムを選択します。アプリケーション構成は複数のオペレーティングシステムに適用することもできます。これは後の手順で変更できます。
3. [**アクション**] > [**新規**] メニューを選択します。ファイルの [構成] 画面が表示され、アプリケーション構成のプロパティと内容を指定できます。
4. プロパティビューで次の情報を指定します。
   - **名前: TM-mysql-dtd**
   - **説明**: This is the application configuration for the mysql.xml and mysql.dtd files for the Travel Manager application.
   - **場所: SAライブラリのどこにアプリケーション構成を保存するかを指定します。**
   - **バージョン**: 0.1
   - **OS**: 適切なオペレーティングシステムを選択します。
5. [内容] ビューで [**アクション**] > [**追加**] を選択するか、[+] ボタンをクリックしてテンプレートをアプリケーション構成に追加します。
6. [構成テンプレートの選択] 画面で、mysql-dtd.tplテンプレートファイルを選択します。
7. [**OK**] を選択します。
8. [**ファイル**] > [**保存**] を選択すると、アプリケーション構成を保存できます。
9. [**ファイル**] > [**閉じる**] を選択します。

5. 管理対象サーバーへのアプリケーション構成のアタッチ

この作業では、Travel Managerアプリケーションがインストールされているサーバーにアプリケーション構成をアタッチし、`mysql.dtd`構成ファイルのパス名を入力します。

アプリケーション構成をサーバーにアタッチするには、次の手順を実行します。

1. SAクライアントのナビゲーションペインで [デバイス] > [サーバー] > [すべての管理対象サーバー] を選択します。
2. サーバーを選択し、[**アクション**] メニューで、[**開く**] を選択します。選択するサーバーのオペレーティングシステムが、アプリケーション構成とテンプレートに指定されたオペレーティングシステムと一致することを確認してください。
3. [管理ポリシー] タブを選択します。
4. [構成されるアプリケーション] ノードを選択します。
5. [インストール済み構成] タブを選択します。
6. [**アクション**] メニューで [**構成の追加**] を選択します。
7. [アプリケーション構成の選択] 画面で、アプリケーション構成TM-mysql-dtdを選択します。
8. [インスタンス名] フィールドに「Value set 1 for mysql.xml」と入力します。
9. [**OK**] を選択します。アプリケーション構成がサーバーにアタッチされます。
10. [**変更の保存**] ボタンを選択します。
11. 次のステップのためにアプリケーション構成画面を開いておきます。

6. 構成ファイルからの値のインポート

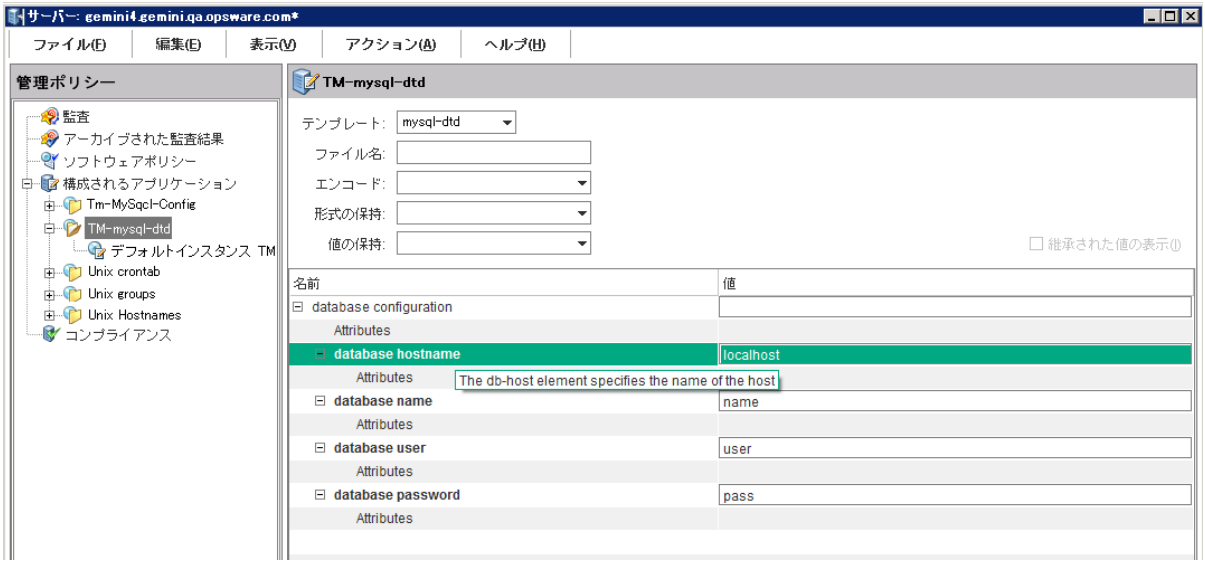
次のステップでは、値セットの値を設定します。値セットの値は手動でも設定できますが、最も簡単な方法は、既存の構成ファイルから値をインポートすることです。このステップでは、サーバー上の構成ファイルから値をインポートします。

以下で説明するように構成ファイルから値をインポートするには、上のTravel ManagerのDTDベースXMLファイルmysql.xmlの例に記載されたXMLをコピーして、管理対象サーバー上のターゲットファイル/var/www/html/we/mysql.xmlに貼り付けます。上のTravel ManagerのXML DTDファイルmysql.dtdの例に記載されたDTDをコピーして、ターゲットファイル`/var/www/html/we/mysql.dtd`に貼り付けます。これにより、次に示すインポートの手順が有効になります。

1. サーバーの管理ポリシーで、[構成されるアプリケーション] ノードを開きます。サーバーにアタッチされているアプリケーション構成が表示されます。
2. [TM-mysql-dtd] ノードを開きます。サーバーインスタンス値セットが表示されます。これはTM-mysql-dtdノードの下のノードです。
3. “Value set 1 for mysql.xml” ノードを選択します。これはmysql-dtd.tpl構成テンプレートのサーバーインスタンス値セットです。

4. [値] 列の下の任意の値を右クリックして、[**値のインポート**] メニューを選択します。
5. 確認ダイアログで [はい] を選択します。これにより、/var/www/html/we/mysql.xmlからサーバーインスタンスレベルの値セットに値がインポートされます。
6. [変更の保存] ボタンを選択します。次の図は、XML-DTDテンプレートとサーバーインスタンス値セットを示します。ACM-DESCRIPTION要素によるマウスオーバーテキストが表示されています。

XML-DTD構成テンプレートの値セットとマウスオーバーテキスト



7. 次のステップのためにアプリケーション構成を表示しておきます。

7. 値の編集と構成のプッシュ

最後のステップでは、値セットエディターで値を編集し、構成をサーバーにプッシュします。アプリケーション構成をプッシュすると、値セットのすべての値が、ターゲット管理対象サーバー上の構成ファイルの値を置き換えます。アプリケーション構成のすべてのスクリプトも実行されます。ターゲットサーバー上に構成ファイルが存在しない場合は、プッシュの際にファイルが作成されます。

値を編集してアプリケーション構成をプッシュするには、次の手順を実行します。

1. ナビゲーションペインで “value set 1 for mysql.xml” を選択して、サーバーインスタンスレベルの値セットを表示しておきます。
2. [値] 列のパスワードの値を変更します。
3. [**変更の保存**] を選択します。

4. [**プレビュー**] を選択して、サーバー上の既存の構成ファイルと、サーバーにプッシュされる構成ファイルとの違いを表示します。
5. 比較画面を確認したら、[閉じる] ボタンを選択します。
6. [**プッシュ**] を選択します。[構成のプッシュ] ウィザードが表示されます。
7. [**ジョブの開始**] を選択します。プッシュ操作が開始されます。
8. プッシュジョブのジョブステータスを確認します。どれかのステップを選択すると、そのステップの詳細が表示されます。
9. ジョブが完了したら、[閉じる] ボタンを選択します。
10. サーバーにログオンして`mysql.xml`構成ファイルを表示し、サーバー上で更新されていることを確認できます。

CMLチュートリアル1 - 単純なWebアプリケーションサーバーに対するアプリケーション構成の作成

本項では、2つのサーバー上で動作するWebアプリケーションサーバーに対する単純な構成ファイルを作成して管理する方法を示します。各サーバーはそれぞれWebアプリケーションサーバーを実行しており、別々に構成する必要があります。このチュートリアルでは、アプリケーション構成、構成テンプレート、値セット、および2つのアプリケーション構成インスタンス(各サーバーに1つずつ)を作成する方法を示します。最後に、アプリケーション構成を各サーバーにプッシュする方法を示します。

1. 管理する構成ファイルの決定

Webアプリケーションサーバーは、WASconfig.txtという名前の1つの構成ファイルを使用します。このファイルは、/opt/WAS/WASconfig.txtディレクトリにあります。このファイルの内容は次のとおりです。

```
size=1000
dir=/tmp/WAS_001
primary=yes
```

2. 構成ファイルのテンプレートの作成

テンプレートを作成するには、次の2つの方法があります。

- テキストファイルでテンプレートを作成し、テキストファイルをSAライブラリにインポートします。
- SAライブラリで直接テンプレートを作成します。

以下では両方の方法を説明します。どちらかの方法を選んで、対応する手順を実行してください。

テンプレートファイルを作成してSAにインポート

1. テキストエディターで、空のファイルに構成ファイルをコピーします。

```
size=1000
dir=/tmp/WAS_001
primary=yes
```

2. このファイルをモデル化するテンプレートをCMLで作成します。最初に、コメントブロックと、名前空間およびターゲット構成ファイル名を定義する必須のCMLメタデータを追加します。
   - 名前空間は、このテンプレートの情報がデータベースに記録されるキーを定義します。
   - ファイル名キーは、デフォルトのファイル名がデータベースに記録されるキーを定義します。
   - デフォルトファイル名は、結果の構成ファイルに使用される名前を指定します。

```
@#########################################################
# /opt/WAS/WASconfig.txt                                 #
# Version 1.0                                            #
# Author <name>                                          #
#########################################################@
@!namespace=/WAS-server-namespace/@
@!filename-key=”/WAS-server”@
@!filename-default=”/opt/WAS/WASconfig.txt”@
size=1000
dir=/tmp/WAS_01
primary=yes
```

3. 次に、構成ファイルの可変部分を、CMLタグを使用して変数に変換します。

```
@#########################################################
# /opt/WAS/WASconfig.txt                                 #
# Version 1.0                                            #
# Author <name>                                          #
```

```
############################################################@
@!namespace=/WAS-server-namespace/@
@!filename-key="/WAS-server"@
@!filename-default="/opt/WAS/WASconfig.txt"@
size=@value_of_size;int@
dir=@value_of_dir;string@
primary=@value_of_primary;boolean@
```

4. ファイルを拡張子 ".tpl" で保存します (例、WASconfig_txt.tpl)。
5. テンプレートファイルをSAライブラリにインポートします。「テンプレートファイルのインポートと検証」(280ページ)の手順を実行します。

SAで直接テンプレートファイルを作成

1. SAクライアントで、[**ライブラリ**] タブを選択します。
2. [**タイプ別**] タブを選択します。
3. [アプリケーション構成] ノードと[テンプレート] ノードを開きます。アプリケーションが動作するOSファミリとOSバージョンに移動します。この例では、Red Hat Enterprise Linux AS 4を選択します。
4. **[アクション] > [新規]** を選択します。[テンプレート] 画面が表示されます。
5. テンプレート名 "WASconfig_txt.tpl" と簡単な説明を入力します。テンプレートファイルを格納するSAライブラリの場所を選択します。バージョン文字列を設定します。[タイプ] を [テンプレートファイル] に設定します。[パーサー構文] を [CML構文] に設定します。
6. [**内容**] ビューを選択して、テキストエディターを表示します。
7. CMLテキストを入力するか貼り付けます。これは上に示したCMLテキストと同じです。
8. **[アクション] > [検証]** を選択して、CMLの構文をチェックします。必要な修正を行います。
9. **[ファイル] > [保存]** を選択してテンプレートを保存します。
10. テンプレート画面を閉じます。

3. アプリケーション構成オブジェクトの作成

構成テンプレートを含むアプリケーション構成オブジェクトを作成します。

1. SAクライアントで、[**ライブラリ**] タブを選択します。
2. [**タイプ別**] タブを選択します。

3. [アプリケーション構成] ノードと[構成] ノードを開きます。アプリケーションが動作するOSファミリとOSバージョンに移動します。この例では、Red Hat Enterprise Linux AS 4を選択します。
4. [**アクション**] > [**新規**] を選択します。[構成] 画面が表示されます。
5. アプリケーション構成の名前 “WAS-app-config”、簡単な説明、バージョン文字列を入力します。アプリケーション構成を格納するSAライブラリの場所を選択します。
6. [**ファイル**] > [**保存**] を選択して、アプリケーション構成を保存します。

4. アプリケーション構成オブジェクトへのテンプレートファイルの追加

1. 前の手順で作成した “WAS-app-config” アプリケーション構成オブジェクトを開きます。
2. [**構成される値**] ビューを選択します。
3. [+] ボタンを選択するか、[**アクション**] > [**追加**] を選択します。[構成テンプレートの選択] 画面が表示されます。
4. “**WASconfig_txt.tpl**”テンプレートファイルを選択して、[**OK**] を選択します。
5. [**ファイル**] > [**保存**] を選択して、アプリケーション構成オブジェクトの変更を保存します。
6. [**ファイル**] > [**閉じる**] を選択して、アプリケーション構成オブジェクトを閉じます。

5. アプリケーション構成オブジェクトのサーバーへのアタッチ

Webアプリケーションサーバーを実行しているサーバーは、RHEL001とRHEL008の2つです。RHEL001がプライマリサーバー、RHEL008がセカンダリサーバーです。次の手順で、これら2つのサーバーにアプリケーション構成オブジェクトをアタッチして、アプリケーション構成の2つのインスタンスを作成します。

1. SAクライアントでプライマリサーバーRHEL001を見つけます。
2. RHEL001サーバーを選択し、[**アクション**] > [**開く**] を選択します。
3. [管理ポリシー] タブを選択します。
4. [構成されるアプリケーション] ノードを選択します。

5. [**アクション**] > [**構成の追加**] メニューを選択します。
6. "**WAS-app-config**"アプリケーション構成オブジェクトを選択します。
7. [インスタンス名] フィールドに「Primary Instance of WAS-app-config」と入力して、[**OK**] を選択します。
8. [**変更の保存**] を選択します。サーバーRHEL001に対するアプリケーション構成のインスタンスが作成されます。
9. 上記の手順をセカンダリサーバーRHEL008に対しても繰り返します。ただし、[インスタンス名] フィールドには「Secondary Instance of WAS-app-config」と入力して [OK] を選択します。サーバーRHEL008に対するアプリケーション構成の2つ目のインスタンスが作成されます。

6. デフォルト値の設定

2つのサーバーに対する構成ファイルに必要な値を次に示します。

Webアプリケーションサーバーを実行する2つのサーバーに対する構成値

サーバー: RHEL001	サーバー: RHEL008

`size=1000`
`dir=/tmp/WAS_001`
`primary=yes`
	`size=1000`
	`dir=/tmp/WAS_008`
	`primary=no`

構成ファイルにデフォルト値を設定できます。個々のサーバーは、デフォルト値を継承することもオーバーライドすることもできます。個々のサーバーは、デフォルト値をオーバーライドしない場合は、継承したデフォルト値を使用します。

次の表は、デフォルト値に設定される値と、個々のサーバーで設定される値を示します。

表7: Webアプリケーションサーバーに対する構成ファイルのアプリケーションレベルのデフォルト値

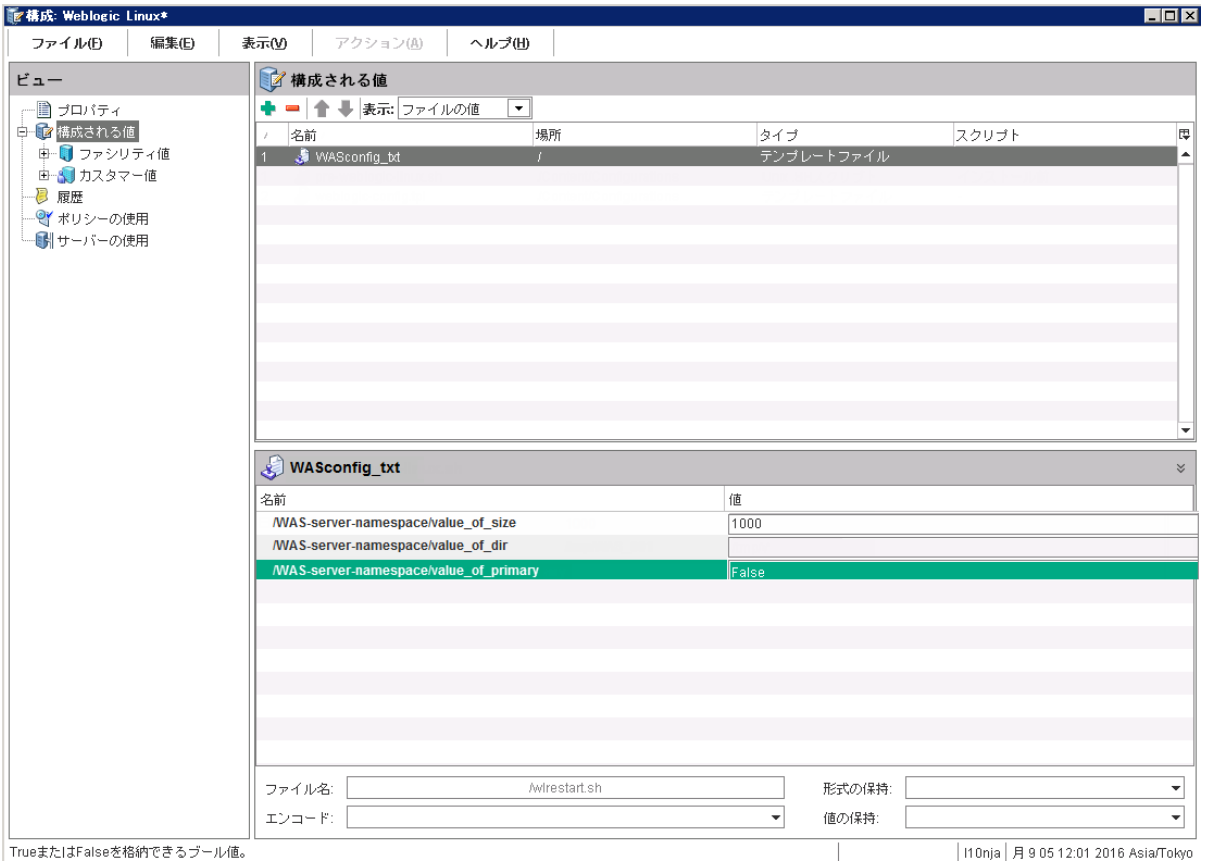
デフォルト値	説明
`size=1000`	これはアプリケーションレベルでデフォルト値1000に設定します。このアプリケーション構成にアタッチされたすべてのサーバーは、オーバーライドしない限りこの値を使用します。
`dir`	これはデフォルト値に設定しないでください。各サーバーは、サーバーレベルまたはサーバーインスタンスレベルでこの値を設定します。
`primary=no`	これはアプリケーションレベルでデフォルト値 "no" に設定します。このアプリケーション構成にアタッチされたすべてのサーバーは、オーバーライドしない限りこの値を使用

表 7: Webアプリケーションサーバーに対する構成ファイルのアプリケーションレベルのデフォルト値 (続き)

デフォルト値	説明
	します。

アプリケーションレベルのデフォルト値の設定

1. 上で作成したアプリケーション構成オブジェクトを開きます。
2. [**構成される値**] ビューを選択します。
3. **WAS-config-template.tpl**テンプレートファイルを選択します。
4. [表示] ドロップダウンリストで [**ファイルの値**] を選択します。アプリケーションレベルでのテンプレートのデフォルト値が表示されます。
5. 次に示すように、“value_of_size” を1000に、“value_of_primary” を “False” (大文字と小文字を区別) に設定します。“value_of_dir” は各サーバーで設定する必要があるので、デフォルト値は設定しません。

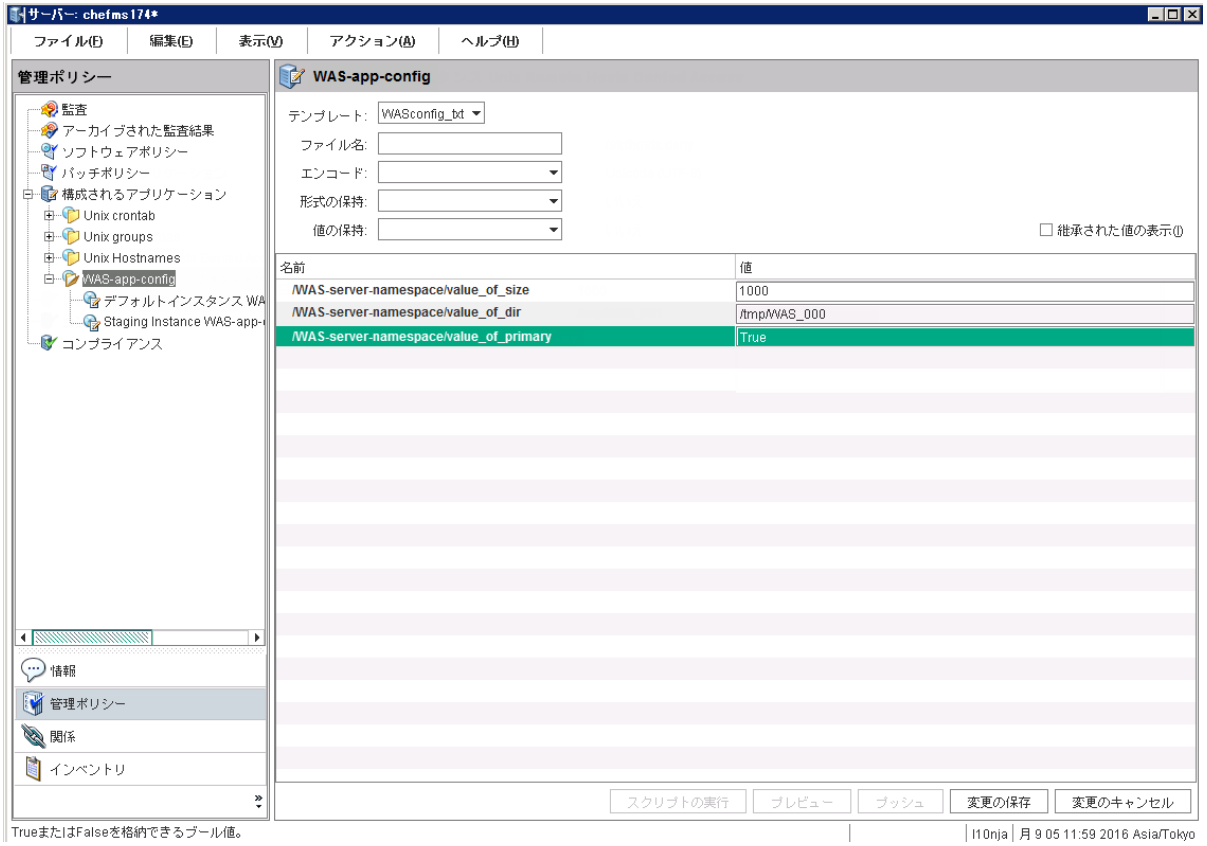


6. [**ファイル**] > [**保存**] を選択して、アプリケーションレベルのデフォルト値を保存します。
7. [**ファイル**] > [**閉じる**] を選択します。

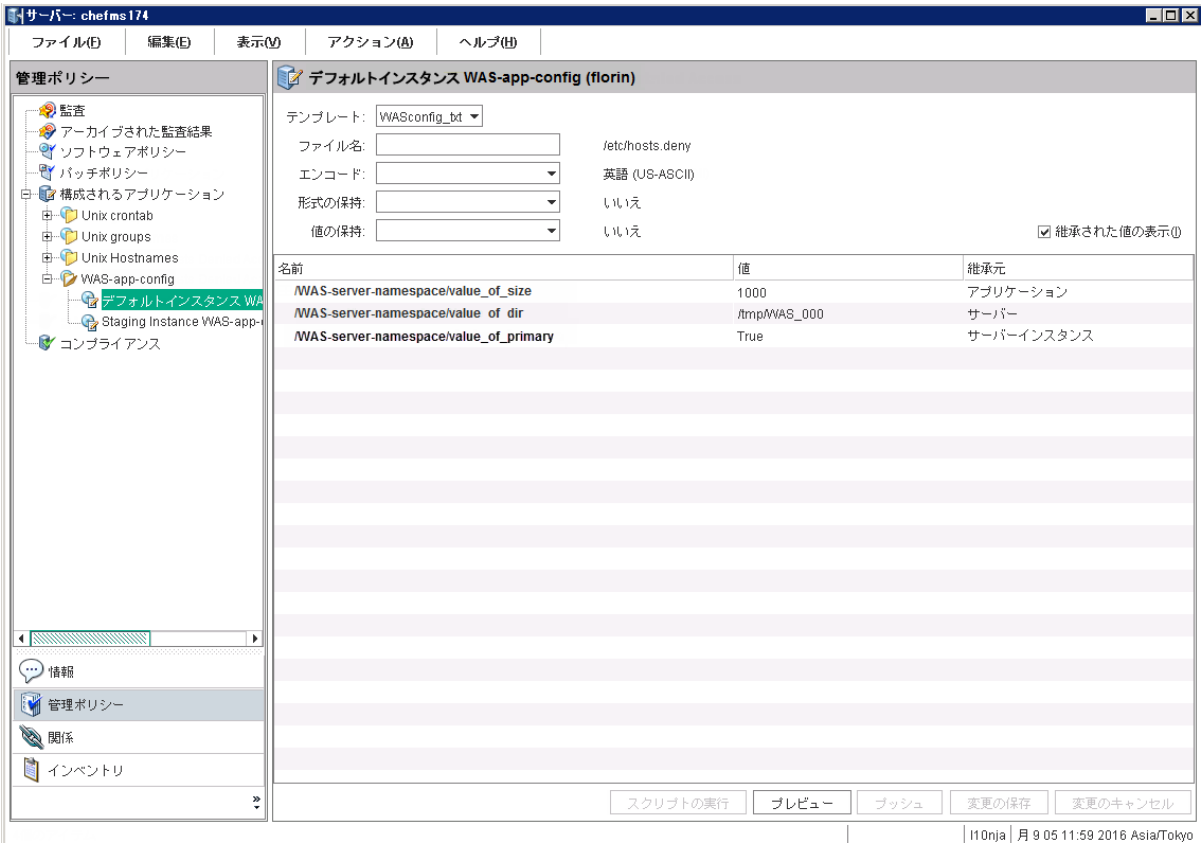
RHEL001に対するサーバーレベルのデフォルト値の設定

サーバーRHEL001は、dir=/tmp/WAS_001とprimary=yesをサーバーレベルで設定する必要があります。sizeについては、アプリケーションレベルで設定した値を使用するため、値を設定する必要はありません。

1. SAクライアントでRHEL001サーバーを見つけます。
2. RHEL001サーバーを選択し、[アクション] > [開く] を選択します。
3. [管理ポリシー] タブを選択します。
4. [構成されるアプリケーション] ノードを開いて、“WAS-app-config” アプリケーション構成オブジェクトを表示します。
5. このサーバーにアタッチされた “WAS-app-config” アプリケーション構成オブジェクトを選択します。サーバーレベルで設定されたデフォルト値が表示されます。サーバーレベルで設定された値は、サーバーインスタンスレベルでオーバーライドされない限り、そのサーバー上のアプリケーション構成のすべてのインスタンスに適用されます。
6. 次に示すように、“value_of_dir” のサーバーレベルのデフォルト値を “/tmp/WAS_001” に設定します。“value_of_size” と“value_of_primary” の値はアプリケーションレベルから継承されるので、これらに対してはデフォルト値を設定しません。



7. [**変更の保存**] ボタンまたは [ファイル] > [保存] メニューを選択して、**サーバーレベルのデフォルト値**を保存します。
8. WAS-app-configノードを開いて、アプリケーション構成インスタンス “Primary Instance of WAS-app-config” を表示します。
9. インスタンス “Primary Instance of WAS-app-config” を選択します。**インスタンスレベルのデフォルト値**が表示されます。これは最下位の値設定レベルであり、他のすべてのレベルをオーバーライドします。インスタンスレベルには値が定義されていません。
10. [継承された値の表示] を選択して、アプリケーションレベルのデフォルトとサーバーレベルのデフォルトから継承される値を表示します。“value_of_size” と “value_of_primary” はアプリケーションレベルから、“value_of_dir” はサーバーレベルから継承されます。
11. インスタンスレベルのデフォルト値を設定できるように、[継承された値の表示] をオフにします。
12. “value_of_primary” を “True” に設定します (大文字と小文字を区別)。
13. [**変更の保存**] ボタンまたは [ファイル] > [保存] メニューを選択して、**インスタンスレベルのデフォルト値**を保存します。
14. [継承された値の表示] をもう一度選択して、アプリケーションレベル、サーバーレベル、インスタンスレベルから継承された値を表示します。次に示すように、“value_of_size” はアプリケーションレベルから、“value_of_dir” はサーバーレベルから、“value_of_primary” はインスタンスレベルから継承されています。



RHEL008に対するサーバーレベルのデフォルト値の設定

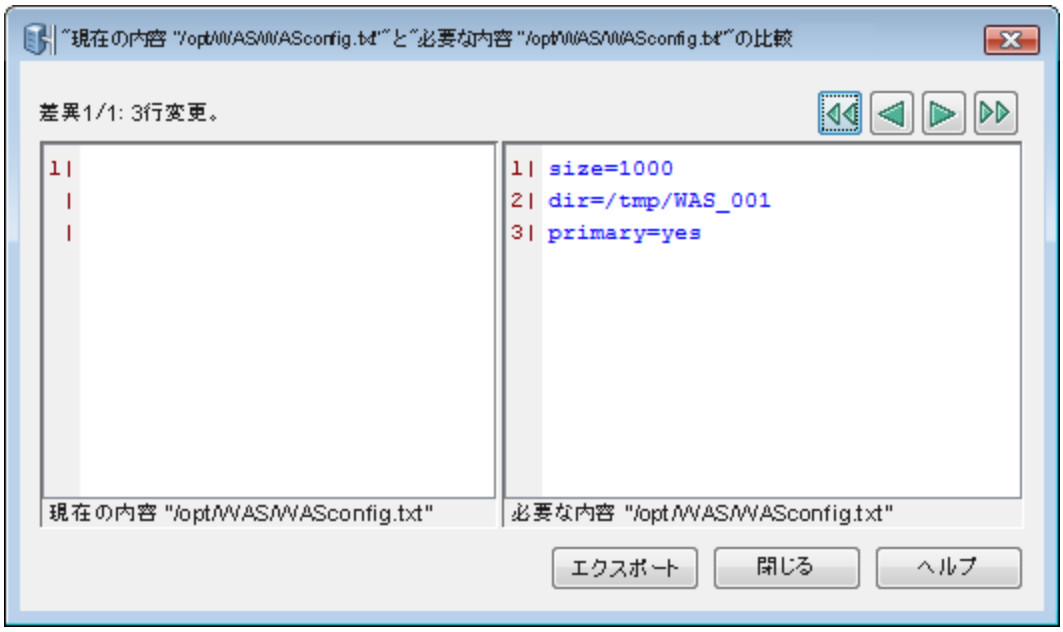
サーバーRHEL008は、`dir=/tmp/WAS_008`をサーバーレベルで設定する必要があります。sizeとprimaryについては、アプリケーションレベルで設定した値を使用するため、値を設定する必要はありません。

1. SAクライアントでRHEL008サーバーを見つけます。
2. RHEL008サーバーを選択し、**[アクション] > [開く]** を選択します。
3. [**管理ポリシー**] タブを選択します。
4. [構成されるアプリケーション] ノードを開いて、“WAS-app-config” アプリケーション構成オブジェクトを表示します。
5. このサーバーにアタッチされた “WAS-app-config” アプリケーション構成オブジェクトを選択します。サーバーレベルで設定されたデフォルト値が表示されます。サーバーレベルで設定された値は、サーバーインスタンスレベルでオーバーライドされない限り、そのサーバー上のアプリケーション構成のすべてのインスタンスに適用されます。

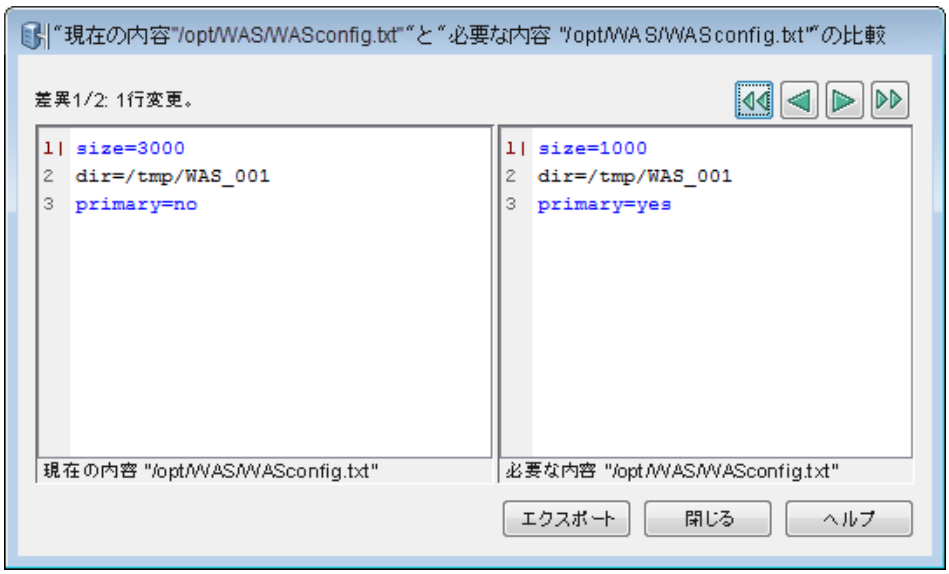
6. “value_of_dir” のサーバーレベルのデフォルト値を “/tmp/WAS_008” に設定します。“value_of_size” と “value_of_primary” の値はアプリケーションレベルから継承されるので、これらに対してはデフォルト値を設定しません。
7. [**変更の保存**] または **[ファイル] > [保存]** メニューを選択して、**サーバーレベルのデフォルト値**を保存します。
8. WAS-app-configノードを開いて、アプリケーション構成インスタンス “Secondary Instance of WAS-app-config” を表示します。
9. インスタンス “Secondary Instance of WAS-app-config” を選択します。**インスタンスレベルのデフォルト値**が表示されます。これは最下位の値設定レベルであり、他のすべてのレベルをオーバーライドします。インスタンスレベルには値が定義されていません。
10. [継承された値の表示] を選択して、アプリケーションレベルのデフォルトとサーバーレベルのデフォルトから継承される値を表示します。“value_of_size” と “value_of_primary” はアプリケーションレベルから、“value_of_dir” はサーバーレベルから継承されます。

7. 実際の構成ファイルと構成テンプレートの比較

オプションで、アプリケーション構成に指定された値と、サーバー上の構成ファイル内の実際の値を比較できます。このためには、サーバー画面で [プレビュー] ボタンを選択します。次に示すのは、サーバー上に構成ファイルがない状態でのRHEL001に関する比較です。



次に示すのは、サーバー上に既存の構成ファイルがあり、その値がアプリケーション構成に指定された値と異なっている場合の比較です。



8. 構成変更のサーバーへのプッシュ

1. SAクライアントでRHEL001サーバーを見つけます。
2. RHEL001サーバーを選択し、[**アクション**] > [**開く**] を選択します。
3. [**管理ポリシー**] タブを選択します。
4. [構成されるアプリケーション] ノードを開いて、“WAS-app-config” アプリケーション構成オブジェクトを表示します。
5. “**WAS-app-config**” アプリケーション構成ノードを開いて、“Primary Instance of WAS-app-config” インスタンスを表示します。
6. “**Primary Instance of WAS-appconfig**” インスタンスを選択します。
7. [**プッシュ**] ボタンを選択します。
8. スケジュール設定と通知に関してジョブのデフォルトを使用するには、[**ジョブの開始**] を選択します。変更するには、[**次へ**] を選択します。
9. [スケジュール設定] 画面では、いつ構成のプッシュジョブを実行するかを指定できます。[**次へ**] を選択します。
10. [通知] 画面では、ジョブが成功または失敗したときに電子メールメッセージを受信する1人以上の人を指定できます。また、チケットIDも指定できます。[次へ] を選択します。

11. [**ジョブの開始**] を選択します。SAはテンプレートと値セットから構成ファイルを生成し、結果の構成ファイルをサーバーにプッシュして、結果を表示します。
12. [**閉じる**] を選択します。

さらに複雑な構成ファイルを扱うチュートリアルについては、「CMLチュートリアル2 - Webサーバー構成ファイルのテンプレートの作成」(338ページ)を参照してください。

CMLチュートリアル2 - Webサーバー構成ファイルのテンプレートの作成

このチュートリアルでは、**構成モデリング言語** (CML) を使用して、Microsoft Internet Information Services (IIS) Webサーバーの構成ファイルUrlScan.iniに基づく構成テンプレートを作成する方法を説明します。CML言語を使用してこのファイルに基づくテンプレートファイルを作成することにより、構成ファイルを管理対象サーバー上で管理できます。

このチュートリアルではCMLに関する詳細な説明は行いませんが、UrlScan.iniからCMLテンプレートを作成することで、CMLと、構成ファイルからの構成テンプレートの作成に関する基礎を学ぶことができます。

UrlScan.iniファイルの例はUrlScan.iniファイルの例にあります。CMLファイル全体のリストは完成したurl_scan_ini.tpl CMLテンプレートにあります。

このチュートリアルを実行するには、次のものが必要です。

- `UrlScan.ini`に関するドキュメント。これはMicrosoft IISドキュメントの中にあります。
- `UrlScan.ini`ファイル。
- CMLファイルを作成するためのテキストエディター。

1. ネイティブ構成ファイルとドキュメントの分析

管理するアプリケーション構成ファイルを特定したら、最初に行うのは、ネイティブ構成ファイルとそのドキュメントを分析することです。構成ファイルの目的、ファイルのすべての要素、構成ファイルで管理するデータの種類を理解する必要があります。

このチュートリアルでは、UrlScan.iniファイルを使用します。例のリストはUrlScan.iniファイルの例にあります。UrlScan.iniファイルは、システム管理者がMicrosoft Internet Information Services (IIS) Webサーバーを構成するために使用します。UrlScan.iniファイルは、[Options]、[AllowVerbs]、[DenyVerbs]、[DenyHeaders]、[AllowExtensions]、[DenyExtensions] などのセクションから構成されます。各セクション

で、IIS管理者は、特定の種類のHTTP要求をIISサーバーで許可するか拒否するかを示すさまざまな構成を設定できます。これらのセクションの順序は任意です。ただし、各セクション内部の情報は、適切な順序で並んでいる必要があります。たとえば、[AllowVerbs] セクションの後には、Webサイトへのアクセスを許可されるHTTP要求が記載されます。

UrlScan.iniの内容は、動詞やファイル拡張子などの文字列のリストと、ブール値 “1” (真) または “2” (偽) を取るオプションです。

2. CMLコメントブロックの作成

CMLテンプレートは、ファイル拡張子.tplを持つシンプルテキストファイルです。テキストエディターで、Url_Scan_ini.tplという名前の新規テキストファイルを作成します。拡張子.tplは、SAでCMLテンプレートに使用される標準の(ただし必須ではない)ファイル拡張子です。

ファイルの先頭に、テンプレートに関する情報を記載した次のCMLコメントブロックを作成します。

```
@################################################
# \system32\inetsrv\urlscan.ini (Windows)    #
# Version 1.0                                  #
# Joe Author (joe_author@your_company.com)   #
################################################@
```

CMLコメントタグは次の構文を使用します。

```
@# <1行のコメント>
```

または

```
@## <複数行にわたるコメント>
        <複数行にわたるコメント> #@
```

3. CMLセットアップ命令の作成

セットアップセクションでは、CMLファイルの解釈方法をパーサーに指示します。namespace、filename-key、filename-defaultの各命令は、すべてのCMLファイルに必須です。次に示すその他の命令は省略可能です。これらの命令は、スペースの処理、ブール値、コメントの形式、順序規則を定義します。

基本的なセットアップセクションを作成するには、CMLテンプレートのコメントブロックの後に次の情報を入力します。

```
@!namespace=/security/@
@!filename-key="/test";filename-default="/c/UrlScan.ini"@
```

```
@!optional-whitespace@
@!boolean-yes-format="1";boolean-no-format="0"@
@!line-comment-is-semicolon@
@!unordered-lines@
```

このように、CML命令タグの先頭は“@!”で、末尾は“@”です。

次の行では、2つのCML命令を1行に結合しています。

```
@!filename-key="/test";filename-default="/c/UrlScan.ini"@
```

この行は、次のように2行に分けて書くこともできます。

```
@!filename-key="/test"@
@!filename-default="/c/UrlScan.ini"@
```

セットアップ命令

次にセットアップ命令の説明を示します。

CMLテンプレートのセットアップ命令

CMLタグ	説明
@!namespace=/security/@	`@!namespace`命令は、このCMLテンプレートで使用される名前空間を定義します。これは、このCMLテンプレートで使用される値がデータベースのどこに記録されるかを定義するものです。各CMLテンプレートは、他のテンプレートと名前が衝突しないように、固有の名前空間を使用する必要があります。 この例では、名前空間は`/security`です。値セットはすべてこの名前空間に記録されます。
@!filename-key="/files/urlscan_ini";filename-default="/c/urlscan.ini"@	`@!filename-key`命令は、ファイル名が記録される名前空間内の場所を定義します。値の先頭が“/”の場合は、独立した名前空間を定義します。値の先頭が“/”でない場合は、`@!namespace`命令で定義された名前空間の後に追加されます。 この例では、デフォルトのファイル名は名前空間“`/file/urlscan-ini`”を使用してデータベースに記録されます。 `@!filename-default`命令は、ネイティブ構成ファイルがサーバー上に保存されるデフォルトのパスを定義します。このパスは、SAクライアントから変更できます。 この例では、構成ファイルが管理対象サーバーにプッシュされると、`/c/urlscan.ini`に置かれます。 パス名には必ずスラッシュを使用します。
@!optional-whitespace@	この命令は、構成ファイルのアイテム間のスペースが省略可能であることを示します。たとえば、このオプションを設定した場合、次のエントリはどちらも有効です。

CMLテンプレートのセットアップ命令 (続き)

CMLタグ	説明
	Key = "value" `Key="value"`
@!boolean-yes-format="1";boolean-no-format="0"@	この命令は、構成ファイルで使用できるブール値を定義します。この例では、真は文字`1`で、偽は文字`0`で示されます。その他の値はブール値には使用できません。
@!line-comment-is-semicolon@	構成ファイル内でセミコロンより後の部分をすべて無視するようにパーサーに指示します。これにより、ネイティブ構成ファイルのセミコロンから始まるコメントに対応できます。
@!unordered-lines@	構成ファイルの各セクションの順序が任意であることをパーサーに指示します。`ordered-lines`を使用した場合、構成ファイルの順序はテンプレートの順序と一致する必要があります。

4. [Options] セクションの定義 — ブロックの開始

次には、CML命令をテンプレートに追加します。UrlScan.iniファイルのセクションのうち、CMLで最初にモデル化するのは、[Options] セクションです。ここには構成ファイルに関するいくつかのオプションがあります。

CMLでは、構成ファイル内の情報のセクションに複数の種類のデータ (CMLパーサーが異なる方法で読み取る必要があるデータ) が存在する場合、「ブロック」を作成することで、情報の各セクションを別々に処理することができます。一般的に、CMLのブロックは、CMLファイルの特定のセクションに対して特別なパーサールールを定義するために使用します。[Options] セクションには2つの情報「ブロック」があります。1つはセクションのタイトル“[Options]”で、もう1つはこのセクションのすべてのオプションです。これらのブロックはまとまって存在するので、これらを設定するには異なるレベルを使用します。すなわち、1番目のブロック (セクションのタイトル) をレベル1、2番目のブロック (セクションの内容) をレベル2にします。このようにブロックをネストすることで、ブロック内のセクションをパーサーで読み取る際にまとめておくことができます。

1. [Options] セクションを定義するには、次の行を入力します。

   ```
   @1[;optional;ordered-lines@
   [Options]
   @2[;unordered-lines@
   ```

2. UrlScan.iniファイルの [Options] セクションには、キーと値のペアが含まれます。このセクションには2種類のデータ (見出し、キーと値のペアのリスト) が含まれるので、2つのレベルに設定されたブロックタグ ([) を使用します。第1レベルのブロックはテキスト文字列“[Options]”を処理し、第2レベルのブロックはこのセクション内のすべてのキーと値のペアを処理します。

次の表に、[Options] セクションに対して2つのブロックレベルを開く方法を示します。

[Options] セクションの開始のマークアップ

CMLタグ	説明
@1[;optional;ordered-lines@	番号1は、複数行ブロックの第1レベルを設定します。 [角括弧は新しいブロックを開始します。 optional このブロック全体が構成ファイルで省略可能であることを示します。 ordered-lines このタグの後にくるもの(文字列 [Options])は、ネイティブUrlScan.ini構成ファイルで最初に来る必要があることを示します。言い換えれば、タイトル[Options]は実際のオプションよりも先に現れる必要があります。
[Options]	ネイティブ構成ファイルのセクションを示す文字列。この文字列は構成ファイルにそのまま現れます。
@2[;unordered-lines@	番号2は、ブロックの第2レベルを設定します。 [角括弧は新しいブロックを開始します。この例では、前のレベル1のブロック内部にネストされたレベル2のブロックです。 unordered-lines ブロック内で[Options]の後に来る行は、構成ファイル内で任意の順序を取れることを示します。すなわち、[Options]セクションのすべてのキーと値のペアは任意の順序で現れることができます。

次に、構成ファイルの[Options]セクションに現れる可能性があるすべてのオプションを定義します。これらのエントリのほとんどは、CML置換タグを使用します。これらは単純なキーと値のペアで、1つの値を置き換えるだけで済むからです。次の表に、各オプションに対応するCMLを示します。

表10: UrlScan.iniの [Options] セクションのキーと値のペアのマークアップ

CMLタグ	説明
AllowDotInPath = @allow_dot_in_path;boolean@	注: キーと値のペアはすべて、(特に指定しない限り)次の例に類似した構文を使用します。 文字列リテラル = @ソース;タイプ@ 文字列リテラルは、構成ファイルに現れる実際のオプション名を定義します。ソースは、値セットの値が記録されるデータベースの場所です。タイプは、値セットに記録されるデータのタイプです。 @allow_dot_in_path この文字列は、この値を記録する名前空間のパスを定義します。この例では、名前空間は相対的なので、テンプレートのヘッダー(@!namespace=/security/@)で定義した名前空間の後に追加され、その名前空間の場所に値が記録されます。すなわち、データ

表10: UrlScan.iniの [Options] セクションのキーと値のペアのマークアップ (続き)

CMLタグ	説明
	ベースに値を記録するために使用されるキーは/security/allow_dot_in_pathです。 このタグは次のように書くこともできます。 AllowDotInPath = @/security/allow_dot_in_path;boolean@ boolean キーと値のペアのタイプはブール値なので、CMLタイプbooleanが使用されます。このテンプレートのヘッダーで、ブール値の真の値が1と定義されているので、IIS管理者が値セットを設定する際に、IISサーバーのパスにドットを許可するには1を入力する必要があります。
AllowHighBitCharacters = @allow_high_bit_characters;boolean@	前の例と同様、AllowHighBitCharactersは構成ファイル内のオプション、allow_high_bit_charactersは相対名前空間パス、booleanはデータのタイプです。 このIISオプションは、URLにハイビット文字が使用できるかどうかを選択するためのもので、構成ファイル内の1は真、0は偽を示します。
AllowLateScanning = @allow_late_scanning;boolean@	IIS管理者が、後でのURLのスキャンを許可するかどうかを選択するために使用します。値を記録する名前空間の場所を定義します。booleanは、このキーが構成ファイルで1 (真) または 0 (偽) を取りうることを示します。
AlternateServerName = @alternate_servername@	ユーザーが入力したか構成ファイルから読み取られた代替サーバー名を記録する名前空間の場所を定義します。タイプは指定されていないので、デフォルトの文字列タイプになります。
EnableLogging = @enable_logging;boolean@	ログ記録をオンにするために使用します。構成ファイル内の1は真、0は偽を表します。
LoggingDirectory = @logging_directory;dir@	ログ記録をオンにしたときに、ログファイルを格納するディレクトリを選択します。タイプdirはディレクトリを示します。
LogLongURLs = @log_long_urls;boolean@	サーバーにアクセスするURLをログに記録するかどうかを選択します。構成ファイル内の1は真、0は偽を表します。
NormalizeUrlBeforeScan = @normalize_url_before_scan;boolean@	サーバーが読み取る前にURLを正規化するかどうかを選択します。構成ファイル内の1は真、0は偽を表します。
PerDayLogging = @per_day_logging;boolean@	日次ログ記録をオンにするかどうかを選択します。構成ファイル内の1は真、0は偽を表します。
PerProcessLogging = @per_process_logging;boolean@	プロセス単位のログ記録をオンまたはオフにするために使用します。構成ファイル内の1は真、0は偽を表します。
RejectResponseUrl =	構文:

表 10: UrlScan.iniの [Options] セクションのキーと値 のペアのマークアップ (続き)

CMLタグ	説明
@reject_response_url;string;r”(HTTP_URLSCAN_STATUS_HEADER)|(HTTP_URLSCAN_ORIGINAL_VERB)|(HTTP_URLSCAN_ORIGINAL_URL)”;optional@	文字列リテラル = @ソース;タイプ;r”正規表現”;オプション@ reject_response_url 名前空間内で文字列が記録されるパスを定義する文字列リテラル。 string 拒否URL応答のデータ型が文字列であることを示します。 r” 正規表現の前に付ける文字列範囲指定子です。この例では、文字列リテラルの範囲です。 (HTTP_URLSCAN_STATUS_HEADER)|(HTTP_URLSCAN_ORIGINAL_VERB)|(HTTP_URLSCAN_ORIGINAL_URL)” パーサーが読み取る文字列リテラル(拒否されるURL応答)で、ステータスヘッダー、元の動詞、元のURLを表します。 optional この値が省略可能であることを示します。すなわち、RejectResponseUrlオプションはUrlScan.iniファイルになくてもかまいません。
RemoveServerHeader = @remove_server_header;boolean@	RemoveServerHeading機能をオンまたはオフにできます。オンにした(1に設定した)場合、クライアントに送信される拒否応答では、メッセージのサーバーヘッダーが削除されます。構成ファイル内の1は真、0は偽を表します。
UseAllowVerbs = @use_allow_verbs;boolean@	UseAllowVerbs機能をオンまたはオフにできます。オンにした(1に設定した)場合、UrlScan.iniファイルのAllowVerbsセクションに明示的に記載されていないHTTP動詞を含むサーバーへの要求はすべて拒否されます。構成ファイル内の1は真、0は偽を表します。
UseAllowExtensions = @use_allow_extensions;boolean@	UseAllowExtension機能をオンまたはオフにできます。オンにした(1に設定した)場合、UrlScan.iniファイルのAllowExtensionセクションに明示的に記載されていないファイル拡張子を含むサーバーへの要求はすべて拒否されます。構成ファイル内の1は真、0は偽を表します。
UseFastPathReject = @use_fast_path_reject;boolean@	UseFastPathReject機能をオンまたはオフにできます。オンにした(1に設定した)場合、RejectResponseUrlオプションは無視され、URLが拒否された場合は短い404応答がクライアントに返されます。構成ファイル内の1は真、0は偽を表します。
VerifyNormalization = @verify_normalization;boolean@	UrlScan.iniでスキャンされるすべてのURLの正規化をオンまたはオフにできます。オンにした(1に設定した)場合、URLは正規化されてからスキャンされます。構成ファイル内の1は真、0は偽を表します。

5. [AllowExtensions] セクションの定義 - 新しいブロックの開始によるブロックの終了

これでUrlScan.iniファイルの [Options] セクションのすべてのオプションの定義が終わったので、次のセクション [AllowExtensions] の定義を始めます。[Options] セクションを開始するときに、2つのブロックを開始したことを思い出してください。これらは、[Options] セクションのタイトルとその内容という2つのレベルの情報を処理するためでした。

[AllowExtensions] セクションの定義を開始するためには、CMLブロックを閉じて前のセクションを終了する必要があります。CMLでブロックを閉じるには、"]"タグで明示的に閉じる方法と、より上位のレベル(小さい番号)または同じレベルの新しいブロックを開始する方法があります。この作業では、[Options] セクションのブロックを開始したときと同様に、新しいレベル1ブロックを開くことによって、[AllowExtensions] のための新しいブロックを開始します。これにより、[Options] セクションによって開始されたブロックは自動的に終了されます。

新しいブロックを開始して [AllowExtensions] セクションを定義するには、次の手順を実行します。

1. [Options] セクションの最後の行の後に次の内容を入力して、[AllowExtensions] セクション用の新しいブロックを開始します。

   ```
   @1[;optional;ordered-lines@
   [AllowExtensions]
   @2[;unordered-lines@
   ```

   次の表に、新しい2レベルのブロックを開始することで前のブロックが終了されることを示します。

   [AllowExtensions] セクション用の新しいブロックの開始

CMLタグ	説明
@1 [;optional;ordered-lines@	番号1は、新しいレベル1ブロックを開始します。これはレベル1のブロックなので、前のブロック([Options] セクションのキーと値のペアのためのレベル2ブロック)よりもレベルが高く、その前のレベル1のブロックと同じレベルなので、前の2つのブロックは終了されます。 ブロックはブロック終了タグによって明示的に終了することもできます。次に例を示します。 @2]@ [新しいブロックを開始するCMLブロックタグ。

[AllowExtensions] セクション用の新しいブロックの開始 (続き)

CMLタグ	説明
	optional このブロック全体が省略可能で、構成ファイルに存在しなくてもよいことを示します。 ordered-lines このタグの後にくるもの (文字列 [AllowExtensions]) は、ネイティブ UrlScan.ini構成ファイルで最初に来る必要があることを示します。言い換えると、ネイティブファイルですべてのオプションを記載してからタイトルを置くことはできません。[AllowExtensions] が先に来る必要があります。CMLでは、ordered-line要素によってこの順序が決まります。
[AllowExtensions]	ネイティブ構成ファイルのセクションを示すリテラル文字列。
@2[;unordered-lines@	番号2は、ブロックの第2レベルを設定します。 [新しいブロックを開始するCMLブロック記号。 unordered-lines ブロック内で [AllowExtensions] の後に来る行は、構成ファイル内で任意の順序を取れることを示します。すなわち、[AllowExtensions] セクションのすべてのキーと値のペアは任意の順序で現れることができます。

2. 次に、UrlScan.iniファイルの [AllowExtensions] セクションはユーザーが入力したファイル拡張子の任意のリストを含むことができるので、CMLのループタグとループターゲットタグを使用して、このセクションの情報を1つずつ読み取るようにパーサーに指示します。前のステップの最後の`@2[;unordered-lines@`テキストのすぐ後に、次のテキストを入力します。

```
@*allow_extension;unordered-string-set@
.@.@
```

次の表に、CMLのループタグとループターゲットタグの動作を示します。

CMLのループタグとループターゲットタグ

CMLタグ	説明
@*allow_ extension;unordered-string-set@	構文 @<レベル><タグタイプ><名前>;<データ型>;<オプション>@ ループタグ (*) は、[AllowExtensions] セクション内に記載された順序なしの文字列セットをループして読み取ります。 allow_extension 名前空間内で文字列が記録されるパスを定義する文字列。

CMLのループタグとループターゲットタグ (続き)

CMLタグ	説明
	unordered-string-set 文字列のリストが特定の順序でなくてもよいことを示します。
.@.@	最初の (.) このセクションで、パーサーが読み取るこの順序なしの文字列セットは、[AllowExtensions] セクションに記載されたファイル拡張子のリストです。ファイル拡張子の先頭は (.) 文字です。 @.@ ループターゲットタグ (.) は、このリスト内のピリオド文字で始まるものをすべて読み取るようにパーサーに指示します。

3. ファイルを保存します。

6. [DenyExtensions] セクションの定義

次に、UrlScan.iniファイルの [DenyExtensions] セクションを、[AllowExtensions] セクションと同じ方法で定義します。新しいレベル1ブロックを開始します。これにより、前の [AllowExtensions] セクションのブロックは終了されます。次に、レベル2のブロックを開始し、UrlScan.iniによってブロックするすべてのファイル拡張子の順序なしのリストを読み取るようにパーサーに指示します。ファイル拡張子の先頭は (.) です。

[DenyExtensions] セクション用のCMLは次のようになります。

```
@1[;optional;ordered-lines@
[DenyExtensions]
@2[;unordered-lines@
@*deny_extension;unordered-string-set@
.@.@
```

7. [AllowVerbs] および [DenyVerbs] セクションの定義

UrlScan.iniファイルの次の2つのセクションは、前のセクションで [DenyExtensions] に使用したのと同じCMLを使用します。第1レベルのブロックを開始して前のブロックを閉じます。またこれにより、次のテキストが順序ありの行として解析されます。

次に、第2レベルのブロックを開始します。これにより、その後の順序なしの文字列のリストが読み込まれます。これは動詞のリストです。これら2つのセクションでは、Webサイトに対するアクセスを許可する動詞のリストと、アクセスを拒否する動詞のリストを読み取るようにパーサーに指示します。

これらのセクションに対するCMLは次のとおりです。

```
@1[;optional;ordered-lines@
[AllowVerbs]
@2[;unordered-lines@
@*allow_verb;unordered-string-set@
@.@

@1[;optional;ordered-lines@
[DenyVerbs]
@2[;unordered-lines@
@*deny_verb;unordered-string-set@
@.@
```

8. [DenyHeaders] セクションの定義

次に、UrlScan.iniファイルの [DenyHeaders] セクションを定義します。ここでは、特定のHTTP要求ヘッダーを拒否するようにIISを構成できます。

このセクションでは、前のセクションと同様に、文字列用の2つのブロックを開始します。ただしここでは、UrlScan.iniファイル内に記載されたHTTPヘッダーのリストを、CMLのシーケンス区切り文字を使用してコロンで区切ります。HTTP要求ヘッダーにはコロン (:) が含まれるため、シーケンス区切り文字を使用して、セクションの各行を読み取る際に、コロン (:) が見つかったら次のエントリに進むようにパーサーに指示する必要があります。

たとえば、UrlScan.iniファイルに記載された拒否するHTTPヘッダーのリストは次のようになります。

Translate:
If:
Lock-Token:

構成ファイルに記載されているヘッダー要求はコロン (:) で終わるため、(:) をエントリの末尾として認識するようにパーサーに指示する必要があります。

1. [DenyHeaders] セクションを定義するには、[DenyVerbs] セクションの最後の行の後に次のテキストを入力して、[DenyHeaders] セクション用の新しいブロックを開始します。

@1[;optional;ordered-lines@
[DenyHeaders]
@2[;unordered-lines@

前のセクションと同様、これらのタグにより、順序ありの行として読み取られるレベル1のブロックが開始され、次に順序なしの行として読み取られるレベル2のブロックが開始されます。

2. その後、次のCMLループタグとループターゲットタグを入力して、ヘッダー要求のリストを読み取るようにパーサーに指示します。

```
@*deny_header;unordered-string-set;;sequence-delimiter=":"@
@.@:
```

次の表に、これら2つのタグの構文を示します。

[DenyHeaders] セクション用のループタグとループターゲットタグ

CMLタグ	説明
@*deny_ header;unordered-string- set;;sequence- delimiter=":"@	* 文字列のリストを読み取るループCMLタグを示します。 deny_header 名前空間内で文字列が記録されるパスを定義する文字列リテラル。 unordered-string-set 文字列のリストの順序が任意であることを示します。 ; 1番目のセミコロンは、タグの2つのセクションを区切ります。 ; 2番目のセミコロンは、次のコロン (:) シーケンス区切り文字が範囲と解釈されないようにするためのものです。 sequence-delimiter=":" パーサーに対して、コロン (:) を文字列の一部として読み取り、その後に次のエントリに移動するように指示します。
@.@	ループターゲットタグは、これらの値をdeny_header名前空間の場所に記録するようにパーサーに指示します。例: /security/deny_header
:	最後のコロン (:) は、このリストの各アイテムの後ろにコロンが付くことをパーサーに通知します。すなわち、この文字は拒否するヘッダーのエントリの一部として記録されます。

3. ファイルを保存します。

9. [DenyURLSequences] セクションの定義

[DenyUrlSequence] の定義は [DenyHeader] セクションと同様に行います。順序ありと順序なしの文字列として読み取られる2つのブロックを開始します。ただし、このセクションでは、テンプレート内のURLシーケンスのリストをフィールド区切り文字で区切ります。ここで使用するフィールド区切り文字は行末要素であり、行末が見つかったらエントリの読み取りを終了するようにパーサーに指示します。

[DenyUrlSequence] セクションを定義するには、次の手順を実行します。

1. [DenyUrlSequence] セクションの最後の行の後に次のテキストを入力して、[DenyUrlSequence] セクション用の新しいブロックを開始します。

   ```
   @1[;optional;ordered-lines@
   [DenyUrlSequence]
   @2[;unordered-lines@
   ```

   前のセクションと同様、これらのタグにより、順序ありの行として読み取られるレベル1のブロックが開始され、次に順序なしの行として読み取られるレベル2のブロックが開始されます。

2. その後、次のCMLループタグとループターゲットタグを入力して、拒否するURLシーケンスのリストを読み取るようにパーサーに指示します。

   ```
   @*deny_url_sequence;unordered-string-set;;field-delimiter-is-eol@
   @.@
   ```

   次の表に、これらのタグの構文を示します。

   [DenyUrlSequence] セクション用のループタグとループターゲットタグ

CMLタグ	説明
@*deny_url_ sequence;unordered-string- set;;field-delimiter-is-eol@	* 文字列のリストを読み取るループCMLタグを示します。 deny_url_sequence 名前空間内で文字列が記録されるパスを定義する文字列リテラル。 unordered-string-set 文字列のリストの順序が任意であることを示します。 ; 1番目のセミコロンは、タグの2つのセクションを区切ります。 ;

[DenyUrlSequence] セクション用のループタグとループターゲットタグ (続き)

CMLタグ	説明
	2番目のセミコロンで、次のフィールド区切り文字を入力します。この入力内容は範囲として解釈されません。 `field-delimiter-is-eol` 次のエントリを行末まで読み取るようにパーサーに指示します。
@.@	ループターゲットタグは、これらの値を`deny_url_sequence`名前空間の場所に記録するようにパーサーに指示します。例: /security/deny_url_sequence

3. ファイルを保存します。

10. [RequestLimits] セクションの定義

[RequestsLimits] の定義は、[DenyUrlSequence] セクションとよく似た方法で行います。順序ありと順序なしの文字列として読み取られる2つのブロックを開始します。ただし、このセクションでは、2つのブロックを開始した後で、CML置換タグを使用して3つのキーと値のペアを定義します。

[RequestsLimits] セクションを定義するには、次の手順を実行します。

1. [RequestsLimits] セクションの最後の行の後に次のテキストを入力して、[RequestsLimits] セクション用の新しいブロックを開始します。

   ```
   @1[;optional;ordered-lines@
   [RequestsLimits]
   @2[;unordered-lines@
   ```

   前のセクションと同様、これらのタグにより、順序ありの行として読み取られるレベル1のブロックが開始され、次に順序なしの行として読み取られるレベル2のブロックが開始されます。すでに述べたように、新しい第1レベルのブロックを開始することにより、前の [DenyUrlSequence] セクションの第2レベルのブロックは終了されます。

2. 次のCML置換タグを入力して、[RequestsLimits] セクションの3つのキーと値の3つのキーと値のペアを定義します。

   ```
   MaxAllowedContentLength = @max_allowed_content_length;int@
   MaxUrl = @max_url;int@
   MaxQueryString = @max_query_string;int@
   @1]@
   ```

次の表に、これらのタグの構文を示します。

[DenyUrlSequence] セクション用のループタグとループターゲットタグ

CMLタグ	説明
MaxAllowedContentLength = @max_allowed_content_length;int@	MaxAllowedContentLength 構成ファイルからの要求制限パラメーター文字列。 max_allowed_content_length 名前空間内で値が記録されるパスを定義する文字列リテラル。 int 記録する値が整数であることを示します。
MaxUrl = @max_url;int@	MaxUrl 構成ファイルからの要求制限パラメーター文字列。 max_url 名前空間内で値が記録されるパスを定義する文字列リテラル。 int 記録する値が整数であることを示します。
MaxQueryString = @max_query_string;int@	MaxQueryString 構成ファイルからの要求制限パラメーター文字列。 max_query_string 名前空間内で値が記録されるパスを定義する文字列リテラル。 int 記録する値が整数であることを示します。
@1]@	このレベル1ブロックタグは、ブロックを終了させます。

3. ファイルを保存します。

11. アプリケーション構成へのテンプレートの追加

UrlScan.iniに対するCMLテンプレートを作成し、url_scan_ini.tplという名前で保存したら、次の作業を実行します。

- テンプレートをSAクライアントにインポートし、CML構文を検証します。「テンプレートファイルのインポートと検証」(280ページ)を参照してください。
- テンプレートをアプリケーション構成に追加します。「アプリケーション構成に対するテンプレートの追加または削除」(282ページ)を参照してください。
- アプリケーション構成をサーバーにアタッチします。「サーバーまたはデバイスグループへのアプリケーション構成のアタッチ」(286ページ)を参照してください。
- テンプレートをテストするため、変更を行ってサーバーにプッシュします。「アプリケーション構成のプッシュ」(290ページ)を参照してください。

これらの手順の説明は、「CMLチュートリアル1 - 単純なWebアプリケーションサーバーに対するアプリケーション構成の作成」(327ページ)にあります。

UrlScan.iniファイルの例

UrlScan.iniファイルの例を次に示します。

```
[Options]
UseAllowVerbs=1                ; If 1, use [AllowVerbs] section, else use the
                               ; [DenyVerbs] section.   The default is 1.

UseAllowExtensions=0           ; If 1, use [AllowExtensions] section, else
                               ; use the [DenyExtensions] section.The
                               ; default is 0.

NormalizeUrlBeforeScan=1       ; If 1, canonicalize URL before processing.
                               ; The default is 1.Note that setting this
                               ; to 0 will make checks based on extensions,
                               ; and the URL unreliable and is therefore not
                               ; recommend other than for testing.

VerifyNormalization=1          ; If 1, canonicalize URL twice and reject
                               ; request if a change occurs.The default
                               ; is 1.
```

```
AllowHighBitCharacters=0        ; If 1, allow high bit (ie.UTF8 or MBCS)
                                ; characters in URL.The default is 0.

AllowDotInPath=0                ; If 1, allow dots that are not file
                                ; extensions.The default is 0.Note that
                                ; setting this property to 1 will make checks
                                ; based on extensions unreliable and is
                                ; therefore not recommended other than for
                                ; testing.

RemoveServerHeader=1            ; If 1, remove the 'Server' header from
                                ; response.The default is 0.

EnableLogging=1                 ; If 1, log UrlScan activity.The
                                ; default is 1.Changes to this property
                                ; will not take effect until UrlScan is
                                ; restarted.

PerProcessLogging=0             ; This property is deprecated for UrlScan
                                ; 3.0 and later.UrlScan 3.0 and later can
                                ; safely log output from multiple processes
                                ; to the same log file.Changes to this
                                ; property will not take effect until
                                ; UrlScan is restarted.

AllowLateScanning=0             ; If 1, then UrlScan will load as a low
                                ; priority filter.The default is 0.Note
                                ; that this setting should only be used in
                                ; the case where there another installed
                                ; filter is modifying the URL and you wish
```

```
                        ; to have UrlScan apply its rules to the
                        ; rewritten URL.Changes to this property
                        ; will not take effect until UrlScan is
                        ; restarted.

PerDayLogging=1         ; If 1, UrlScan will produce a new log each
                        ; day with activity in the form
                        ; 'UrlScan.010101.log'.If 0, UrlScan will
                        ; log activity to urlscan.log.The default
                        ; is 1.Changes to this setting will not
                        ; take effect until UrlScan is restarted.

UseFastPathReject=0     ; If 1, then UrlScan will not use the
                        ; RejectResponseUrl.On IIS versions less
                        ; than 6.0, this will also prevent IIS
                        ; from writing rejected requests to the
                        ; W3SVC log.UrlScan will log rejected
                        ; requests regardless of this setting.The
                        ; default is 0.

LogLongUrls=0           ; This property is deprecated for UrlScan 3.0
                        ; and later.UrlScan 3.0 and later will
                        ; always include the complete URL in its log
                        ; file.

UnescapeQueryString=1   ; If 1, UrlScan will perform two passes on
                        ; each query string scan, once with the raw
                        ; query string and once after unescaping it.
                        ; If 0, UrlScan will only look at the raw
                        ; query string as sent by the client.The
                        ; default is 1.Note that if this property is
```

```
                                ; set to 0, then checks based on the query
                                ; string will be unreliable.

RejectResponseUrl=

LoggingDirectory=Logs

[AllowVerbs]

;
; The verbs (aka HTTP methods) listed here are those commonly
; processed by a typical IIS server.
;
; Note that these entries are effective if "UseAllowVerbs=1"
; is set in the [Options] section above.
;

GET
HEAD
POST

[DenyVerbs]

;
; The verbs (aka HTTP methods) listed here are used for publishing
; content to an IIS server via WebDAV.
;
; Note that these entries are effective if "UseAllowVerbs=0"
; is set in the [Options] section above.
;
```

```
PROPFIND
PROPPATCH
MKCOL
DELETE
PUT
COPY
MOVE
LOCK
UNLOCK
OPTIONS
SEARCH

[DenyHeaders]

;
; The following request headers alter processing of a
; request by causing the server to process the request
; as if it were intended to be a WebDAV request, instead
; of a request to retrieve a resource.
;

Translate:
If:
Lock-Token:
Transfer-Encoding:

[AllowExtensions]

;
; Extensions listed here are commonly used on a typical IIS server.
;
```

```
; Note that these entries are effective if "UseAllowExtensions=1"
; is set in the [Options] section above.
;

.htm
.html
.txt
.png
.png
.png

[DenyExtensions]

;
; Extensions listed here either run code directly on the server,
; are processed as scripts, or are static files that are
; generally not intended to be served out.
;
; Note that these entries are effective if "UseAllowExtensions=0"
; is set in the [Options] section above.
;
; Also note that ASP scripts are denied with the below
; settings.If you wish to enable ASP, remove the
; following extensions from this list:
;    .asp
;    .cer
;    .cdx
;    .asa
;

; Deny executables that could run on the server
```

```
.exe
.bat
.cmd
.com

; Deny infrequently used scripts
.htw     ; Maps to webhits.dll, part of Index Server
.ida     ; Maps to idq.dll, part of Index Server
.idq     ; Maps to idq.dll, part of Index Server
.htr     ; Maps to ism.dll, a legacy administrative tool
.idc     ; Maps to httpodbc.dll, a legacy database access tool
.shtm    ; Maps to ssinc.dll, for Server Side Includes
.shtml   ; Maps to ssinc.dll, for Server Side Includes
.stm     ; Maps to ssinc.dll, for Server Side Includes
.printer ; Maps to msw3prt.dll, for Internet Printing Services

; Deny various static files
.ini     ; Configuration files
.log     ; Log files
.pol     ; Policy files
.dat     ; Configuration files
.config  ; Configuration files

[DenyUrlSequences]
;
; If any character sequences listed here appear in the URL for
; any request, that request will be rejected.
;

..; Don't allow directory traversals
./; Don't allow trailing dot on a directory name
```

```
\   ; Don't allow backslashes in URL
:    ; Don't allow alternate stream access
%   ; Don't allow escaping after normalization
&   ; Don't allow multiple CGI processes to run on a single request
```

完成したurl_scan_ini.tpl CMLテンプレート

完成したurl_Scan_ini.tplテンプレートを次に示します。

```
@#############################################
# \system32\inetsrv\urlscan.ini (Windows)    #
# Version 1.0                                #
# Joe Author (joe_author@your_company.com)   #
#############################################@

@!namespace=/security/@
@!filename-key="/test";filename-default="/c/UrlScan.ini"@
@!optional-whitespace@
@!boolean-yes-format="1";boolean-no-format="0"@
@!line-comment-is-semicolon@
@!unordered-lines@

@#########################################
# Begin data                             #
#########################################@

@1[;optional;ordered-lines@
[Options]
@2[;unordered-lines@
AllowDotInPath = @allow_dot_in_path;boolean@
AllowHighBitCharacters = @allow_high_bit_characters;boolean@
AllowLateScanning = @allow_late_scanning;boolean@
AlternateServerName = @alternate_servername@
EnableLogging = @enable_logging;boolean@
LoggingDirectory = @logging_directory;dir@
LogLongURLs = @log_long_urls;boolean@
NormalizeUrlBeforeScan = @normalize_url_before_scan;boolean@
PerDayLogging = @per_day_logging;boolean@
PerProcessLogging = @per_process_logging;boolean@
RejectResponseUrl =
@reject_response_url;string;r”(HTTP_URLSCAN_STATUS_HEADER)|(HTTP_URLSCAN
_ORIGINAL_VERB)|(HTTP_URLSCAN_ORIGINAL_URL)”;optional@
RemoveServerHeader = @remove_server_header;boolean@
```

```
UnescapeQueryString = @unescape_query_string;boolean@
UseAllowVerbs = @use_allow_verbs;boolean@
UseAllowExtensions = @use_allow_extensions;boolean@
UseFastPathReject = @use_fast_path_reject;boolean@
VerifyNormalization = @verify_normalization;boolean@

@1[;optional;ordered-lines@
[AllowExtensions]
@2[;unordered-lines@
@*allow_extension;unordered-string-set@
.@.@

@1[;optional;ordered-lines@
[DenyExtensions]
@2[;unordered-lines@
@*deny_extension;unordered-string-set@
.@.@

@1[;optional;ordered-lines@
[AllowVerbs]
@2[;unordered-lines@
@*allow_verb;unordered-string-set@
@.@

@1[;optional;ordered-lines@
[DenyVerbs]
@2[;unordered-lines@
@*deny_verb;unordered-string-set@
@.@

@1[;optional;ordered-lines@
[DenyHeaders]
@2[;unordered-lines@
@*deny_header;unordered-string-set;;sequence-delimiter=":"@
@.@:

@1[;optional;ordered-lines@
[DenyURLSequences]
@2[;unordered-lines@
@*deny_url_sequence;unordered-string-set;;field-delimiter-is-eol@
@.@

@1[;optional;ordered-lines@
[RequestLimits]
@2[;unordered-lines@
MaxAllowedContentLength = @max_allowed_content_length;int@
MaxUrl = @max_url;int@
MaxQueryString = @max_query_string;int@
@1]@
```

CML入門

この項では、**CML** (構成モデリング言語)の概要を紹介します。CMLの詳細については、「CMLリファレンス」(375ページ)を参照してください。詳細については、「CMLチュートリアル1 - 単純なWebアプリケーションサーバーに対するアプリケーション構成の作成」(327ページ)および「CMLチュートリアル2 - Webサーバー構成ファイルのテンプレートの作成」(338ページ)も参照してください。

SAは、**構成テンプレート**を作成することによって構成ファイルを管理します。構成テンプレートは次の用途に用いられます。

- 構成ファイルの構文のモデル化。
- 構成ファイルから値を抽出して、**値セット**としてSAデータベースに保存。これらの値を保存したら、SAクライアントを使用して値を管理できます。
- 値セットからの新しい構成ファイルの作成。
- 新しい構成ファイルのサーバーへのプッシュ。
- サーバー上での構成ファイルの監査によるコンプライアンスの検証。

用語

- **構成ファイル** - SAによって管理されるファイル。
- **値セット** - サーバーごとに異なる可能性がある構成ファイル内のデータ値。値セット内の値は、SAデータベースに「キー=値」の形式で保存されます。
- **名前空間** - 値セットをSAデータベースに保存するための構造。
- **構成テンプレート** - CMLで書かれた構成ファイルのモデル。
- **構成モデリング言語 (CML)** - 構成テンプレート内で構成ファイルをモデル化するために使用される命令タグのセット。
- **命令**または**タグ** - 実行するアクションを定義するキーワードと文字。すべての命令は、先頭と末尾が“@”文字です。「命令」と「タグ」という用語は同じ意味で用いられます。
- **アプリケーション構成オブジェクト** - 構成テンプレートのコンテナーで、値セットとの組み合わせで構成ファイルを生成します。生成された構成ファイルは管理対象サーバーにプッシュされます。これには、プッシュ操作の過程で実行されるスクリプトを含めることもできます。

CMLの基本概念

CML (構成モデリング言語) は、構成ファイルの構文をモデル化します。CMLを使用して、ターゲット構成ファイルのモデルである**構成テンプレート**を作成します。このための一般的な最善の手段は、ターゲット構成ファイルのドキュメントを入手して、構成ファイル内の有効な値と範囲を調べ、それをモデル化する最善の方法を判断することです。

構成テンプレートを最大限に機能させるには、構成ファイル全体をCMLでモデル化するのが最善です。ただし、構成ファイルの一部の行だけを対象としたCMLを作成することも可能です。これは部分テンプレートと呼ばれ、「部分テンプレート」(374ページ)で説明しています。

必須のCML命令タグ

次の3つのCML命令 (CMLタグとも呼ぶ) は、すべての構成テンプレートに必須です。

- `namespace`は、値セットのデータがSAデータベースに記録されるキーを定義します。「namespaceタグ」(363ページ)を参照してください。
- `filename-key`は、ターゲット構成ファイル名がSAデータベースに記録されるキーを定義します。「filename-keyタグ」(364ページ)を参照してください。
- `filename-default`は、ターゲット構成ファイルのデフォルト名を指定します。「filename-defaultタグ」(365ページ)を参照してください。

その他のタグはすべてオプションであり、特定の構成ファイルの内容をモデル化するために使用されます。CMLの詳細については、「CMLリファレンス」(375ページ)を参照してください。これら3つの必須タグの詳細については、「CMLのグローバルオプション属性」(401ページ)を参照してください。

namespaceタグ

`namespace`命令は、値セットがSAデータベースに記録されるキーを定義します。

構文:

```
@!namespace=<パス>@
```

ここで、<パス> はディレクトリパスと同様の形式の文字列で、SAデータベースで値セットが記録されるキーを定義します。

例:

```
@!namespace=/example/namespace/@
```

この例は、このテンプレートの他のすべての命令のベース名前空間を“/example/namespace/”に設定します。すなわち、値セットのすべての値は、別の指定がない限り、キー“/example/namespace/”に記録されます。

Description:

namespace命令は、値セットの値のキー/値マッピングのキーを指定します。これは任意の文字列で、数値以外なら何でもかまいません。この命令は、値セットの値がSAデータベースで記録されるキーを決定します。構成ファイルの置換タグ(およびその他のタグ)は、namespaceキーを使用してSAデータベースから値を取得します。

namespaceの値は絶対形式である必要があります。これ以降の値は、相対パス名でも絶対パス名でもかまいません。

- 絶対名は、“/” 文字で始まるフルパス名です。これらの名前は、namespace命令で指定された値を使用しません。たとえば、次のタグがあったとします。

  `@/testval@`

  このタグに一致する値は、値セットでキー “/testval” の下に記録されます。

- 相対名は、namespace命令で指定された値の後に追加されます。たとえば、次のタグがあったとします。

  `@testval@`

  このタグに一致する値は、値セットでキー “/example/namespace/testval” の下に記録されます。

なお、ループに含まれる名前付きタグは、名前の前にドット “.” を付ける必要があります。このタグの名前空間は、現在のループの名前空間の後に追加されます。次に例を示します。`@.testval@`

filename-keyタグ

filename-key命令は、ターゲット構成ファイル名がSAデータベースに記録されるキーを指定します。

構文:

`@!filename-key=<パス>@’`

ここで、<パス> はディレクトリパスと同様の形式の任意の文字列で、SAデータベースで構成ファイル名が記録されるキーを定義します。

例:

`@!filename-key=/files/example@`

この例は、ファイル名がSAデータベースでキー “/files/example” に記録されることを指定します。この場合、<パス> の値の先頭が “/” 文字なので、絶対パスを表します。

`@!filename-key=files/example@`

この例は、値の先頭が“/”文字でないので、相対パスを指定しています。これを前のnamespaceの例と組み合わせると、構成ファイル名は “/example/namespace/files/example” に記録されます。

Description:

filename-key命令は、SAデータベースでターゲット構成ファイル名を記録するために使用されるキーを指定します。たとえば、SAクライアントでテンプレートの[ファイル名]フィールドを設定した場合、そのファイル名は値セットでキー “/files/example” の下に記録されます。filename-keyはファイルシステムのパスではなく、単にパスと似た方法で書かれるキーです。

これは、インストール前およびインストール後スクリプトで、構成ファイルがターゲットサーバーにプッシュされる前または後にそのファイル名を知る必要がある場合に便利です。たとえば、インストール後スクリプトでプッシュ後に構成ファイルの末尾に行を追加する必要がある場合、スクリプトは次のようになります。

```
echo “#end of the file” >> @/files/example@
```

filename-defaultタグ

構文:

```
@!filename-default=<ファイル>@
```

ここで、<ファイル> は、管理対象サーバーのファイルシステムでのターゲット構成ファイルのディレクトリパスとファイル名です。これは、生成された構成ファイルが管理対象サーバーにプッシュされる場所です。

例:

```
@!filename-default=/etc/hosts@
```

この例は、ターゲット構成ファイルが/etc/hostsであることを指定します。この値は、filename-key命令で指定されたキーによってSAデータベースに記録されます。

Description:

filename-default命令は、ターゲット構成ファイルの標準のファイルシステムパスを指定します。この値は、デフォルトのファイル名とディレクトリであり、filename-key命令で指定されたキーによって記録されます。これはSAクライアントの[ファイル名]フィールドのデフォルト値です。

タグの1行への結合

複数の命令タグを1つの命令に結合するには、命令の間をセミコロンで区切り、先頭に感嘆符を1個だけ付けます。次に例を示します。

```
@!namespace=/example/namespace/;filename-key=/files/example;
filename-default=/etc/example@
```

こうすれば、ファイルに1行を入れるだけで有効なCMLテンプレートになるので (もちろん他のCMLタグもすべて正しい形式であることが条件ですが) 便利です。

使用法1 - 単純なキー=値の構成ファイル

最も単純な種類の構成ファイルは、1つ以上のキー=値のエントリから構成されます。次に例を示します。

```
Port = 1280
IPAddress = 192.168.0.1
ServerName = server01
```

この構成ファイルでは、タイプと説明を容易に判別できます。

置換命令の使用

この構成ファイルのテンプレートを作成するには、CMLタグを使用して、値が存在する場所と、値が値セットの名前空間に記録される場所を指定します。このためには、置換タグを使用します。

置換タグはいくつかのフィールドから構成され、CML言語のすべてのタグと同様に、先頭と末尾は“@”文字で、フィールドはセミコロンで区切ります。形式は次のようになります。

```
@ <名前> ; [タイプ] ; [範囲] ; [オプション] ; [オプション] … @
```

すべてのフィールドの中で、<名前>フィールドだけが必須です。したがって、上の構成ファイルを表現する最も基本的なCMLは次のようになります。

```
@!namespace=/example/namespace/@
@!filename-key=/files/example@
@!filename-default=/etc/example@
Port = @port@
IPAddress = @ipaddress@
ServerName = @servername@
```

俺は、ポート番号の値がSAデータベースのキー “/example/namespace/port” に記録されることを指定します。IPアドレスはキー “/example/namespace/ipaddress” に、サーバー名はキー “/example/namespace/servername” に記録されます。

これは技術的には動作しますが、フィールドの検証やエラーチェックのためのさまざまな機能はいっさい利用できません。たとえば、ポートに “someport” のような無効なデータが入力されるのを防ぐことはできません。

置換命令タグの <名前> フィールド

<名前> フィールドが相対形式の場合 (先頭が "/" または "." でない場合)、現在の名前空間の後に追加され、このタグによってSAデータベースから読み取られる値を記録するキーの一部となります。

名前が絶対形式の場合 (先頭が "/" の場合)、これはキー全体を表し、値はこのキーの下に記録されます。

また、名前の先頭がドット "." の場合、これが含まれるループの名前空間の後に追加されます。わずかな例外を除いて、ループ内のすべてのタグは先頭にドット "." を付けます。

置換命令タグの <タイプ> フィールド

<タイプ> フィールドを使用すると、既知のタイプに基づく定義済みの範囲とエラーのチェックを値に適用できます。タイプの一覧については、「CMLタイプ属性」(392ページ)を参照してください。この構成ファイルでは、定義済みのタイプ "port"、"ip"、"hostname" を各エントリに次のように使用します。

```
@!namespace=/example/namespace/@
@!filename-key=/files/example@
@!filename-default=/etc/example@
Port = @port;port@
IPAddress = @ipaddress;ip@
ServerName = @servername;hostname@
```

これらのタイプを追加すると、値が制限され、検証とエラーチェックが行われます。

また、前に "ordered-" または "unordered-" を付け、後ろに "-set" または "-list" を付けることにより、繰り返し値のシーケンスを置換タグで表現できます。これについては次の例で詳しく説明します。

このフィールドのデフォルトは "string" で、任意の値に一致します。

置換命令タグの <範囲> フィールド

<範囲> フィールドを使用すると、値の許容範囲を設定できます。整数範囲または文字列範囲を設定できます。整数範囲は整数だけから成る任意のタイプに使用でき、文字列範囲はその他のすべてのタイプに使用できます。

範囲を論理ORで結合するには、カンマ "," を使用します。範囲を論理ANDで結合するには、アンパサンド文字 "&" を使用します。"!" 文字は範囲を否定します。

指定した範囲は、構成ファイルの読み取りと、SAクライアントからの値の入力時に使用されます。テンプレートで設定した範囲外の値が構成ファイルにある場合、ファイルの解析中にエラーが報告されます。構成ファイルのドキュメントに基づいて、有効範囲を指定します。

整数範囲

整数範囲では、<および=記号だけを使用して、「より小さい」または「より大きい」範囲を指定します。比較に使用する数値の位置を次のように指定します。

整数範囲の指定

範囲の条件	使用する記号
より大きい	n<
以上	n<=
より小さい	<n
以下	<=n
等しい	=n

たとえば、構成ファイル内のポートが1024から2048まで (両端含む) でなければならない場合は、次のような範囲をタグに追加します。

```
Port = @port;port;1024<=&<=2048@
```

文字列範囲

文字列範囲は、引用符で囲んだ有効な文字列のリストと、r" で始まり引用符で終わる正規表現のリストです。たとえば、ServerNameフィールドの値が "server" という語で始まらなければならない場合は、servernameタグに次のような範囲を追加します。

```
ServerName = @servername;hostname;r"server.*"@
```

置換命令タグの [オプション] フィールド

タグには必要な数だけのオプションを追加できます。3つ目のセミコロンより後ろのすべてのものはオプションと見なされ、オプションの間はセミコロンで区切ります。

たとえば、構成ファイルのIPAddress行がオプションで、この行がなくても構成ファイルは有効であり、IPアドレスの末尾にスラッシュを付けられる場合は、次のようなオプションを追加します。

```
IPAddress = @ipaddress;ip;;optional;delimiter="/"@
```

これは次のエントリに一致します。

```
IPAddress = 192.168.0.1
```

また、これは次のエントリにも一致します。

```
IPAddress = 192.168.0.2/
```

範囲フィールドが空であることに注意してください。あるフィールドをデフォルトのままにする場合でも、その後のフィールドを指定する場合には、そのフィールドを表現することが必要です。たとえば、次の2行はどちらも有効です。

```
@ipaddress@
```

```
@ipaddress;;;optional@
```

ただし、次の行は有効ではありません。フィールド “optional” がオプションでなく<タイプ> フィールドと見なされるため、エラーが発生します。

```
@ipaddress;optional@
```

最終的なCMLテンプレート

タイプ、オプション、範囲を設定すると、CMLテンプレートは次のようになります。

```
@!namespace=/example/namespace/@
@!filename-key=/files/example@
@!filename-default=/etc/example@
Port = @port;port;1024<=&<=2048@
IPAddress = @ipaddress;ip;;optional;delimiter="/"@
ServerName = @servername;hostname;r"server.*"@
```

結果の値セット

上の構成テンプレートを使用して上の例のターゲット構成ファイルを読み込むと、SAデータベースに次の値セットが記録されます。

```
/example/namespace/port = 1280
/example/namespace/ipaddress = 192.168.0.1
/example/namespace/servername = server01
```

このように、名前空間内のキーは、テンプレートの名前空間 (/example/namespace) と、各タグの名前 (すべて相対名を使用しているため) の組み合わせになります。

使用法2 - 構成ファイル内の反復する値

構成ファイルが単に値のリストから構成される場合があります。たとえば、あるディレクトリへの書き込みアクセス権を持つユーザー名のリストだけから成るファイルです。このファイルの形式は、次のようになります。

```
admin;
user1;
user2;
```

このファイル内の反復する行を処理するには、前の例で説明した置換タグでは不十分です。この構成ファイルに一致する最も基本的なCMLは、次のループ命令です。

```
@!namespace=/wuserlist/namespace/@
@!filename-key=/wuserlistfile/example@
@!filename-default=/etc/wusers.txt@
@*users@
@.@
```

ループ命令タグの使用

ループ命令は、値のセットが構成ファイルに複数回出現する可能性がある場合に使用されます。ループ命令のデフォルトの動作は、その直後の1行のCMLをループ処理することですが、複数の行または1行の中でループするように変更できます。

形式は次のようになります。

```
@ [グループレベル] * <名前> ; [ "ordered" | "unordered" ] - [タイプ] - ["set" | "list" ] ; [範囲] ; [オプション] ; [オプション] … @
```

このように、前の項の置換タグと共通の部分と、異なる部分があります。次に、ループタグの各オプションについて説明します。

<グループレベル> フィールド

この例の構成ファイルではグループレベルは重要でないので、詳しくは後で説明します。現時点では、単にループするセクション (単一行か複数行か) を指定するために用いられると理解しておいてください。

この例では、グループレベルは空白になっています。これは、このループタグがその直後の1行のCMLだけを反復することを示します。

命令タイプ指定子フィールド、*

“*”は命令タイプ指定子です。これは、この命令のタイプ、すなわちこの場合はループ命令を表します。置換命令は最も頻繁に用いられるため、デフォルトの命令になっています。置換命令には命令タイプ指定子はありません。

<名前> フィールド

<名前> フィールドのルールは、置換タグの場合と同じです。復習するには、「置換命令タグの <名前> フィールド」(367ページ)を参照してください。このループに含まれる名前付きタグは、先頭に“.”(ドット) を付ける必要があります。そのタグの名前空間は、このループの名前空間の後に追加されます。同様に、このループが別のループの中に含まれる場合は、名前の前に “.” が必要です。

この例では、名前 “users” が次のように使用されています。

```
@*users@
```

[タイプ] フィールド

ループタグの [タイプ] フィールドは、置換タグの [タイプ] フィールドと2つの重要な点で異なっています(復習するには、「置換命令タグの <タイプ> フィールド」(367ページ)を参照してください)。

- orderedとunordered、setとlist

  基本タイプは置換タグと同じすべてのタイプを含みますが、これは値の反復シーケンスなので、シーケンスに関する情報を指定する必要があります。この情報を指定するために、[タイプ] フィールドのタイプ前に “ordered” または “unordered” とダッシュを付け、後ろにダッシュと “set” または “list” を付けます。
  - “ordered” を前に付けると、値の順序が保持されることを指定します。
  - “unordered” を前に付けると、値が任意の順序になることを示します。
  - “set” を後ろに付けると、値が一意でなければならないことを指定します。
  - “list” を後ろに付けると、値が繰り返されてもよいことを指定します。

    置換タグの場合はこれは省略可能ですが、ループタグの場合は、orderedまたはunorderedオプションと、setまたはlistオプションが必須です。

    この例では、データはunorderedで、“set” を後ろに付ける必要があります。アクセスリストに順序が付いていたり、重複する値があったりする意味はないからです。

- 名前空間タイプ
  このタイプは、ループタグに固有のものです。反復するセクションに複数の値が含まれる場合、名前空間タイプを使用する必要があります。

このタグのデフォルトのタイプは “unordered-string-list” です。

この例ではデフォルト値でもかまいませんが、タイプを “user” と指定する方がよいでしょう。順序なしのセットを仮定すると、結果のタグは次のようになります。

```
@*users;unordered-user-set@
```

[範囲] フィールド

範囲は、名前空間タイプ以外のすべてのタイプに関しては、置換タグの場合と同じです。名前空間タイプはそれぞれ異なる範囲を持つ複数のタグにわたって反復される可能性があるので、範囲は使用できません。

この例では “user” タイプが使用されているので、範囲を使用することができます。たとえば、この構成ファイルのドキュメントに、“root” は有効なユーザーではないと記載されている場合、root以外に対して有効となる範囲を次のように設定できます。

```
@*users;unordered-user-set;!"root";@
```

[オプション] フィールド

オプションフィールドは、置換タグのオプションフィールドと同じです。復習するには、「置換命令タグの [オプション] フィールド」(368ページ)を参照してください。

この例では、ユーザー名から ";" を削除するために、セミコロンをフィールド区切り文字に設定し、次のように反復する行に含めることによって、読み取られる値から削除することができます。

```
@*users;unordered-user-set;!"root";field-delimiter-is-semicolon@
@.@;
```

ループターゲットタグ

ループターゲットタグは次のようになります。

```
@.@
```

これは単に、ループ値の位置、すなわち結果の構成ファイルでデータが置かれる場所を指定するために用いられます。

最終的なCML

タイプ、オプション、範囲を設定すると、CMLテンプレートは次のようになります。

```
@!namespace=/wuserlist/namespace/@
@!filename-key=/wuserlistfile/example@
@!filename-default=/etc/wusers.txt@
@*users;unordered-user-set;!"root";field-delimiter-is-semicolon@
@.@;
```

結果の値セット

読み取られるすべての値は、一意のキーで値セットに記録する必要があります。シーケンスを処理する場合、CMLは、1から始まって反復のたびに1ずつ増加する一意の番号を、名前空間の後ろに追加します。

上のCMLを使用した構成ファイルの例の結果の値セットは、次のようになります。

```
/example/namespace/users/1 = admin
/example/namespace/users/2 = user1
/example/namespace/users/3 = user2
```

使用法3 - 構成ファイル内の複雑な反復する値

この例は、/etc/hostsファイルをモデル化したものです。このファイルは、次に示すように、IPアドレスのリストの後にホスト名のリストが付いたものです。

```
127.0.0.1 localhost
192.168.0.1 server1 server1.domain.com
192.168.0.2 server2 server2.domain.com
```

この例は前の例に似ていますが、この例では反復する行がより複雑です。これを考える最善の方法は、まず行の1つのインスタンスをCMLの置換命令を使用して次のようにモデル化することです。

```
@ip-addr;ip@ @sname;unordered-hostname-set@
```

この行は2つの置換命令を定義します。1つはIPアドレス、もう1つはサーバー名を対象とします。

“ip-addr” 置換タグはIPアドレスを表すので、データタイプは “ip” と指定されています。

“sname” 置換タグのタイプは “unordered-hostname-set” と指定されています。これはホスト名のリストに一致し、リストは対応するIPアドレスとともに記録されることを示します。これはループタグの動作に似ており、値は同じ方法で記録されます。

このCMLが1回の反復に用いられます。次のステップでは、これをループの中に入れます。この場合、各行で複数の値に対する反復を行うため、次のように名前空間ループを使用し、ループ内のタグ名の前に “.” を付けます。

```
@*entries;unordered-namespace-set@
@.ip-addr;ip@ @.sname;unordered-hostname-set@
```

ループタグ (@*文字で示される) はループを定義します。“unordered-hostname-set” は、データがホスト名で、ホスト名は任意の順序でよく、値は一意でなければならないことを示します。

置換命令の “ip-addr” と “sname” の前に付けられた “.” 文字は、これらがループ命令のターゲットであることを示します。

最終的なCML

上記のループおよび置換CMLを、必要な名前空間およびファイル行に追加すると、次のようになります。

```
@!namespace=/example/namespace/@
@!filename-key=/files/example@
```

```
@!filename-default=/etc/hosts@
@*entries;unordered-namespace-set@
@.ip-addr;ip@ @sname;unordered-hostname-set@
```

結果の値セット

サンプルファイルのエントリをSAクライアントで読み込んだ場合にSAデータベースに記録される値を次に示します。

```
/example/namespace/entries1/ip-addr = 127.0.0.1
/example/namespace/entries1/sname/1 = localhost
/example/namespace/entries2/ip-addr = 192.168.0.1
/example/namespace/entries2/sname/1 = server1
/example/namespace/entries2/sname/2 = server1.domain.com
/example/namespace/entries3/ip-addr = 192.168.0.2
/example/namespace/entries3/sname/1 = server2
/example/namespace/entries3/sname/2 = server2.domain.com
```

部分テンプレート

構成ファイルの全体をモデル化するのが最善ですが、部分テンプレートを使用して構成ファイルの一部だけをモデル化することも可能です。部分テンプレートを作成するには、テンプレートで読み取るために、サーバー上に構成ファイル全体のコピーが存在する必要があります。また、[形式の保持] オプションを使用して、ファイルの残りの部分を保持する必要があります。

次に示すのは単純な構成ファイルです。

```
UserName = alice
Password = pass
HomeDir = /home/alice
```

ホームディレクトリの行だけを管理するには、`@!partial-template`命令を使用して、管理する行だけをモデル化します。テンプレートは次のようになります。

```
@!namespace=/example/@
@!filename-key=/files/example@
@!filename-default=/usr/example@
@!partial-template@
HomeDir = @homedir;dir@
```

[形式の保持] 設定の詳細については、「値セットエディターでの値の設定」(235ページ)を参照してください。詳細については、「@!full-templateおよび@!partial-template属性」(402ページ)も参照してください。

CMLリファレンス

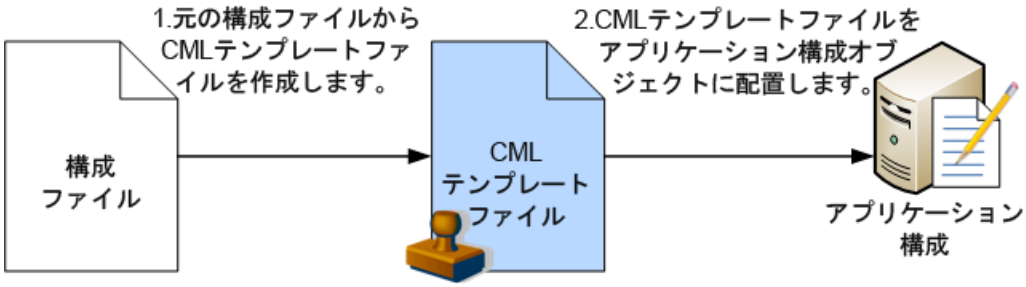
構成モデリング言語 (CML) は、**構成ファイルのテンプレート**を作成して、SAから管理できるようにするためのものです。CMLテンプレートは、構成ファイルの形式をモデル化するために作成する独立したファイルであり、構成ファイル内の可変の値をサーバーごとに異なる値に設定できます。

テンプレートファイルには、テンプレートと値のセットから実際の構成ファイルを生成するためのデータ、指示、定義が含まれています。

CMLは双方向の変換を定義します。構成ファイルからSAデータベースの値セットに値を移行する方法と、値セットのデータをテンプレートとマージして、管理対象サーバーにプッシュできる正しい形式の構成ファイルを作成する方法を指定します。

また、管理対象サーバーに構成をプッシュする際に実行するスクリプトもCMLで作成します。詳細については、「アプリケーション構成でのスクリプトの実行」(247ページ)を参照してください。

CMLテンプレート



XML構成ファイル

SAはXML構成ファイルも管理できます。XML構成テンプレートの使用法の詳細については、「XML構成ファイルの管理」(301ページ)を参照してください。

構成テンプレート

構成テンプレートは、実際の構成ファイルをテンプレート化したもので、値が変数に変換されています。SAクライアントを使用すると、テンプレートの値セットを定義し、SAデータベースに保存して、これらの値を管理対象サーバー上の実際の構成ファイルに反映させることができます。

値セットは、SAデータベースに保存されます。SAデータベースにすべての値を保存することで、構成値を1か所で管理でき、データセンターのすべてのアプリケーションを通じた構成の一貫性を保証できます。

構成ファイルのテンプレートバージョンを作成してアプリケーション構成オブジェクトに追加し、テンプレートに対する値セットを作成したら、これらの値を管理対象サーバー上の構成ファイルにプッシュできます。

CMLの概要

構成テンプレートは、一連のCMLタグから構成されます。各タグは、構成ファイル内のテキストの解釈方法をCMLパーサーに伝える命令か、構成ファイル内の値の位置と値セットへのマップ方法を指定するプレースホルダーです。

構成テンプレートには値は含まれないことに注意してください。単に、SAデータベース内の値セットと、管理対象サーバー上の構成ファイルインスタンスとの間の値の移行方法を定義するだけです。詳細については、「値セット」(232ページ)を参照してください。

CMLで作成したテンプレートファイルの拡張子は通常は ".tpl" ですが、これは必須というわけではありません。

CMLタグの構造

CMLタグの基本的な構成要素を次に示します。

@{レベル}{タグタイプ}{ソース};{タイプ};{範囲};{オプション};...;{オプション}@

> **注:**
> CMLタグには、空白と'@'のいずれも指定できません。'@'記号は、その前に'@'をもう1つ付けることでエスケープできます。

次のルールは、すべてのCMLタグに適用されます。

- すべてのCMLタグの先頭と末尾は@記号です。
- 属性を省略する場合は、プレースホルダーとしてセミコロン (;) を入れます。たとえば、次の例では、@name属性とoptional@属性の間に2つの属性が省略されています。

  ```
  @name;;;optional@
  ```

- セミコロンの右側の属性がすべて空の場合、セミコロンは省略可能です。次に例を示します。

  ```
  @name@
  ```

- **{レベル}** - ブロックレベルは、ブロックのネストレベルを指定する整数です。このレベルは、ブロックが複数行にわたるか1行に含まれるかも決定します。レベルが1～99の場合は、複数行のブロックです。101以上の場合は、1行の中のブロックです。ブロック開始タグがあると、それより前にあるそれ以上のレベルのブロックはすべて閉じられます。

  レベル100は予約されており、使用できません。
- {**タグタイプ**} - CMLでは次のタグタイプが定義されています。各タイプはCMLパーサーへの命令を表します。詳細については、「CMLタグのタイプ」(379ページ)を参照してください。
  - コメントタグ: @#および@## ...#@ - テンプレート内のコメントを定義します。
  - 置換タグ: @ - 変数を値セットからの値に置き換える方法を定義します。
  - 命令タグ: @!- CMLパーサーに命令を与えます。
  - ブロックタグ: @[@...@]@- 新しいスコープを作成します。
  - ループタグ: @* - 複数の類似した値を処理するループを作成します。
  - ループターゲットタグ: @.- ループを終了します。
  - 条件タグ: @?
  - DTDタグ: @~ - 定義します
- {**ソース**} - 値セットで値が保存されているキーを定義します。絶対パス名は先頭が"/"文字です。相対パス名は先頭が"/"ではなく、`@!namespace`命令で定義された名前空間キーの値に結合されます。
- {**タイプ**} - 構成ファイルで要求される値と、値セット内の対応する値のデータ型を定義します。たとえば、整数 (int)、文字列 (string)、ブール値 (boolean)、IPアドレス (ip) などです。また、順序あり (ordered) および順序なし (unordered) のリスト (list) およびセット (set) を指定することもできます。
- {**範囲**} - エラーチェックに使用するデータ値の制約を定義します。
- {**オプション**} - CMLタグの動作を変更するために指定できる追加パラメーターを定義します。

必須のCMLタグ

すべてのCMLファイルは、次のCMLタグを使用して、名前空間と、テンプレートでモデル化する構成ファイルのデフォルトのファイル名を定義する必要があります。

- `@!namespace`は、テンプレートの名前空間を定義します。テンプレートで使用される値セットのすべての値は、`@!namespace`タグで定義されるキーでSAデータベースに記録されます。詳細については、「@!namespace CMLタグによる名前空間の定義」(378ページ)を参照してください。

- @!filename-keyは、デフォルトのファイル名がSAデータベースに記録されるキーを定義します。このキーは、独立の名前空間を持つことも、@!namespaceタグで定義された名前空間の後に追加することもできます。詳細については、「@!filename-keyおよび@!filename-default CMLタグによるデフォルト構成ファイル名の定義」(378ページ)を参照してください。
- @!filename-defaultは、テンプレートでモデル化される構成ファイルのディレクトリと名前を定義します。この値は、値セットによって変更することができます。詳細については、「@!filename-keyおよび@!filename-default CMLタグによるデフォルト構成ファイル名の定義」(378ページ)を参照してください。

@!namespace CMLタグによる名前空間の定義

CMLテンプレートファイルの名前空間は、データがデータベースに記録される固有のキー値を定義します。名前空間の値はパス名として表され、ファイルシステムのディレクトリパス名や、Webブラウザーのアドレスバーの URIと共通の形式です。名前空間はnamespaceタグで定義します。

個々の値のパス名は、絶対または相対形式を取ります。絶対パス名は先頭が"/"で、値セット内の値の場所の完全な表現です。先頭が"/"でないパス名は相対パス名で、その値は名前空間の現在の値に結合されます。

namespaceタグは必須です。

CMLテンプレート内のキー名はASCIIである必要があります。他のフィールドやテキストには、ASCIIまたは非ASCIIのテキストが使用できます。

CMLテンプレートのnamespaceタグの例を次に示します。

```
@!namespace=/security/@
```

@!filename-keyおよび@!filename-default CMLタグによるデフォルト構成ファイル名の定義

各テンプレートは、生成した構成ファイルをサーバーにプッシュする際に使用するデフォルトの構成ファイル名を定義する必要があります。このファイル名は、値セットでオーバーライドすることができます。デフォルトのファイル名を使用するには、filename-defaultタグを使用します。

デフォルトのファイル名がSAデータベースに記録される固有のキーも定義する必要があります。このキーは、名前空間と結合されて、デフォルトのファイル名の固有の記録場所を生成することができます。キーはパス名として表される名前空間を定義します。キー値はfilename-keyタグで定義します。

filename-defaultタグとfilename-keyタグは必須です。

CMLテンプレート内のキー名はASCIIである必要があります。他のフィールドやテキストには、ASCIIまたは非ASCIIのテキストが使用できます。

次のCMLの例は、デフォルトのファイル名をSAデータベースに記録するために使用されるキー値が "/files/hosts" であることを指定します。また、生成される構成ファイルのデフォルトのファイル名が "/etc/hosts" であることも指定します。

```
@!filename-key="/files/hosts"@
@!filename-default="/etc/hosts"@
```

CMLタグは次のように1行にまとめることもできます。

```
@!filename-key="/files/hosts";filename-default="/etc/hosts"@
```

/etc/hostsに対するCMLの例

次に示すのは、代表的な/etc/hostsファイルをモデル化するCMLテンプレートの例です。

```
@##############################################
#                                            #
# /etc/hosts (multiplatform)                 #
# Version 2.0                                #
# Joe Author (joe_author@your_company.com)   #
#                                            #
##############################################@
@!namespace=/system/dns/@
@!filename-key="/files/hosts";filename-default="/etc/hosts"@
@!unordered-lines;missing-values-are-error@
@!relaxed-whitespace@
@!sequence-delimiter-is-whitespace@
@!line-comment="#"@
@~host/.ip
type = ip
printable = IP address
description = This is an IP address
@
@~host/.hostnames
type = unordered-hostname-set
printable = Hostnames
description = A set of hostnames
@
@1*host;unordered-namespace-set;;sequence-append@
@.ip@   .hostnames@
@1]@
```

CMLタグのタイプ

主要なCMLタグを次に示します。これらの詳細についてはこの後で説明します。

コメントタグ: @#および@##

このタグは、CMLテンプレートファイルのコメントを定義します。1行のコメントまたは複数行のコメントを定義できます。

構文

```
@#    <1行のコメント>
```

または:

```
@## <複数行にわたるコメント>
     <複数行にわたるコメント>
     <複数行にわたるコメント> #@
```

説明

コメントタグは、CMLファイルの任意の場所にコメントを入れるために使用できます。

ベストプラクティスとしては、CMLテンプレートの先頭にコメントタグを使用してヘッダーを作成し、テンプレート名、元になる構成ファイル、テンプレートの目的、作成者、作成日などの情報を記述することが推奨されます。

属性

なし。

例

次に示すのは1行のコメントです。

```
@# This comment ends at the end of this line.
```

次に示すのは複数行のコメントです。

```
@##
 This comment spans
   multiple lines.
#@
```

次に示すのも複数行のコメントです。

```
@#############################################################
# /etc/hosts (multiplatform) #
# $Id: hosts.tpl 8650 2006-06-05 05:28:03Z joe_author $
#############################################################@
```

置換タグ: @

このタグは、テンプレートファイル内のテキストを、値セットからの値に置き換えます。

構文

@{ソース}[;[{タイプ}][;[{範囲}[;{オプション}[;{オプション}]...]]]]@

説明

置換タグは、CML行内のタグを、名前空間の指定された場所からのデータに置き換えます。これは、この場所にあるテキストがデータであることを示すとともに、そのデータを記録する方法と検証する方法の詳細を指定します。ソース名は、値セット内にデータが存在するインデックスキーを示します。置換タグのその他のフィールドは、データの記録と検証の方法に関する詳細を指定します。

置換タグは、“@”文字の後の特殊文字によって示されない唯一のタグです。置換タグの必須要素はソースだけです。他の要素はすべて省略可能です。

属性

- **ソース**: ソース属性は、値セット内に値を記録してアクセスするために使用されるキーです。ソース属性が相対形式の場合 (先頭が“/”または“.”でない場合)、現在の名前空間の後に追加され、このタグによって読み取られる値を記録するキーの一部となります。名前が絶対形式の場合 (先頭が“/”の場合)、これはキーを表し、値はこのキーの下に記録されます。
- 置換タグの必須要素はソースだけです。他の要素はすべて省略可能です。名前の先頭がドット“.”の場合、これが含まれるループの名前空間の後に追加されます。ループ内部のタグは通常先頭が“.”です。

- **タイプ**: タイプ属性は、置換タグのタイプを指定します。これにより、さまざまな値に対する定義済みのいくつかの制約とエラーチェックが適用されます。置換タグのデフォルトのタイプは “string” です。
- 利用可能なタイプは、「CMLタイプ属性」(392ページ)に記されています。
- **範囲**: 範囲属性を使用すると、値の範囲を設定できます(すべての範囲は、ファイルの読み取り時と、ユーザーによる値の入力時に使用されます)。指定した範囲外の値が構成ファイルにある場合、ファイルの解析中にエラーが発生します。
- 範囲の説明は「CML範囲属性」(398ページ)にあります。
- **オプション**: オプション属性は、タグの動作を変更します。ほとんどのタグは、末尾に複数のオプションを追加できます。オプションの間はセミコロンで区切ります。3つ目のセミコロンより後のすべてのものはオプションと見なされます。オプションは命令タグとしても使用できます。

オプションの説明は、「CMLのグローバルオプション属性」(401ページ)と「CMLの通常オプション属性」(403ページ)にあります。

例1

```
Title=@main_title@
```

この例で、`main_title`は、構成ファイルで“`Title=`“のテキストの後にある文字列を抽出して、値セットのキー/main_titleに記録します。

また、プッシュを実行する際には、`main_title`は`/main_title`キーに記録されている値を値セットから抽出して、構成ファイルの“`Title=`“のテキストの後にプッシュします。

例2

```
Port = @port;port;1024<=&<=2048@
IPAddress = @ipaddress;ip;;optional;delimiter=”/”@
ServerName = @servername;hostname;”localhost”,r”server.*”@
```

命令タグ: @!

このタグは、パーサーのアクションを指定します。利用可能な命令の一覧については、「CMLのグローバルオプション属性」(401ページ)と「CMLの通常オプション属性」(403ページ)を参照してください。

構文

```
@!{オプション}[[;{オプション}]...]@
```

説明

命令タグは、解析時に使用されるオプションを設定します。例としては、名前空間の定義、リストがソート済み、順序あり、順序なしのどれであるか、パーサーが空白をどう解釈するか、使用可能な区切り文字、コメント文字の定義などがあります。

命令タグで使用される属性は、オプションだけです。1つの命令タグに1つ以上のオプションを使用できます。複数のオプションはセミコロンで区切ります。特定の命令タグがパーサーの動作をどのように変えるかについては、埋め込みオプションの説明を参照してください。

属性

命令タグで使用される属性はオプション属性だけです。

オプション: 命令タグのオプション属性は、タグの動作を定義します。ほとんどのタグは、末尾に複数のオプションを追加できます。オプションの間はセミコロンで区切ります。多くのオプションは、他のオプションとの切り替えになっています。このような切り替えグループのオプションの1つがブロック内で使用されている場合、同じグループの他のオプションを同じブロック内で使用することはできません。

オプションの説明は、「CMLのグローバルオプション属性」(401ページ)と「CMLの通常オプション属性」(403ページ)にあります。

例1

次の命令タグは、テンプレート内の空白をタブと空白の任意の組み合わせに一致させることを、CMLパーサーに指示します。

```
@!relaxed-whitespace@
```

例2

次の命令タグの2つのオプションは、構成ファイル内の行の相対的順序が、これらの行から値セットへの値のマッピングにおいて重要でなく、値セット内の値が構成ファイル内のテキストに一致しなくてもエラーではないことを、CMLパーサーに指示します。

```
@!unordered-lines;missing-values-are-null@
```

例3

```
@!namespace=/test/@
@!filename-key="/test";filename-default="/tmp/test.txt"@
@!optional-whitespace@
@!boolean-yes-format="1";boolean-no-format="0"@
@!line-comment-is-semicolon@
@!unordered-lines@
```

ブロック (またはグループ) タグ: @[@...@]@

ブロックタグは、グループタグと呼ばれることもあります。このタグは、関連するタグのブロックまたはグループを作成するもので、関連するタグのグループをネストするために使用できます。

構文

ブロックタグには、1行の構文と複数行の構文があります。

ブロックタグの1行の構文は次のとおりです (リテラル文字列は引用符付き)。

```
"@" [{レベル}] "[" [ ";" {オプション}[ ";" {オプション}]...]"@" {CML statements} "@"
[{level}] "]@"
```

グループタグの複数行の構文は次のとおりです。

```
"@" {レベル} "[" [ ";" {オプション}[ ";" {オプション}]...]"@"
{CML statements}
"@" {レベル} "]@"
```

@[{レベル}][[;{オプション}[;{オプション}]...]@{CMLタグのセット} @[{レベル}]]@

@[{レベル}][[;{オプション}[;{オプション}]...]@

{CMLタグのセット}

@[{レベル}]]@

レベルは、ブロックが複数行にわたるか1行に含まれるかを決定する整数です。レベルが1～99の場合は、複数行のブロックです。101以上の場合は、1行の中のブロックです。レベル100は予約されており、使用できません。

ブロックは明示的に終了されるか、暗黙に終了します。ブロックを明示的に終了するには、終了ブロックタグをレベル番号とともに使用します。たとえば、次のタグはレベル3のブロックを明示的に終了します。`@3]@`.

ブロックを暗黙に終了するには、より小さいレベル番号の終了ブロックタグを使用して外側のブロックを終了するか、より小さいレベルの新しいブロックを定義します。ブロック開始タグがあると、それより前にあるそれ以上のレベルのブロックはすべて閉じられます。

説明

ブロックタグを使用すると、関連するタグをグループ化して、タグのグループをネストできます。ブロックを使用することで、構成ファイルの各部分に対して異なる解析ルールを定義できます。

より大きいレベル番号を使用することで、ブロックの内部に別のブロックをネストできます。その後にレベル値を指定したタグが現れた場合、同じまたはそれより大きいレベル値を持つ閉じていないブロックはすべて閉じられます。ブロック終了タグ@]@は必須ではありません。

開始ブロックタブにはオプション属性を指定できます。属性は、開始タグで宣言されたレベルにあるブロック内部のタグだけに影響します。これに対して、ブロック内部にある命令タグは、現在のレベルとネストしているすべてのブロックの動作に影響します。

ブロックを使用することで、構成ファイルの各セクションにそれぞれ異なる固有のオプションを指定できます。たとえば、構成ファイルのあるセクションでは、真と偽の値がそれぞれ“1”と“0”で表されるとします。また、同じファイルの別のセクションでは、真と偽の値が“T”と“F”で表されるとします。この場合、ブロックタグを使用することで、真と偽の2つの異なる表現方法を分離することができます。

別の例として、構成ファイルの1つのセクションではスペースの数が重要であるのに対して、別のセクションではスペースをいくつ使用してもかまわないという場合が挙げられます。ブロックタグを使用すれば、スペースの数の扱いが異なる部分を指定できます。

属性

ブロックタグでは、名前、タイプ、範囲の各属性は使用されません。

- **レベル**: ブロックレベルは、ブロックのネストレベルを指定する整数です。このレベルは、ブロックが複数行にわたるか1行に含まれるかも決定します。レベルが1～99の場合は、複数行のブロックです。101以上の場合は、1行の中のブロックです。ブロック開始タグがあると、それより前にあるそれ以上のレベルのブロックはすべて閉じられます。

  > **注:**
  > レベル100は予約されており、使用できません。

- **オプション**: オプション属性は、ブロック内のCMLタグの動作を変更します。ブロック内の命令タグは、現在のブロックとネストしたブロック内のCMLタグの動作に影響します。ほとんどのタグは、末尾に複数のオプションを追加できます。オプションの間はセミコロンで区切ります。オプションは命令タグとしても使用できます。

  オプションの説明は、「CMLのグローバルオプション属性」(401ページ)と「CMLの通常オプション属性」(403ページ)にあります。

例1

次の例は2つのブロックを作成します。1つのブロックはもう1つのブロックの内部にネストしています。1行目は1番目のブロックを定義しています。これは外側のブロックです。4行目は2番目のブロックを定義しています。これは1番目のブロックの内部にネストした内側のブロックです。最後から2行目は、内側のブロックを閉じています。この行は省略可能です。最後の行は、外側のブロックを閉じています。最後から2行目を省略した場合、最後の行で両方のブロックが閉じられます。

```
@1[@
@!ordered-lines@
[SectionOne]
@2[@
@!unordered-lines@
optionA = @section_one/option_a@
optionB = @section_one/option_b@
@2]@
@1]@
```

例2

この例は、WindowsのUrlScan.iniファイルの[Options]と[AllowVerbs]の2つのセクションをモデル化します。このファイルの2つのセクションには、キーと値のペアが含まれます。

1番目のセクション(1～3行目)を定義するには、2つのレベルに設定されたブロックタグ([)を使用します。このセクションには、固定の見出しと、キーと値のリストという2種類のデータがあるからです。第1レベルのブロックはテキスト文字列"[Options]"を処理し、第2レベルのブロックはこのセクション内のすべてのキーと値のペアを処理します。

2番目のセクション(4～6行目)は、[AllowVerbs]セクションを定義します。1番目のセクションは、前の例と異なり、@2]@と@1]@のタグで明示的に閉じられていません。これは、次のレベル1セクションの開始(4行目)によって、前のセクションが暗黙に閉じられるからです。

```
@1[;optional;ordered-lines@
[Options]
@2[;unordered-lines@
@1[;optional;ordered-lines@
[AllowVerbs]
@2[;unordered-lines@
```

ループタグ: @*

このタグは、処理ループを定義します。「ループターゲットタグ: @.」(388ページ)も参照してください。

構文

@[{レベル}]*{ソース}[;[{タイプ}][;[{範囲}[;{オプション}[;{オプション}]...]]]]@ {ターゲット}

説明

ループタグは、値のセットが構成ファイルに複数回出現する可能性がある場合に使用されます。ループタグのデフォルトの動作は、その直後の1行のCMLを反復処理することですが、複数の行または1行の中で反復処理するように変更できます。

ループはグループタグの形式の1つです。詳細についてはグループタグを参照してください。

ループタグを使用すると、シーケンス(リストとセット)を列挙することができます。ループ要素に関連するブロックは、入力ファイル内でそのブロックが出現するたびに処理され、値セットでそのデータが出現するたびに出力ファイルに生成されます。

ループ要素に関連するグループに関しては、構成ファイルにそのグループが出現するたびに値セットに新しい要素が記録され、値セットにそのデータが出現するたびに構成ファイルに値がプッシュされます。ソース属性は、値セット内の値をマップするために使用されるインデックスキーです。

属性

- **レベル**: グループレベルは、グループが複数行にわたるか1行に含まれるかを決定する整数です。レベルが1～99の場合は、複数行のグループです。101以上の場合は、1行の中のグループです。レベル100は内部用途に予約されています。グループ開始タグがあると、それより前にあるそれ以上のレベルのグループはすべて閉じられます。
- **ソース**: ソース属性は、値セット内の値にアクセスするために使用されるキーです。ソース属性が相対形式の場合 (先頭が"/"または"."でない場合)、現在の名前空間の後に追加され、このタグによって読み取られる値を記録するキーの一部となります。名前が絶対形式の場合 (先頭が"/"の場合)、これはキーを表し、値はこのキーの下に記録されます。ループタグの必須要素はソースだけです。他の要素はすべて省略可能です。ソース名の先頭がドット "." の場合、これが含まれるループの名前空間の後に追加されます。ループ内部のタグは通常先頭が"."です。
- **タイプ**: タイプ属性は、置換タグのタイプを指定します。これにより、さまざまな値に対する定義済みのいくつかの制約とエラーチェックが適用されます。置換タグのデフォルトのタイプは "string" です。

  タイプの一覧については、「CMLタイプ属性」(392ページ)を参照してください。

  タイプの前に "ordered-" または "unordered-" を付けることができます。また、タイプの後ろに "-set" または "-list" を付けることができます。
  - "ordered-" を前に付けると、値の順序が重要であることを示します。
  - "unordered-" を前に付けると、値が任意の順序になることを示します。
  - "-set" を後ろに付けると、値が一意でなければならないことを指定します。
  - "-list" を後ろに付けると、値が繰り返されてもよいことを指定します。
- **範囲**: 範囲属性を使用すると、値の範囲を設定できます。すべての範囲は、ファイルの読み取り時と、ユーザーによる値の入力時に使用されます。指定した範囲外の値が構成ファイルにある場合、ファイルの解析中にエラーが発生します。

- 範囲の説明は「CML範囲属性」(398ページ)にあります。
- **オプション**: オプション属性は、タグの動作を変更します。ほとんどのタグは、末尾に複数のオプションを追加できます。オプションの間はセミコロンで区切ります。3つ目のセミコロンより後のすべてのものはオプションと見なされます。オプションは命令タグとしても使用できます。

  オプションの説明は、「CMLのグローバルオプション属性」(401ページ)と「CMLの通常オプション属性」(403ページ)にあります。

例1

アスタリスク文字はループタグを示します。次に例を示します。

```
@1*includegroup;ordered-namespace-set;;optional@
#BEGIN_ALTERNATE
@*.include@
#INCLUDE @.@
#END_ALTERNATE
@1]@
```

例2

```
@*users;unordered-user-set;!”root”;field-delimiter-is-semicolon@
@.@;
```

ループターゲットタグ: @.

ループターゲットタグは、ループタグの反復を定義します。「ループタグ: @*」(386ページ)を参照してください。

構文

```
@.[{ソース}[;[{タイプ}][;[{範囲}[;{オプション}[;{オプション}]...]]]]]@
```

説明

ループターゲットタグは、ループ内の値のプレースホルダーを示します。ループタグはループの開始を示すもので、グループタグに似ているのに対して、ループターゲットタグは置換タグに似ています。

グループ内にこのタグがあると、ループの反復のたびに、構成ファイルの現在位置のテキストが、値セット内の現在の値に置き換えられます。オプションのソース属性を使用した場合、ソースはループで作成される名前空間の後に追加されます。

属性

なし。

例

ループターゲットタグは、“@”文字の後のピリオドで示されます。次に例を示します。

```
@*keys;unordered-namespace-set@
@.key@ = @.value@
```

条件タグ: @?

このタグは、条件を定義します。

構文

@[{レベル}]?{ソース}@{テキスト}

説明

条件タグは、構成ファイルにテキストが存在するかどうかを、名前空間内のブール値にマップします。ターゲット構成ファイルを読み取る際には、テキストが一致した場合に名前空間の値は真になり、一致しない場合には偽になります。構成ファイルに書き込む際には、名前空間の値が真の場合、構成ファイルにテキストが書き込まれ、偽の場合は書き込まれません。

これは、タグ外部のものが実際の値になる珍しいタグの1つです。このタグの主な用途は、タグの後のテキストが存在した場合に、名前空間内の場所にブール値の真の値を記録することです。

属性

条件タグには、タイプ、範囲、オプション属性はありません。

- **レベル**: レベルは、グループが複数行にわたるか1行に含まれるかを決定する整数です。レベルが1～99の場合は複数行のグループで、101以上の場合は1行のグループです。レベル100は内部用途に予約されています。グループ開始タグがあると、それより前にあるそれ以上のレベルのグループはすべて閉じられます。
- **ソース**: ソース属性は、ブール値にアクセスするために使用されるキーです。ソース属性が相対形式の場合 (先頭が“/”または“.”でない場合)、現在の名前空間の後に追加され、このタグによって読み取られる値を記録するキーの一部となります。名前が絶対形式の場合 (先頭が“/”の場合)、これはキーを表し、値はこのキーの下に記録されます。

  ソース名の先頭がドット “.” の場合、これが含まれるループの名前空間の後に追加されます。ループ内部のタグは通常先頭が“.”です。

例1

条件タグは疑問符 (?) で表されます。次に例を示します。

```
@?debug@options debug
```

この例では、構成ファイルを構成テンプレートにインポートする際に、テキスト “options debug” が構成ファイル内に存在すれば、キー/debugの値がtrueに設定されます。

アプリケーション構成をプッシュする際には、キー **/debug**に記録された値がtrueの場合、テキスト “options debug” が構成ファイルにプッシュされます。

例2

たとえば、構成ファイル内にキーワード “threaded” がある場合にアプリケーションがスレッド形式になると指定されている場合、CMLは次のようになります。

```
@?is_threaded@threaded
```

これは、名前空間キー/is_threadedの値を、構成ファイルに値 “threaded” が存在する場合にtrueに設定し、構成ファイルに値 “threaded” が存在しない場合にfalseに設定します。

DTDタグ: @~

このタグは、DTDを定義します。

構文

```
@~{ソース}
[type = {タイプ}]
[description = {説明}]
[printable = {ラベル}]
[range = {範囲}]
[{オプション}
...]
@
```

説明

CMLは、ドキュメント型定義 (DTD) タグによる他のCMLタグの属性の事前定義をサポートします。DTDを使用すれば、タグの特性をすべて別の場所に記録しておき、タグ自体の名前を参照するだけで済むため、CMLテンプレートの実際に機能する部分が簡潔になります。

DTD定義を使用すると、ソース属性を持つ任意のタグを定義できます。ループタグ、ループターゲットタグ、置換タグなどがこれにあたります。ただし、命令タグやグループタグは (ソース属性を持たないため) 定義できません。

CMLでDTDタグを使用することのもう1つの利点は、'printable' と'description' の値を定義できることです。'printable' と'description' の値は、フィールドの用途に関する情報をユーザーに与える役割を果たします。'description' 属性の文字列値は、値セットエディター画面でフィールドの上にマウスカーソルを置いたときに表示されます。'printable' 属性の文字列値は、値セットエディターに表示されるパス名を、読みやすいフィールドラベルに置き換える役割を果たします。

CML内のDTDタグは、本質的に複数行のタグです。最初と最後以外の行は任意の順序を取ることができ、printableとdescriptionを除くすべての要素はフィールドに関連付けられます。printableとdescriptionの2つは、DTDで定義されたタグに対してのみ有効です。

構成テンプレートでのDTDタグの使用法の詳細については、「CMLでのDTDタグの使用」(416ページ)を参照してください。XMLテンプレートについては、「XML DTD要素表示のカスタマイズ」(306ページ)を参照してください。

属性

DTDタグではレベル属性は使用されません。DTDタグの必須属性はソースだけです。他の属性はすべて省略可能です。ただし、名前だけが定義されたDTDタグは、何の機能も果たしません。

- **ソース**: ソース属性は、値にアクセスするために使用されるキーです。ソース属性が相対形式の場合 (先頭が"/"または"."でない場合)、現在の名前空間の後に追加され、このタグによって読み取られる値を記録するキーの一部となります。名前が絶対形式の場合 (先頭が"/" の場合)、これはキーを表し、値はこのキーの下に記録されます。ソース名の先頭がドット "." の場合、これが含まれるループの名前空間の後に追加されます。ループ内部のタグは通常先頭が"."です。
- **タイプ**: タイプ属性は、既知のタイプに基づく定義済みの制約とエラーのチェックを値に適用します。置換タグのデフォルトのタイプは "string" で、任意のものに一致します。
- タイプの一覧は、このドキュメントの「CMLタイプ属性」(392ページ)にあります。
- **説明**: 説明属性の値は、タグが表す値に関する簡単な説明の文字列です。この属性は、SAクライアントの値セットエディターでマウスポインターを上に置いたときに表示されるテキストです。
- **ラベル**: ラベル属性の値は、変数のわかりやすい名前の文字列です。これは、SAクライアントの値セットエディターで属性の名前として表示されます。
- **範囲**: 範囲属性を使用すると、値の範囲を設定できます。すべての範囲は、ファイルの読み取り時と、ユーザーによる値の入力時に使用されます。テンプレートで設定した範囲外の値が構成ファイルにある場合、通常はファイルの解析中に例外が発生します。構成ファイルのドキュメントに基づいて、正しい範囲を使用してください。
- 範囲の詳細な説明は「CML範囲属性」(398ページ)にあります。
- **オプション**: オプション属性は、タグの動作を変更または調整する役割を果たします。ほとんどのタグは、末尾に複数のオプションを追加できます。オプションの間はセミコロンで区切ります。タグには任意の数のオプションを追加できます。3つ目のセミコロンより後のすべてのものはオプションと見なされます。オプションの間はセミコロンで区切ります。オプションは命令タグとしても使用できます。

オプションの詳細な説明は、「CMLのグローバルオプション属性」(401ページ)と「CMLの通常オプション属性」(403ページ)にあります。

例

```
@~port
type = port
range = 1024<=&<=2048
printable = Port
description = The port used for this application.It
should be a port number between 1024 and 2048
@
```

CMLタイプ属性

CML属性は、CMLタグの意味を定義し、制御します。この項では、CMLテンプレートで使用できるタイプを定義します。いくつかのタイプは、“-set” または “-list” を後ろに付けることにより、繰り返し値のシーケンスを表すように変更できます。いくつかのタイプは、“ordered-“ または “unordered-“ を前に付けることにより、繰り返し値のシーケンスの順序を無視するように変更できます。

intタイプ

intは数値タイプです。

構文

```
@[{レベル}]{タグタイプ}[[{ソース}][;int][;[{範囲}][;{オプション}[;{オプション}]...]]]@
```

説明

整数値…、-2、-1、0、1、2、… (Z)。

decimalタイプ

decimalは数値タイプです。

構文

```
@[{レベル}]{タグタイプ}[[{ソース}][;decimal][;[{範囲}][;{オプション}[;{オプション}]...]]]@
```

説明

小数です。

guidタイプ

guidは数値タイプです。

構文

```
@[{レベル}]{タグタイプ}[[{ソース}][;guid][;[{範囲}][;{オプション}[;{オプション}]...]]]@
```

説明

グローバルー意識別子 (GUID)、128ビットのID。

stringタイプ

stringは非数値タイプです。

構文

```
@[{レベル}]{タグタイプ}[[{ソース}][;string][;[{範囲}][;{オプション}[;{オプション}]...]]]@
```

説明

stringは、すべての値について、他のタイプを明示的に指定しない場合のデフォルトのタイプです。

quotedstringタイプ

quotedstringは非数値タイプです。

構文

```
@[{レベル}]{タグタイプ}[[{ソース}][;quotedstring][;[{範囲}][;{オプション}[;{オプション}]...]]]@
```

説明

引用符付き文字列です。

booleanタイプ

booleanは非数値タイプです。

構文

@[{レベル}]{タグタイプ}[[{ソース}][;boolean][;[{範囲}][;{オプション}[;{オプション}]...]]]@

説明

ブール値です。

durationタイプ

durationは非数値タイプです。

構文

@[{レベル}]{タグタイプ}[[{ソース}][;duration][;[{範囲}][;{オプション}[;{オプション}]...]]]@

説明

期間です。

ipv6タイプ

IPv6はシステム固有のタイプです。

構文

@[{レベル}]{タグタイプ}[[{ソース}][;ipv6][;[{範囲}][;{オプション}[;{オプション}]...]]]@

説明

CMLは、IPv6アドレスをテキスト文字列として表すための、次の2つの規則形式をサポートしています。

x:x:x:x:x:x:x:x。「x」は、IPv6アドレスの8つの16ビット部分に含まれる1から4桁の16進数を表します。次に例を示します。

ABCD:EF01:2345:6789:ABCD:EF01:2345:6789

IPv6アドレスの「::」は、値がゼロの16ビットの1つまたは複数のグループを示します。「::」がアドレスに現れるのは、1回だけです。「::」は、アドレス先頭または末尾のゼロを圧縮して表す場合にも使用できます。次に例を示します。

2001:DB8:0:0:8:800:200C:417Aは2001:DB8::8:800:200C:417Aになります。

0:0:0:0:0:0:0:1は::1になります。

ipv4タイプ

IPv4はシステム固有のタイプです。

構文

```
@[{レベル}]{タグタイプ}[[{ソース}][;ipv4][;[{範囲}][;{オプション}[;{オプション}]...]]]@
```

説明

IPv4アドレスです。

ipタイプ

ipはシステム固有のタイプです。

構文

```
@[{レベル}]{タグタイプ}[[{ソース}][;ip][;[{範囲}][;{オプション}[;{オプション}]...]]]@
```

説明

IPアドレス(IPv4およびIPv6)です。

hostnameタイプ

hostnameはシステム固有のタイプです。

構文

```
@[{レベル}]{タグタイプ}[[{ソース}][;hostname][;[{範囲}][;{オプション}[;{オプション}]...]]]@
```

説明

ホストサーバーの名前です。

hostタイプ

hostはシステム固有のタイプです。

構文

@[{レベル}]{タグタイプ}[[{ソース}][;host][;[{範囲}][;{オプション}[;{オプション}]...]]]@

説明

ホストのIPアドレスまたはホスト名です。

networkタイプ

networkはシステム固有のタイプです。

構文

@[{レベル}]{タグタイプ}[[{ソース}][;network][;[{範囲}][;{オプション}[;{オプション}]...]]]@

説明

IPv4ネットワークです。

portタイプ

portはシステム固有のタイプです。

構文

@[{レベル}]{タグタイプ}[[{ソース}][;port][;[{範囲}][;{オプション}[;{オプション}]...]]]@

説明

TCPまたはUDPポートです。

userタイプ

userはシステム固有のタイプです。

構文

@[{レベル}]{タグタイプ}[[{ソース}][;user][;[{範囲}][;{オプション}[;{オプション}]...]]]@

説明

ユーザー名です。

groupタイプ

groupはシステム固有のタイプです。

構文

@[{レベル}]{タグタイプ}[[{ソース}][;group][;[{範囲}][;{オプション}[;{オプション}]...]]]@

説明

グループ名です。

file – システム固有のタイプ

構文

@[{レベル}]{タグタイプ}[[{ソース}][;file][;[{範囲}][;{オプション}[;{オプション}]...]]]@

説明

ファイル名です。

dirタイプ

dirはシステム固有のタイプです。

構文

@[{レベル}]{タグタイプ}[[{ソース}][;dir][;[{範囲}][;{オプション}[;{オプション}]...]]]@

説明

ディレクトリパス名です。

emailタイプ

emailはシステム固有のタイプです。

構文

```
@[{レベル}]{タグタイプ}[[{ソース}][;email][;[{範囲}][;{オプション}[;{オプション}]...]]]@
```

説明

電子メールアドレスです。

CML範囲属性

CML属性は、CMLタグの意味を定義し、制御します。この項では、CMLテンプレートで使用できる範囲属性を定義します。CMLタイプの範囲属性は、範囲指定子を使用してタグの有効な値を定義し、制限します。

!& , – 論理演算子

!– NOT指定子

& – AND指定子

, – OR指定子

構文

```
@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;![{範囲}][;{オプション}[;{オプション}]...]]]@
@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;{範囲}&{範囲}][;{オプション}[;{オプション}]...]]]@
@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;{範囲},{範囲}][;{オプション}[;{オプション}]...]]]@
```

説明

範囲指定子を論理演算子で修飾することで、入力の検証方法を制御できます。使用できる演算子は、(優先順位が高い順に) NOT、AND、ORの3つです。

- NOT演算子は感嘆符で表され、前置単項演算子です。これは範囲の意味を反転します。すなわち、範囲を満たす項目は偽を返し、範囲を満たさない項目は真を返します。
- AND演算子はアンパサンドで表され、中置二項演算子です。両方のオペランドが真を返す場合のみ真を返します。
- OR演算子はカンマで表され、中置二項演算子です。どちらかのオペランドが真を返す場合のみ真を返します。

範囲の指定ではスペースは意味を持ちません

> **注:**
> 現在のCMLパーサーでは、タグ内部に空白と'@'のいずれも存在しないことが要求されています。

n< n<= <n <=n =n – 比較指定子

n< –「より大きい」指定子

n<= –「以上」指定子

<n –「未満」指定子

<=n –「以下」指定子

=n –「等しい」指定子

構文

@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;{数値}<][;{オプション}[;{オプション}]...]]]@

@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;{数値}<=][;{オプション}[;{オプション}]...]]]@

@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;<{数値}][;{オプション}[;{オプション}]...]]]@

@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;<=数値}][;{オプション}[;{オプション}]...]]]@

@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;=数値}][;{オプション}[;{オプション}]...]]]@

説明

数値タイプに対して利用可能な指定子は、「より大きい」、「以上」、「未満」、「以下」、「等しい」です。

「より大きい」指定子 (n<) は、数値の後に開き角括弧文字を指定したものです。この範囲が満たされるのは、値が指定した数値よりも大きい場合です。

「以上」指定子 (n<=) は、数値の後に開き角括弧文字と等号文字を指定したものです。この範囲が満たされるのは、値が指定した数値以上の場合です(なお、数値nに対して、n<=は!<nと等価であり、n<,=nとも等価ですが、便宜のために用意されています)。

「未満」指定子 (<n) は、開き角括弧文字の後に数値を指定したものです。この範囲が満たされるのは、値が指定した数値よりも大きい場合です。

「以下」指定子 (<=n) は、開き角括弧文字と等号文字の後に数値を指定したものです。この範囲が満たされるのは、値が指定した数値以上の場合です(なお、数値nに対して、<=nは!n<と等価であり、<n,=nとも等価ですが、便宜のために用意されています)。

「等しい」指定子 (=n) は、等号文字の後に数値を指定したものです。この範囲が満たされるのは、値が指定した数値に等しい場合です。

2つの範囲指定子をAND演算子で結合する場合は、0<=&<256のように、「より大きい」(または「以上」)指定子の後に「未満」(または「以下」)指定子を置くことを推奨します。

範囲の指定ではスペースは意味を持ちません

> **注:**
> 現在のCMLパーサーでは、タグ内部に空白と'@'のいずれも存在しないことが要求されています。

" – 文字列リテラル指定子

構文

```
@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;"{文字列}"][;{オプション}[;{オプション}]...]]]@
```

説明

文字列リテラル指定子は、二重引用符文字、テキスト文字列、二重引用符文字の順に並んだものです。引用符とエスケープのルールは、C言語と同じです。すなわち、文字列内の引用符はバックスラッシュでエスケープします。改行は\n、タブ文字は\t、リテラルのバックスラッシュは\\で表されます。この範囲が満たされるのは、文字列値がテキストに正確に一致する場合です。

範囲の指定ではスペースは意味を持ちません

> **注:**
> 現在のCMLパーサーでは、タグ内部に空白と'@'のいずれも存在しないことが要求されています。

r" – 正規表現指定子

構文

```
@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;r"{正規表現}"][;{オプション}[;{オプション}]...]]]@
```

説明

正規表現指定子は、"r" 文字、二重引用符文字、正規表現、二重引用符文字 (") に並んだものです。引用符とエスケープのルールは、Pythonの正規表現とほぼ同じですが、引用符文字はバックスラッシュ文字でエスケープする必要があります。この範囲が満たされるのは、文字列値が正規表現に一致する場合です。

範囲の指定ではスペースは意味を持ちません

> **注:**
> 現在のCMLパーサーでは、タグ内部に空白と'@'のいずれも存在しないことが要求されています。

CMLのグローバルオプション属性

CML属性は、CMLタグの意味を定義し、制御します。この項では、CMLテンプレートで使用できるグローバル属性を定義します。グローバルオプションは命令タグのみで使用でき、他のタグタイプのオプション属性としては使用できません。

@!filename-key属性

構文

`@!filename-key={キー}@`

`{キー}`にはデフォルト値はありません。

説明

`filename-key`は、プッシュの際に生成するファイルの名前を含む値セットのキーのパスを指定します。

`filename-key`の値はパス名です。これは相対パスで指定でき、先頭がスラッシュ (/) でなくてもかまいません。

filename-keyの値の末尾に/を置くことはできません。この要件は、今後のバージョンでは緩和される可能性があります。

@!filename-default属性

構文

`@!filename-default={ファイル名 }@`

`{filename}`にはデフォルト値はありません。

説明

`filename-default`は、値セットにファイル名がない場合に返されるデフォルトのファイル名を指定します。たとえば、ユーザーが値セットエディターにファイル名を入力して、filename-defaultの値をオーバーライドする可能性があります。

@!full-templateおよび@!partial-template属性

構文

```
@!full-template@
```

```
@!partial-template@
```

full-templateがデフォルトの動作です。

説明

full-templateはデフォルトの動作であり、ファイルで予期されるすべてのデータがテンプレートでモデル化される必要があることを示します。

partial-templateは、ファイル内で一致しないデータは無視され、出力に直接渡されることを示します。このオプションは、preserve-formatと組み合わせる必要があります。

@!timeout属性

構文

@!timeout={分}@

{分}のデフォルト値は1です。

説明

timeoutは、構成の合計タイムアウトに加算する時間を分単位で指定します。有効なタイムアウトは、0～999 (両端含む) の整数です。構成内のすべてのテンプレートのタイムアウトの合計に、構成のデフォルトのタイムアウト (10分) を加算したものが、構成全体の最終的なタイムアウト値になります。

@!unix-newlinesおよび@!windows-newlines属性

構文

```
@!unix-newlines@
@!windows-newlines@
```

unix-newlinesがデフォルトの動作です。

説明

unix-newlinesはデフォルトの動作であり、このテンプレートから生成される構成ファイルの改行が、Unixスタイル(ASCIIラインフィード文字)であることを示します。

windows-newlinesは、このテンプレートから生成される構成ファイルの改行が、Windowsスタイル(ASCIIキャリッジリターンとラインフィードの組み合わせ)であることを示します。

CMLの通常オプション属性

CML属性は、CMLタグの意味を定義し、制御します。この項では、CMLテンプレートで使用できるオプション属性を定義します。通常オプションは、命令タグまたは、他のタグタイプのオプション属性として使用できます。

@! unordered-linesおよび@!ordered-lines属性

命令タグの構文

@!unordered-lines@
@!ordered-lines@

unordered-linesがデフォルトの動作です。

オプション属性の構文

@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;unordered-lines[;{オプション}]...]]]]@
@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;ordered-lines[;{オプション}]...]]]]@

グループに対して有効です。

説明

unordered-linesを指定すると、テンプレートの子タグは任意の順序で現れることができます。ただし、順序ありシーケンス要素の中の項目の位置は保持されます。unordered-linesがデフォルトの動作です。

ordered-linesは、テンプレートオブジェクトの子タグ(行、ループ、条件など)が、テンプレートに指定された順序でファイルに現れる必要があることをパーサーに指示します。

unordered-elementsおよびordered-elements属性

命令タグの構文

@!unordered-elements@
@!ordered-elements@

unordered-elementsがデフォルトの動作です。

オプション属性の構文

@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;unordered-elements[;{オプション}]...]]]]@
@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;ordered-elements[;{オプション}]...]]]]@

グループに対して有効です。

説明

unordered-elementsを指定すると、現在のグループの子タグは任意の順序で現れることができます。ただし、順序ありシーケンス要素の中の項目の位置は保持されます。unordered-elementsがデフォルトの動作です。

ordered-elementsは、グループオブジェクトの子タグ(ループ、条件、要素など)が、テンプレートに指定された順序でファイルに現れる必要があることをパーサーに指示します。

relaxed-whitespaceおよびstrict-whitespace属性

命令タグの構文

@!relaxed-whitespace@
@!strict-whitespace@

relaxed-whitespaceがデフォルトの動作です。

オプション属性の構文

@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;relaxed-whitespace[;{オプション}]...]]]]@
@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;strict-whitespace[;{オプション}]...]]]]@

グループに対して有効です。

説明

relaxed-whitespaceを指定すると、テンプレート内のスペースはタブとスペースの任意の組み合わせに一致します。relaxed-whitespaceがデフォルトの動作です。

strict-whitespaceを指定すると、テンプレート内の空白はファイルと正確に一致する必要があります。

required-whitespaceおよびoptional-whitespace属性

命令タグの構文

@!required-whitespace@
@!optional-whitespace@

required-whitespaceがデフォルトの動作です。

オプション属性の構文

@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;required-whitespace[;{オプション}]...]]]]@
@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;optional-whitespace[;{オプション}]...]]]]@

グループに対して有効です。

説明

required-whitespaceを指定すると、テンプレート内のスペースはファイルに存在する必要があります。optional-whitespaceを指定すると、意味のないスペースはファイルに存在しなくてもよくなります。

missing-values-are-nullおよびmissing-values-are-error属性

命令タグの構文

@!missing-values-are-null@

@!missing-values-are-error@

missing-values-are-nullがデフォルトの動作です。

オプション属性の構文

@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;missing-values-are-null[;{オプション}]...]]]]@
@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;missing-values-are-error[;{オプション}]...]]]]@

説明

missing-values-are-nullは、ファイルに見つからない値がNullであり、値セットに格納されないことを指定します。

missing-values-are-errorを指定すると、テンプレートに指定されたすべての値がファイルまたは値セットに存在しない場合、エラーが発生します。

case-insensitive-keywordsおよびcase-sensitive-keywords属性

命令タグの構文

```
@!case-insensitive-keywords@
@!case-sensitive-keywords@
```

case-insensitive-keywordsがデフォルトの動作です。

オプション属性の構文

```
@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;case-insensitive-keywords[;{オプション}]...]]]]@
@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;case-sensitive-keywords[;{オプション}]...]]]]@
```

説明

case-insensitive-keywordsは、ファイル内のリテラルテキストに大文字と小文字を区別しないで一致します。case-insensitive-keywordsがデフォルトの動作です。

case-sensitive-keywordsは、テンプレート内のリテラルテキストが大文字と小文字を区別してファイルと一致する必要があることを指定します。

reluctant属性

命令タグの構文

```
@!reluctant@
```

オプション属性の構文

```
@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;reluctant[;{オプション}]...]]]]@
```

説明

reluctantは、特定のループまたはシーケンスで構成ファイルからの可能な限り少ない要素との照合を試みるように指定します。これは、ループおよびシーケンスのデフォルトの動作ではありません。

requiredおよびoptional属性

命令タグの構文

```
@!required@
@!optional@
```

requiredがデフォルトの動作です。

命令タグでoptionalを使用すると、意図しない結果が生じる可能性があります。

オプション属性の構文

```
@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;required[;{オプション}]...]]]]@
@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;optional[;{オプション}]...]]]]@
```

説明

required要素は一致する必要があります(オプションのグループ内にネストされている場合を除く)。
requiredがデフォルトの動作です。

optional要素はオプションです。

optionalをオプション属性として使用するのは、命令タグ以外のすべてのタグに対して有効です。命令タグでoptionalを使用すると、意図しない結果が生じる可能性があります。

skip-lines-without-valuesおよびshow-lines-without-values属性

命令タグの構文

```
@!skip-lines-without-values@
@!show-lines-without-values@
```

skip-lines-without-valuesがデフォルトの動作です。

オプション属性の構文

```
@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;skip-lines-without-values[;{オプション}]...]]]]@
```

@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;show-lines-without-values[;{オプション}]...]]]]@

説明

skip-lines-without-valuesは、行に置換要素があり、それらの要素のすべての値がNullの場合、その行を出力しないことを指定します。skip-lines-without-valuesがデフォルトの動作です。

show-lines-without-valuesは、Null値の有無に関わらず、すべての行を出力することを指定します。

skip-groups-without-valuesおよびshow-groups-without-values属性

命令タグの構文

@!skip-groups-without-values@
@!show-groups-without-values@

skip-groups-without-valuesがデフォルトの動作です。

オプション属性の構文

@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;skip-groups-without-values[;{オプション}]...]]]]@
@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;show-groups-without-values[;{オプション}]...]]]]@

説明

skip-groups-without-valuesは、グループに置換要素があり、それらの要素のすべての値がNullの場合、そのグループを出力しないことを指定します。skip-groups-without-valuesがデフォルトの動作です。

show-groups-without-valuesは、Null値の有無に関わらず、すべてのグループを出力することを指定します。

sequence-append、sequence-replace、sequence-prepend属性

命令タグの構文

@!sequence-append@
@!sequence-replace@
@!sequence-prepend@

sequence-appendがデフォルトの動作です。

ループとシーケンスに対して有効です。

オプション属性の構文

@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;sequence-append[;{オプション}]...]]]]@
@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;sequence-replace[;{オプション}]...]]]]@
@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;sequence-prepend[;{オプション}]...]]]]@

説明

sequence-appendを指定すると、シーケンス要素の子範囲は親範囲内のシーケンス要素の後に追加されます。sequence-appendがデフォルトの動作です。

sequence-replaceを指定すると、シーケンス要素の子範囲は親範囲内のシーケンス要素を置き換えます。

sequence-prependを指定すると、シーケンス要素の子範囲は親範囲内のシーケンス要素の前に追加されます。

not-primary-fieldおよびprimary-field属性

命令タグの構文

@!not-primary-field@
@!primary-field@

not-primary-fieldがデフォルトの動作です。

オプション属性の構文

@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;not-primary-field[;{オプション}]...]]]]@
@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;primary-field[;{オプション}]...]]]]@

説明

not-primary-fieldは、リストの集約の際に、重複するアイテムを識別する目的でこのフィールドを使用できないことを示します。

not-primary-fieldがデフォルトの動作です。

primary-fieldは、リストの集約の際に、重複するアイテムを識別する目的でこのフィールドを使用することを示します。

シーケンス内部のシーケンスおよび置換タグに対して有効です。

namespace属性

命令タグの構文

@!namespace={名前空間}@

{名前空間}のデフォルト値は"/" (ルート名前空間)です。

オプション属性の構文

@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;namespace={名前空間}[;{オプション}]...]]]]@

{名前空間}のデフォルト値は"/" (ルート名前空間)です。

説明

namespaceは、非修飾名 (前にスラッシュまたはピリオドが付かない名前)を持つ要素が記録される名前空間を識別する文字列です。

{名前空間}のデフォルト値は、文字列"/" (スラッシュ)で表されるルート名前空間です。

名前空間の値はパス名です。先頭はスラッシュ(/)である必要があります。

boolean-no-format属性

命令タグの構文

@!boolean-no-format={文字列}@

{文字列}のデフォルト値は "no" です。

オプション属性の構文

@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;boolean-no-format={文字列}[;{オプション}]...]]]]@

{文字列}のデフォルト値は“no”です。

説明

boolean-no-formatは、偽のブール値要素に一致する文字列を指定します。ブール値置換タグに対して有効です。

boolean-yes-format属性

命令タグの構文

@!boolean-yes-format={文字列}@

{文字列}のデフォルト値は“yes”です。

オプション属性の構文

@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;boolean-yes-format={文字列}[;{オプション}]...]]]]@

{文字列}のデフォルト値は“yes”です。

説明

boolean-yes-formatとboolean-no-formatは、ブール値要素に一致する文字列を指定します。{文字列}のデフォルト値は“yes”です。

ブール値置換タグに対して有効です。

delimiter属性

whitespace-delimited
comma-delimited
semicolon-delimited
tab-delimited
quote-delimited
delimiter

命令タグの構文

@!whitespace-delimited@
@!comma-delimited@
@!semicolon-delimited@
@!tab-delimited@

@!quote-delimited@
@!delimiter={文字列}@

whitespace-delimitedがデフォルトの動作です。

オプション属性の構文

@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;whitespace-delimited[;{オプション}]...]]]]@
@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;comma-delimited[;{オプション}]...]]]]@
@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;semicolon-delimited[;{オプション}]...]]]]@
@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;tab-delimited[;{オプション}]...]]]]@
@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;quote-delimited[;{オプション}]...]]]]@
@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;delimiter={文字列}[;{オプション}]...]]]]@

whitespace-delimitedがデフォルトの動作です。

説明

delimiterはデフォルトの区切り文字を設定します。sequence-delimiterとfield-delimiterを明示的に指定しない場合は、この値が継承されます。

置換およびシーケンスタグに対して有効です。

line-comment属性

line-comment-is-comma
line-comment-is-semicolon
line-comment-is-tab
line-comment-is-whitespace
line-comment

命令タグの構文

@!line-comment-is-comma@
@!line-comment-is-semicolon@
@!line-comment-is-tab@
@!line-comment-is-whitespace@
@!line-comment={文字列}@

{文字列}のデフォルト値はありません。

オプション属性の構文

```
@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;line-comment-is-comma[;{オプション}]...]]]]@
@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;line-comment-is-semicolon[;{オプション}]...]]]]@
@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;line-comment-is-tab[;{オプション}]...]]]]@
@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;line-comment-is-whitespace[;{オプション}]...]]]]@
@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;line-comment={文字列}[;{オプション}]...]]]]@
```

{文字列}のデフォルト値はありません。

説明

`line-comment`は、行の残りの部分をコメントと解釈することを示す文字を設定します。

sequence-delimiter属性

```
sequence-delimiter-is-comma
sequence-delimiter-is-semicolon
sequence-delimiter-is-tab
sequence-delimiter-is-whitespace
sequence-delimiter-is-quote
sequence-delimiter
```

命令タグの構文

```
@!sequence-delimiter-is-comma@
@!sequence-delimiter-is-semicolon@
@!sequence-delimiter-is-tab@
@!sequence-delimiter-is-whitespace@
@!sequence-delimiter-is-quote@
@!sequence-delimiter={文字列}@
```

デフォルトでは、`sequence-delimiter`はdelimiterの値を使用します。delimiterのデフォルト値は`whitespace-delimited`です。

オプション属性の構文

@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;sequence-delimiter-is-comma[;{オプション}]...]]]]@
@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;sequence-delimiter-issemicolon[;{オプション}]...]]]]@
@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;sequence-delimiter-is-tab[;{オプション}]...]]]]@
@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;sequence-delimiter-is-whitespace[;{オプション}]...]]]]@
@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;sequence-delimiter-is-quote[;{オプション}]...]]]]@
@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;sequence-delimiter={文字列}[;{オプション}]...]]]]@

デフォルトでは、sequence-delimiterはdelimiterの値を使用します。delimiterのデフォルト値はwhitespace-delimitedです。

説明

sequence-delimiterは、シーケンス内のアイテムを区切る文字を設定します。デフォルトでは、sequence-delimiterはdelimiterの値を使用します。delimiterのデフォルト値はwhitespace-delimitedです。

シーケンスに対して有効です。

field-delimiter属性

field-delimiter-is-comma
field-delimiter-is-semicolon
field-delimiter-is-tab
field-delimiter-is-eol
field-delimiter-is-whitespace
field-delimiter-is-quote
field-delimiter

命令タグの構文

@!field-delimiter-is-comma@
@!field-delimiter-is-semicolon@
@!field-delimiter-is-tab@
@!field-delimiter-is-whitespace@
@!field-delimiter-is-quote@
@!field-delimiter={文字列}@

デフォルトでは、field-delimiterはdelimiterの値を使用します。delimiterのデフォルト値はwhitespace-delimitedです。

オプション属性の構文

```
@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;field-delimiter-is-comma[;{オプション}]...]]]]@
@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;field-delimiter-is-semicolon[;{オプション}]...]]]]@
@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;field-delimiter-is-tab[;{オプション}]...]]]]@
@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;field-delimiter-is-whitespace[;{オプション}]...]]]]@
@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;field-delimiter-is-quote[;{オプション}]...]]]]@
@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;field-delimiter={文字列}[;{オプション}]...]]]]@
```

デフォルトでは、field-delimiterはdelimiterの値を使用します。delimiterのデフォルト値はwhitespace-delimitedです。

説明

field-delimiterは、置換要素値の解析を終了させる文字を設定します。デフォルトでは、field-delimiterはdelimiterの値を使用します。delimiterのデフォルト値はwhitespace-delimitedです。

置換およびシーケンスタグに対して有効です。

line-continuation属性

命令タグの構文

```
@!line-continuation={文字列}@
```

オプション属性の構文

```
@[{レベル}]{タグタイプ}[[{ソース}][;[{タイプ}][;[{範囲}][;line-continuation={文字列}[;{オプション}]...]]]]@
```

説明

line-continuationは、構成ファイルの現在の行を次の行に継続させることを指定する文字を設定します。

CMLでのDTDタグの使用

CMLは、ドキュメント型定義 (DTD) タグによるCMLタグの属性の事前定義をサポートします。DTDタグをCMLで使用することにより、SAクライアントでテンプレートが表示される方法の一部を変更できます。DTD定義は通常はファイルの先頭にあり、タグは名前とタグタイプだけに短縮されます。

CMLでDTDタグを使用することの最大の利点は、'printable' と'description' の値を定義できることです。これらはSAクライアントに反映され、使いやすさの向上につながります。DTD定義を使用すると、名前を持つ任意のタグを定義できます。ループタグ、ループターゲットタグ、置換タグなどがこれにあたります。ただし、命令タグやブロックタグは定義できません。CML内のDTDタグは、本質的に複数行のタグです。

DTDタグの例

ここでは、1つのタグを例に取り、そのタグのDTDバージョンを作成します。CML内のDTDタグは、通常のCMLタグとそれほど異なるわけではありません。「タグタイプ」以外のタグのすべての要素が含まれています。

次のCMLタグを例に取ります。

```
@*deny_header;unordered-string-set;;sequence-delimiter=":";optional@
```

これは、CMLの次の形式を表すインスタンスです。

```
@<タグタイプ><名前>;<データ型>;;<オプション1>;<オプション2>@
```

これのDTDバージョンは、既存の要素を対象として、次のように並べ替えます。

```
<コードブロック開始>
@~<名前>
type = <データ型>
description = <説明>
printable = <ラベル>
<オプション1>
<オプション2>
...
@
@<タグタイプ><名前>@
<コードブロック終了>
```

これからわかるように、この使用法では2つの新しい要素が使用可能です。“description” と“printable” です。“printable” を定義すると、SAクライアントでのこのタグのメインテキストが定義されます。“description” を定義すると、SAクライアントでのこの値に関する説明が作成されます。この説明は、SAクライアントの値セットエディターでフィールドの上にマウスポインターを移動したときに表示されます。

このタグの完全なDTD形式を次に示します。

```
<コードブロック開始>
@~deny_header
type = unordered-string-set
printable = Headers to Deny
description = This is a list of headers that IIS should deny
sequence-delimiter = ":"
optional
@
@*deny_header@
<コードブロック終了>
```

上の例には、いくつか注意すべき点があります。“description” の値を定義する際に、値は複数行にわたることができますが、2行目以降の1文字目はスペースである必要があります。

オプションは`<option>=<value>`という形式の単独の行で定義されます。`"="`記号の前後にはスペースが必要です。

これ以降、`@*deny_header@`というタグを使用すると、パーサーはこのタグの情報すべてに対して、先に定義されたDTDを使用します。

DTD定義されたタグ`@*deny_header@`を`@*deny_header;unordered-string-set@`のような行によって再定義すると、CMLテンプレートが無効になります。

> **注:**
> また、DTDスタイルのCMLは現時点では必須ではありませんが、SAクライアントでアプリケーション構成を表示する場合にはわかりやすいという利点があります。DTDタグを使用しない場合、'printable' および 'description' フィールドは表示されず、元の変数名だけが表示されます。

シーケンスの集約

アプリケーション構成の値は、アプリケーション構成継承階層 (継承スコープとも呼びます) 内のさまざまなレベルにわたって設定できるので、アプリケーション構成をサーバーにプッシュする際に、複数のシーケンス値をマージする方法を管理できることが重要です。

ACMには、継承スコープ全体でシーケンス値をマージする方法を管理する機能があります。これによってたとえば、カスタマースコープ、グループスコープ、サーバースコープ内のシーケンスに値をいくつか追加し、この値をすべてマージして最終的なシーケンスを作成することができます。

シーケンス値をマージする方法はCMLテンプレートの特殊タグを使って管理し、シーケンスのマージは次の3つのモードで行います。

- 「シーケンスの置換」(418ページ): 具体的なスコープのシーケンス値で、一般的なスコープの値を置換します。この処理は、セットとリストの両方のシーケンスに適用されます。
- 「シーケンスのアペンド」(419ページ): リストの場合、一般的なスコープの値は、具体的なスコープの値にアペンド (末尾に追加) されます。重複する値があっても、削除されません。セットも同様ですが、重複した値をマージする点が異なります。リストの場合、重複する値には`primary-key`タグを付けて子要素とし、マージします。スカラーの場合、重複した値は単に削除されるので、最も具体的なスコープの値のみが残ることになります (最後のオカレンスはマージされたシーケンス)。これがデフォルトのモードであり、他に何も指定しない場合に適用されます。
- **シーケンスのプリペンド**: 動作はアペンドと同じですが、一般的なスコープの値が、具体的なスコープの値にプリペンド (先頭に追加) されます。

以上の処理のしくみを、次の2つのセットを例に説明します。

- “a, b” – 継承スコープの具体的な (内部) レベルにあります。たとえば、サーバーインスタンスのレベルなどです。
- “a, b” – 継承スコープの一般的な (外部) レベルにあります。たとえば、サーバーグループのレベルなどです。

アプリケーション構成テンプレートをサーバーにプッシュすると、次のようにマージされます。

- シーケンスの置換: “a, b”
- シーケンスのアペンド: “a, b, c, d”
- シーケンスのプリペンド: “c, d, a, b”

シーケンスの集約はスコープ間だけでなく、1つのスコープ内でも発生します。このような集約は、名前空間のシーケンスに重複した値がある場合に行われます。

シーケンスの置換

置換のマージモード (CMLタグは“`sequence-replace`”) では、スコープで定義されたシーケンスの内容は、それよりも一般的なスコープの内容を置換し、シーケンスの個々の要素がマージされることはありません。

たとえば、構成テンプレートのCMLソース内のリストにsequence-replaceタグを設定すると、サーバーインスタンスのレベルで設定した値は、グループレベルとアプリケーション構成のデフォルト値のレベルで設定した値を上書き(置換)します。

たとえばetc/hostsファイル内で、次のリストをグループレベル(外部)で定義したとします。

```
/system/dns/host/1/ip                      127.0.0.1
/system/dns/host/1/hostnames/1  localhost
/system/dns/host/1/hostnames/2  mymachine
/system/dns/host/2/ip           10.10.10.10
/system/dns/host/2/hostnames/1  loghost
```

同じリストを、デバイススコープ(内部)で次のように定義します。

```
/system/dns/host/1/ip           127.0.0.1
/system/dns/host/1/hostnames/1  localhost
/system/dns/host/1/hostnames/2  mymachine.mydomain.net
/system/dns/host/2/ip           10.10.10.100
/system/dns/host/2/hostnames/1  mailserver
```

テンプレートで、/system/dns/host要素にsequence-replaceタグを指定している場合、構成ファイルをサーバーにプッシュすると、次のようになります。

```
127.0.0.1 localhost mymachine.mydomain.net
10.10.10.100 mailserver
```

シーケンスのアペンド

アペンドのマージモード (CMLタグは“sequence-append”) をシーケンスで使用すると、一般的なスコープの値が具体的なスコープの値の末尾に追加されます。シーケンスのアペンドは、リスト値のマージに適用されるデフォルトモードです。テンプレートのCMLで何も設定しない場合、シーケンスはアペンドされます。

たとえばetc/hostsファイル内で、次のようにリストをグループレベル(外部)で定義したとします。

```
/system/dns/host/1/ip                      127.0.0.1
/system/dns/host/1/hostnames/1  localhost
/system/dns/host/1/hostnames/2  mymachine
/system/dns/host/2/ip           10.10.10.10
/system/dns/host/2/hostnames/1  loghost
```

同じリストを、デバイススコープ(内部)で次のように定義します。

```
/system/dns/host/1/ip           127.0.0.1
/system/dns/host/1/hostnames/1  localhost
/system/dns/host/1/hostnames/2  mymachine.mydomain.net
/system/dns/host/2/ip           10.10.10.100
/system/dns/host/2/hostnames/1  mailserver
```

上記の値セットの場合、/system/dns/host要素がリストであり、構成テンプレートでsequence-appendタグが指定されている場合、構成ファイルをサーバーにプッシュすると次のようになります。

```
127.0.0.1 localhost mymachine.mydomain.net
10.10.10.100 mailserver
127.0.0.1 localhost mymachine
10.10.10.10 loghost
```

ホストファイルには重複エントリを含めることができないので、/system/dns/host要素は、リストではなく、重複エントリが許可されるセットとして構成テンプレートで指定されます。上記の例でリスト値が重複しないようにするには、構成テンプレートの作成時にプライマリキーオプションを使用します。

プライマリキーオプションを使ったシーケンスマージ

セットをアペンドモードで操作する場合、具体的なスコープに新しい値が追加されると、これは一般的なスコープの値の末尾に追加されます。重複値は結果の値とマージされて、具体的なスコープの位置に従って結果シーケンス内に配置されます。

この処理が、マージ後のシーケンスの値にどのような影響を与えるかは、シーケンス内に含まれるデータのタイプによって異なります。

- シーケンス内の要素がスカラーである場合、最も具体的なスコープの値が使用されます。つまり、サーバーインスタンスレベルの値が、グループレベルの値を置換します。
- シーケンス内の要素が名前空間である場合、要素で指定されているマージモード (この例ではアペンド) に基づいて、プライマリフィールドのマッチングによって値が取得されます。

/system/dns/host/.ip値が重複しないようにするには、構成テンプレートの作成時にCML primary-keyオプションを使用します。このオプションを設定すると、/system/dns/host/.ipの値が同じエントリは同一であるとみなされ、内容をマージします。

上記の例でこのオプションを使用し、構成ファイルをサーバーにプッシュすると次のようになります。

```
127.0.0.1 localhost mymachine.mydomain.net mymachine
10.10.10.100 mailserver
10.10.10.10 loghost
```

> **注:**
> セットにはプライマリキーが設定されていないこともありますが、シーケンス内にスカラーがある場合に

> は、すべてのスカラー値を集約してプライマリキーとして使用します。スカラーがない場合、最初のシーケンスの値をすべて集約し、これをプライマリキーとして使用します。この方法では精度は低くなりますが、ほとんどのケースにおいてマージを効率化できます。正しい値をプライマリキーとして使用するには、シーケンスでプライマリキーを明示的に設定することをお勧めします。

シーケンスのプリペンド

プリペンドのマージモード (CMLタグは“sequence-prepend”) をシーケンスで使用すると、一般的なスコープの値が具体的なスコープの値の先頭に追加されます。

たとえばetc/hostsファイル内で、次のシーケンスをグループレベル (外部) で定義したとします。

```
/system/dns/host/1/ip 127.0.0.1
/system/dns/host/1/hostnames/1 localhost
/system/dns/host/1/hostnames/2 mymachine
/system/dns/host/2/ip 10.10.10.10
/system/dns/host/2/hostnames/1 loghost
```

同じシーケンスを、デバイススコープ (内部) で次のように定義します。

```
/system/dns/host/1/ip 127.0.0.1
/system/dns/host/1/hostnames/1 localhost
/system/dns/host/1/hostnames/2 mymachine.mydomain.net
/system/dns/host/2/ip 10.10.10.100
/system/dns/host/2/hostnames/1 mailserver
```

/system/dns/host要素がセットであり、構成テンプレートで sequence-prependタグが指定されている場合、構成ファイルをサーバーにプッシュすると次のようになります。

```
10.10.10.10 loghost
127.0.0.1 mymachine localhost mymachine.mydomain.net
10.10.10.100 mailserver
```

CMLの文法

次の表に、何種類かのCMLタグを対象としてCMLの文法を示します。

CMLの文法

CMLタグ/要素	説明
置換タグ	"@" ソース [";" [タイプ] [";" [範囲] *オプション]] "@"
データ定義タグ	"@~" ソース CRLF *定義行 "@"

CMLの文法 (続き)

CMLタグ/要素	説明
条件タグ	"@" [グループレベル] "?" ソース [";" [タイプ] [";" [範囲] *オプション]] "@"
ループタグ	“@" [グループレベル] "*" ソース [";" [タイプ] [";" [範囲] *オプション]] "@"
ループターゲットタグ	"@.@"
ブロックタグ	"@" [グループレベル] "[" *オプション "@"
ブロック終了タグ	"@" [グループレベル] "]@"
行継続タグ	"@\"
命令タグ	"@!"*オプション "@"
1行コメント	"@#" [文字列 CRLF]
複数行コメント	"@##" *[文字列 / CRLF] "#@"
定義行	タイプ行 / 範囲行 / オプション行 / ラベル行 / 説明行
タイプ行	"type" WSP "=" WSP タイプ要素 CRLF
範囲行	"range" WSP "=" WSP 範囲 CRLF
オプション行	オプション要素 CRLF
ラベル行	"printable" WSP "=" WSP 文字列 CRLF
説明行	"description" WSP "=" *[WSP 文字列 CRLF]
グループレベル	整数
ソース	絶対パス / 相対パス / ローカルパス
絶対パス	"/" パスコンポーネント* 名前
相対パス	[パスコンポーネント*] 名前
パスコンポーネント	(名前 / シーケンスID) "/"
シーケンスID	整数
ローカルパス	"." 名前
名前	文字列
タイプ	シーケンス / タイプ要素
シーケンス	[順序 "-"] タイプ要素 "-" シーケンス要素
シーケンス要素	"set" / "list"

CMLの文法 (続き)

CMLタグ/要素	説明
タイプ要素	"int" / "string" / "ip" / "port" / "file" / ...
順序	"ordered" / "unordered"
範囲	AND範囲 *["," AND範囲]
AND範囲	範囲要素 *["&" 範囲要素]
範囲要素	数値範囲 / 文字列範囲
数値範囲	より大きい範囲 / 以上範囲 / 未満範囲 / 以下範囲 / 等しい範囲
文字列範囲	文字列リテラル / 正規表現
大きい範囲	整数 ">"
以上範囲	整数 ">="
未満範囲	">" 整数
以下範囲	">=" 整数
等しい範囲	"=" 整数
文字列リテラル	<"> 文字列 <">
正規表現	"r" <"> 文字列 <">
オプション	";" オプション要素
オプション要素	オプション名 / オプション名 値
オプション名 値	オプション名 値
オプション名	文字列
オプション値	文字列

アプリケーションデプロイメント

- **アプリケーションエキスパート**は、複数階層のアプリケーションがどのように動作し、アプリケーションの各部分 (コードやスクリプトなど) をどのようにデプロイすれば最適な結果が得られるかを正確に理解しています。
- **オペレーションエキスパート**は、データセンターのハードウェアを理解しており、それをどのように使用すれば複数階層のアプリケーションを最も効果的に実行できるかを知っています。
- **環境の所有者**は、特定のデプロイメント環境内のハードウェアとソフトウェアに対して責任を負っています (例: 開発、QA、ユーザー受け入れテスト、運用前、運用など)。
- **デプロイメントスペシャリスト**は、関連するライフサイクル内の環境にアプリケーションをデプロイする責任を負っています。
- **アプリケーションデプロイメント管理者は、**アプリケーションをデプロイするために使用される、環境、階層、ライフサイクル、スクリプト、コードコンポーネントソースタイプの指定に責任を負っています。

次の関係者にとっても、アプリケーションデプロイメントの動作の仕組みを理解することは利益があります。

- **SA管理者**は、HPE Server Automationのすべての管理作業の責任を負っています。その役割は、ユーザーの役割ごとの権限とアクセス権を制御し、SAによって管理されるサーバーを決定することです。また、SAのインストールと更新の責任を負う場合もあります。
- **開発チーム**のメンバーとマネージャーは、最終的に運用環境にデプロイされるアプリケーションを設計し、実装します。
- **QAチーム**のメンバーとマネージャーは、アプリケーションがどのようにライフサイクルを通過し、QA環境にどのような影響があるかを理解する必要があります。これらのユーザーは、アプリケーションデプロイメントを使用して、アプリケーションをデプロイし、テストすることができます。

前提条件

アプリケーションデプロイメントはServer Automation (SA) の一部です。SAの前提条件については、『SAインストールガイド』を参照してください。

本ガイドに記載されている各種の操作を実行するには、適切なアクセス権が必要です。詳細については、「アクセス権の設定」(586ページ)を参照してください。

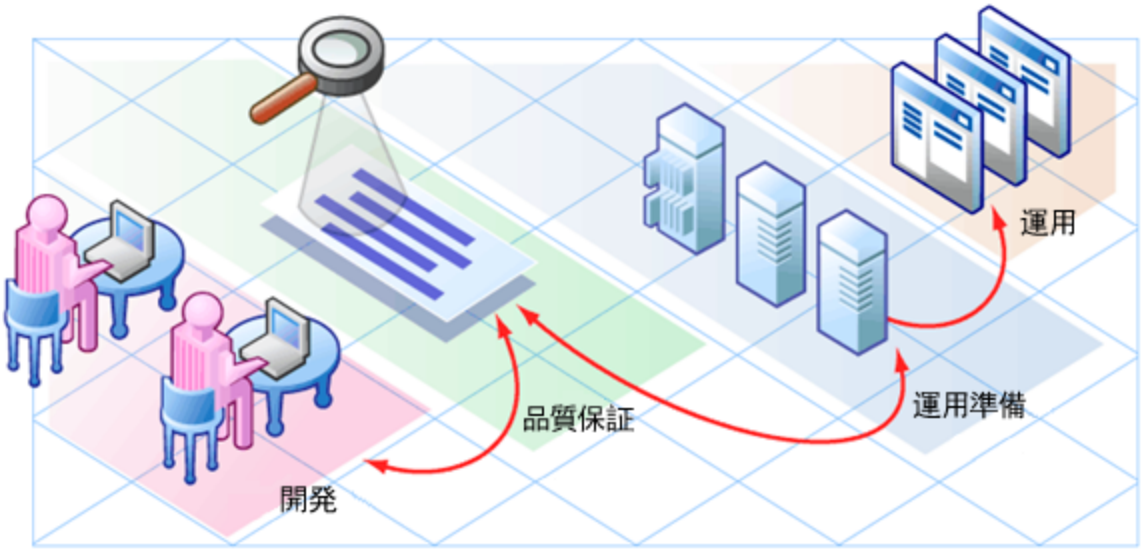
アプリケーションデプロイメントユーザーインタフェースには、Adobe Flash Playerバージョン9.0 (またはそれ以降) が必要です。

概要

アプリケーションデプロイメントの自動化が重要である理由

カスタムアプリケーションの作成、デプロイ、およびテストは複雑なプロセスであるため、ソフトウェア開発ライフサイクルが必要です。ソフトウェア開発ライフサイクルは、関連するさまざまな活動の枠組みとなります。以下に簡単なソフトウェア開発ライフサイクルの例を示します。

一般的なソフトウェア開発ライフサイクル



このソフトウェア開発ライフサイクルでは、アプリケーションは開発から品質保証 (QA) に移されます。QAではアプリケーションをテストします。通常、QAと開発の間でテストと修正が何度か繰り返されます。QAでアプリケーションが承認されると、(初めてのデプロイか、アップグレードかに関わらず) アプリケーションをデプロイして、アプリケーションが (運用環境とよく似た) 運用前環境で動作することを確認できます。このデプロイメントは、パフォーマンステストの実行に使用されることがあります。これが成功して初めてアプリケーションは運用環境に移されます。

このプロセスを手動で実行した場合、時間がかかり、ミスが発生しやすくなります。問題を難しくしているのは、スタッフとプロセスの両方が細分化されていることです。たとえば、アプリケーションに必要なコンポーネントを地理的に離れた複数チームが開発する場合があります。QAはQA環境を管理できますが、運

用前環境と運用環境を管理するのはシステム管理者です。このため、多くのコミュニケーションや調整が必要になります。

他にも次のような課題があります。

- 複雑な環境によってプロセスが複雑化する。
- 開発がリリースをデプロイするために必要な手順を記述する一貫した方法が通常存在しない。
- QAは環境を作成するために必要なマシンを保有する必要がある。QAはこれらの環境の作成と撤収を繰り返し行う必要がある。
- アプリケーションを最新の運用レベルに対応させるため、複数のデプロイメントを実行しなければならない場合がある。
- 2つの環境で使用されるプロセスに一貫性がないため、QAで正常に動作した製品が運用前環境では動作しないことがある。
- 運用前環境では運用環境を正確に再現する必要がある。
- コードが変わるたびに、ソフトウェア技術者がコードをパッケージし直す必要がある。
- 運用マシンの担当者を配置して、更新が必要な内容を把握する必要がある。
- リリースマネージャーとチームが、リリースの現在の状態を統合的に情報交換する場が存在しない。
- 監査、報告、およびメトリックスが存在しないため、必要な修正を組織が把握することができず、プロセスが改善されない。
- 開発と運用のニーズが競合するため、セキュリティ上の問題が発生する。

HPE Server Automationのアプリケーションデプロイメントでは、単一のアクセスポイントから、すべてのユーザーがそれぞれの役割に応じて関連データを表示または入力できるので、アプリケーションのデプロイメントに伴う複雑なコミュニケーションを簡略化できます。

また、アプリケーションデプロイメントはHPEソフトウェアの他のテクノロジーと連携することで、アプリケーションを運用環境に確実に移行できます。

アプリケーションアーキテクチャー

アプリケーションは、次のような各種**コンポーネント**で構成されています。

- コードコンポーネント (例: HTMLファイル、JARファイル、WARファイル、.NETアセンブリ、データベース)
- スクリプト
- 構成ファイル
- SAアプリケーション構成

- SAソフトウェアポリシー
- SAライブラリのパッケージ
- HPE Operations Orchestration (OO) フロー
- Windowsレジストリ設定

これらのコンポーネントは、Webサーバー、アプリケーションサーバー、データベースなどの該当する階層にデプロイされます。

2つの階層を含む単純なアプリケーション

1つのアプリケーションには複数の**リリース**が存在できます。また、通常、1つのリリースには複数の**バージョン**が存在します。バージョンとはある時点におけるリリースの不変の「スナップショット」です。アプリケーションを環境にデプロイする場合は、特定のバージョンをデプロイします。

リリースに含まれるコンポーネントを指定する際には、各コンポーネントの**ロールバック**手順と**アンデプロイ**手順の両方を指定できます。

ロールバックが有効である場合、アプリケーションデプロイメントはデプロイを行う前にバックアップを実行します。デプロイメント中にエラーが発生した場合、ロールバック手順を使用してターゲットを以前の状態に戻します。この場合、アプリケーションデプロイメントは手順を後戻りし、ロールバック手順に従ってデプロイメントの各手順を元に戻します。

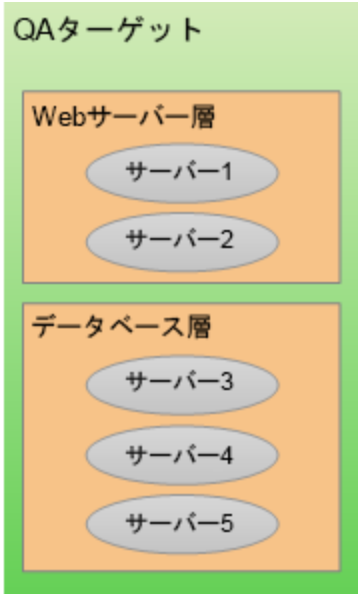
アンデプロイが有効である場合、アンデプロイ手順を使用してアプリケーションをアンインストールします。

アプリケーション**グループ**を作成することで、複数のアプリケーションを最も便利な方法で整理することができます。これはもっぱらユーザーの便宜のために提供されている整理ツールであり、アプリケーションのデプロイメントには影響しません。

運用アーキテクチャー

運用側には、アプリケーションをデプロイする**ターゲット**が存在します。各ターゲットはそれぞれの構造に応じて1つ以上のアプリケーションをサポートするのに適しているのが一般的です。ターゲットは**階層** (アプリケーションサーバー階層やデータベース階層など) を使用して構成されます。各ターゲットには**環境** (QAや運用など) が関連付けられます。環境の作成が完了すると、ソフトウェア開発**ライフサイクル**に環境がどのようにマップされているかを参照できます。

2つの階層を含むターゲット



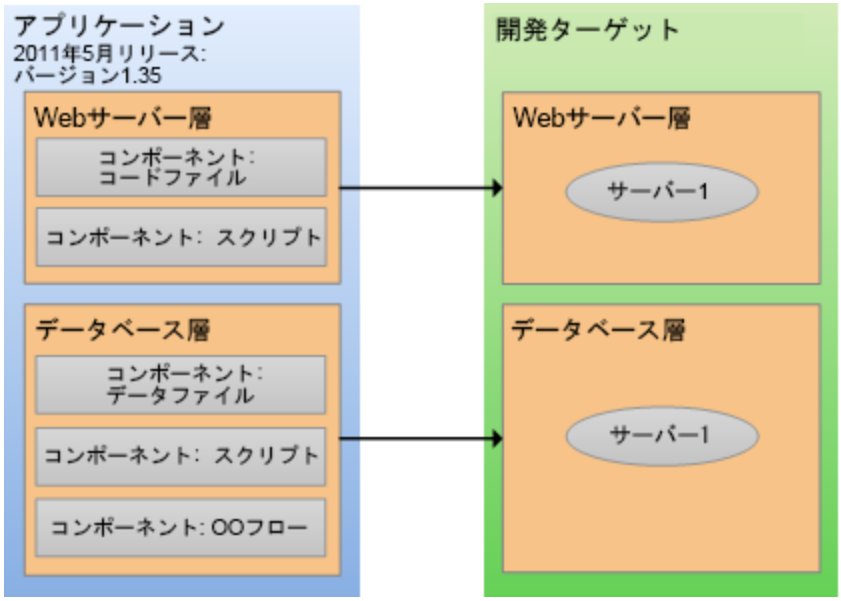
アプリケーションアーキテクチャーと運用アーキテクチャーの連携

階層はアプリケーションをターゲットに結び付けるために使用する仕組みです。アプリケーションをデプロイするには、各階層に含まれるコンポーネントを同じ階層に属するターゲットサーバーにマップします。ターゲットの規模や構成に関係なく、アプリケーションの階層は変わりません。

以下では、単純なアプリケーション (パッチなど) が開発環境からQA環境、そして最終的に運用環境へと進んでいく例について考えます。

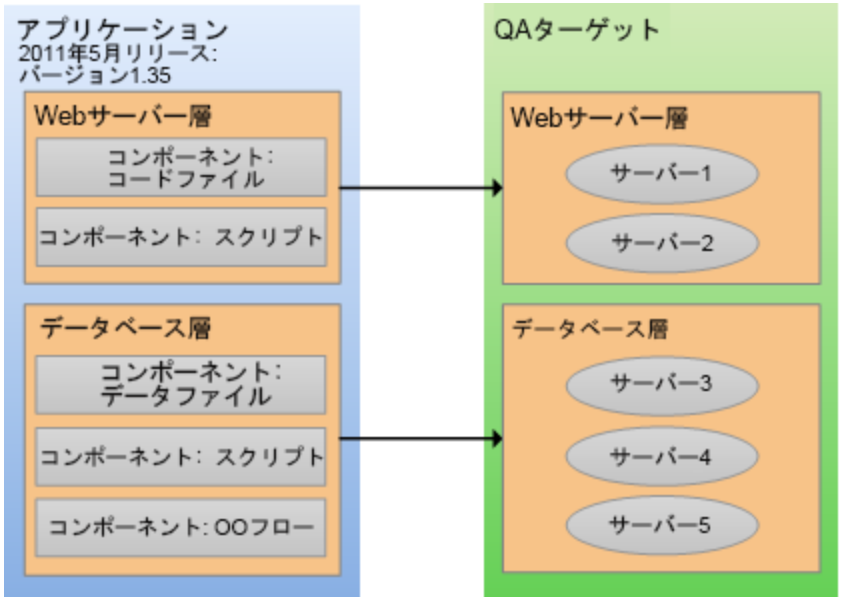
開発フェーズでは、アプリケーションは頻繁に変更され、数多くのバージョンが小規模なテスト環境にデプロイされます。この場合、同じサーバーでWebサーバーとデータベースの両方が稼働しています。

小規模な開発ターゲットへのデプロイ



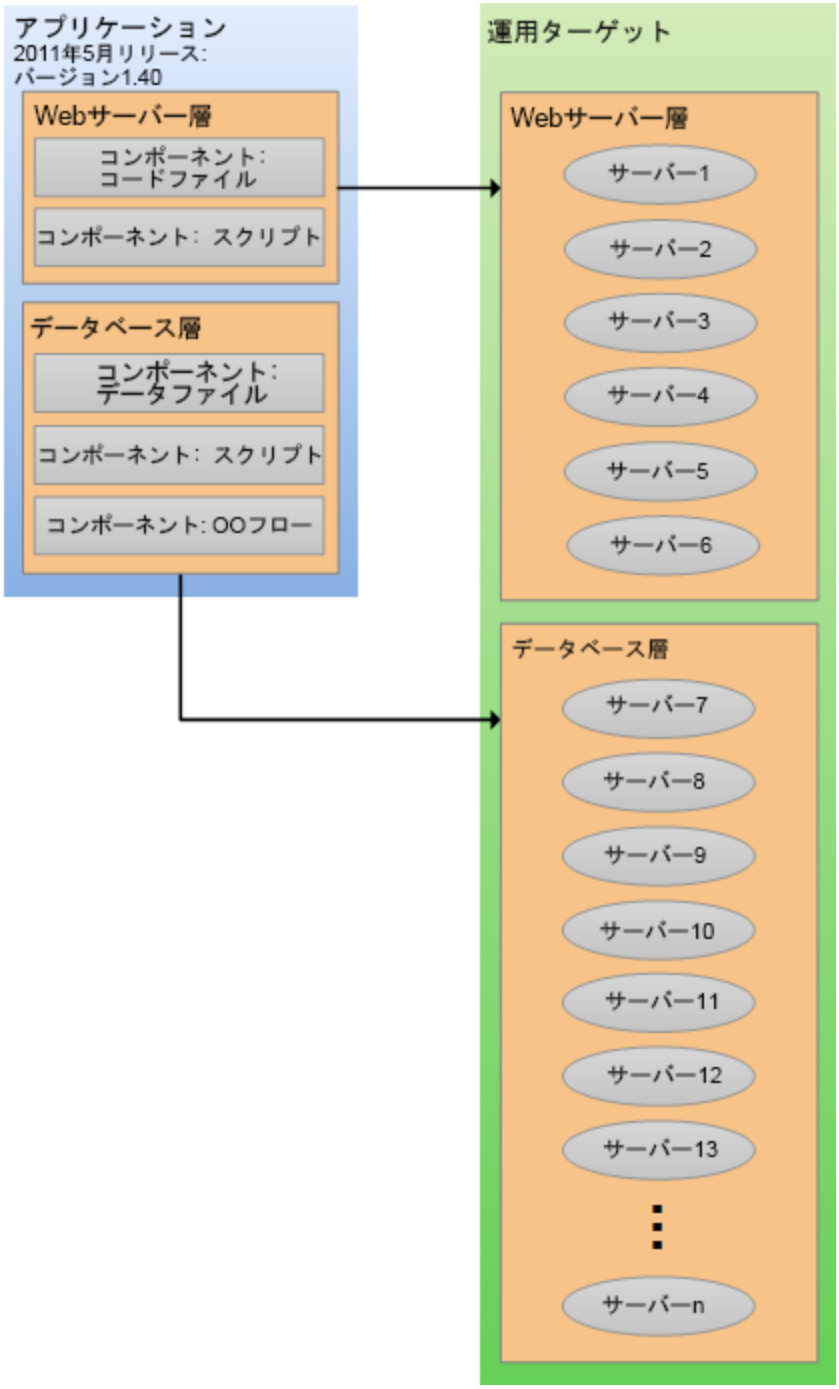
アプリケーションがQA環境に進むと、反復回数は少なくなります。環境の規模はやや大きくなります。QAフェーズでは、問題を検出して修正を行うため、アプリケーションを開発環境とQA環境の両方にデプロイするのが一般的です。

やや大きなQAターゲットへのデプロイ



アプリケーションが安定して全面的なテストが済むと、アプリケーションは運用環境に進みます。通常、環境の規模はこれまでよりもかなり大きくなります (上の図を参照)。

運用環境へのデプロイ



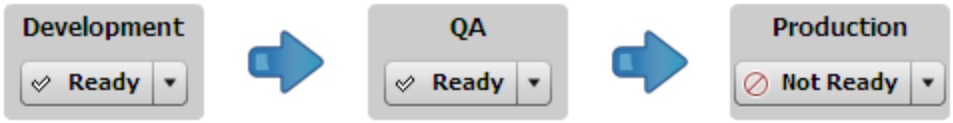
この例では、運用データベースはOracle Real Application Clusters (RAC) などのサーバークラスターにホストされています。

アプリケーションのソフトウェアライフサイクルの遷移

アプリケーションデプロイメントではカスタマイズしたライフサイクルを作成できるため、開発フェーズから運用環境における完全なデプロイメントに至るまで、ライフサイクルの進行に合わせてアプリケーションをきめ細かに管理することができます。

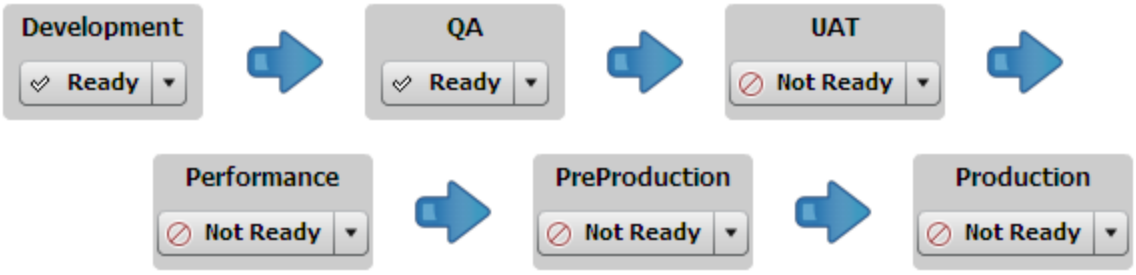
ライフサイクルは一連の環境で構成されます。それぞれの環境はサーバーのグループで構成されます。用途に応じて異なるライフサイクルを使用できます。小さなアプリケーションやマイナーパッチなどの場合は、次のようなシンプルなライフサイクルを使用できます。

パッチライフサイクル



新製品やメジャーアップグレードなどの複雑なアプリケーションの場合は、次のようなより複雑なライフサイクルを使用します。

完全ライフサイクル



アプリケーションデプロイメントは、ライフサイクルを使用することで、アプリケーション開発と運用の間の遷移をシームレスに確実に行えるようにします。

ライフサイクルの初期では、通常、アプリケーションは頻繁に変更されます。たとえば、ナイトリービルドごとに新規バージョンを作成し、それを開発環境のサーバーにデプロイします。

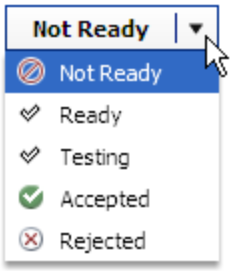
アプリケーションが安定化してきたら、正式なQAテストを行うことができます。この時点では、たとえば、週に2回、QA環境の各サーバーに最新のビルドをインストールします。

アプリケーションがライフサイクルを進むにつれて、変更の頻度が少なくなり、必要なデプロイメントの数も少なくなります。最終的に、少数のリリース候補版に絞り込まれ、ユーザー受け入れテスト (UAT) やパフォーマンステスト用にデプロイされます。

UATに合格し、許容できるパフォーマンスが実証されたら、リリース候補版を運用前環境にデプロイ (最終的には運用環境にデプロイ) できます。

SAでは、各企業の基準に従って、さまざまなライフサイクルに含める環境が特定されます。

環境ステータスの設定



環境には環境ごとのステータスがあります。アプリケーションデプロイメントではステータスが「Not Ready」の環境にはデプロイできません (この場合、環境内のターゲットを選択できません)。

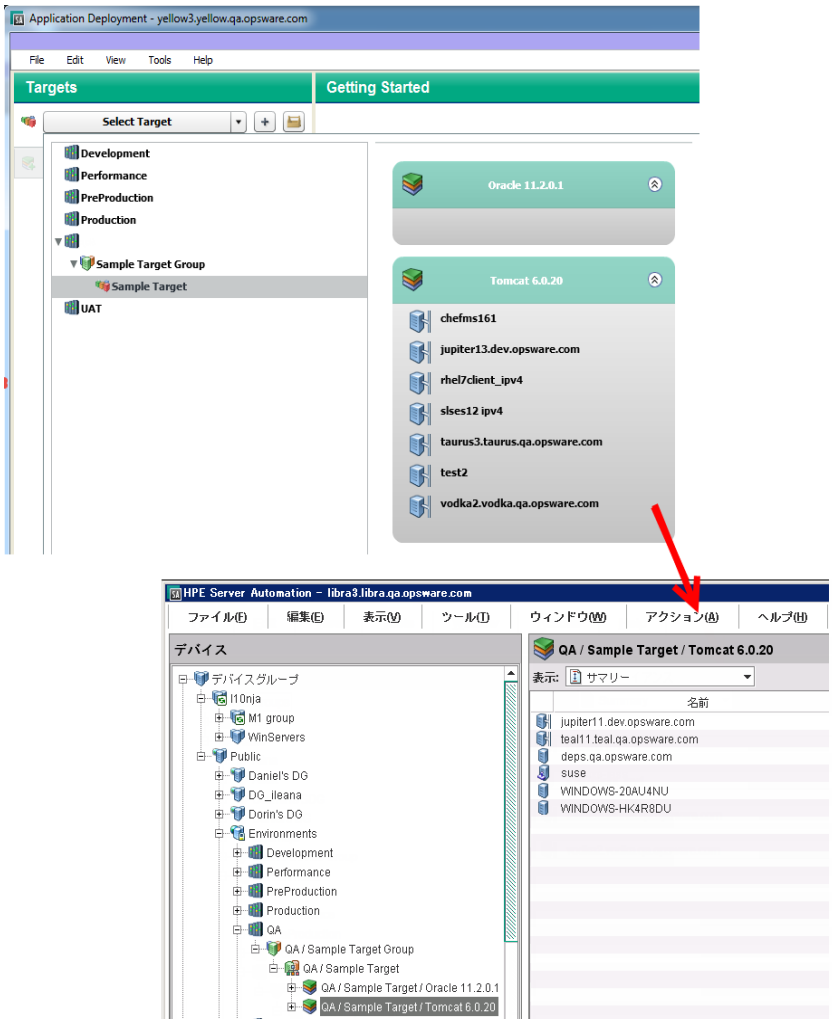
ステータスフィールドを使用すると、アプリケーションをデプロイできる場所を制御できます。各環境の初期設定はアプリケーションデプロイメント管理者によって設定されます。ユーザーに特定の環境にデプロイするアクセス権がある場合、このユーザーはデプロイメント時に環境のステータスを変更できます。

どのユーザーがどの環境にアプリケーションをデプロイするアクセス権を持つかは、アプリケーションデプロイメント管理者が決めます。詳細については、「アクセス権の設定」(586ページ)を参照してください。

環境とデバイスグループ

アプリケーションデプロイメントにおける環境はSAのデバイスグループとして反映されます。

アプリケーションデプロイメントおよびSAの表示



アプリケーションデプロイメントの環境は、[デバイス] 画面の [Environments] デバイスグループの下に配置されます。これらのデバイスグループは読み取り専用 (ユーザーによる変更は不可) で、アプリケーションデプロイメントによって管理されます。

[Environments] の下の各デバイスグループには、次のように、アプリケーションデプロイメントオブジェクトのタイプを示す特別なアイコンが表示されます。

- 環境
- ターゲット
- 階層

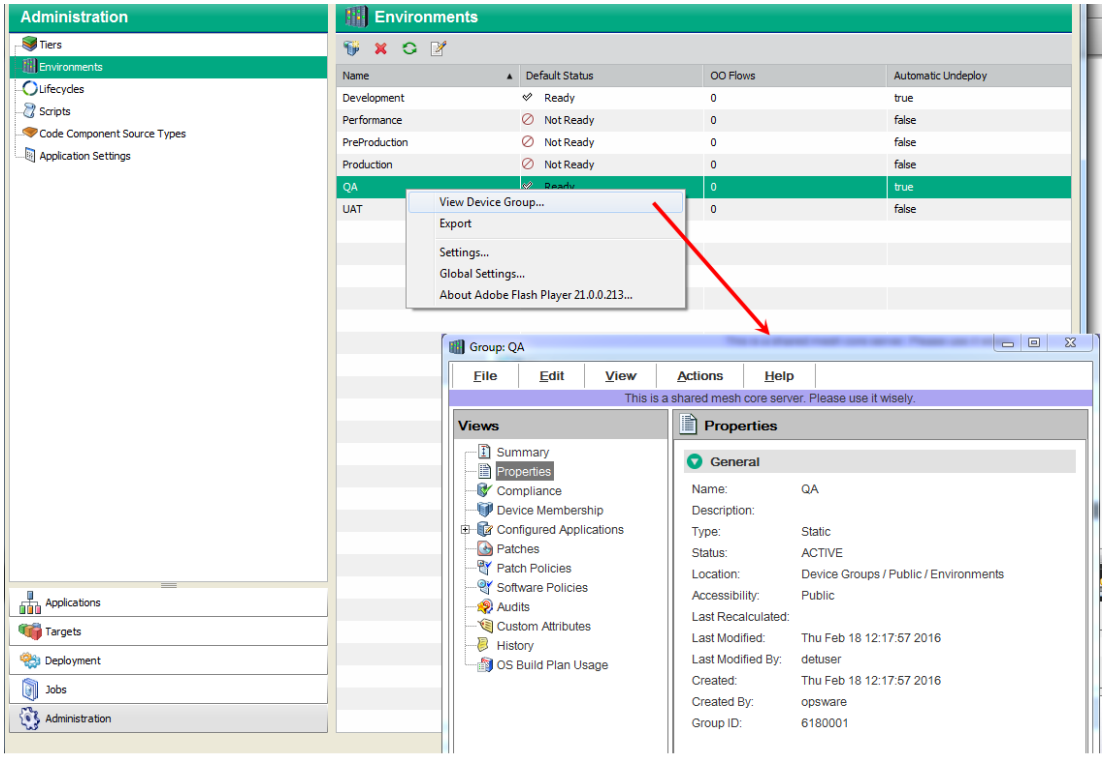
ターゲットおよび階層のデバイスグループ名は、それぞれ親の環境およびターゲットを表すパスになります。

アプリケーションデプロイメントのユーザーインタフェースで環境またはターゲットに変更を加えると (たとえば、ターゲットの名前を変更すると)、対応するデバイスグループにも変更が反映されます。これには時間

がかかることがあり、この間に「同期中」のメッセージが表示されることがあります。また、変更を表示するのに、SAのメインのユーザーインタフェースで更新の実行が必要になる場合があります。

環境所有者またはアプリケーションデプロイメント管理者は、以下に示すように、右クリックのショートカットメニューを使用してデバイスグループウィンドウを開くか、即時の同期を強制的に実行することができます(「環境の管理」(604ページ)を参照)。

環境に対するデバイスグループウィンドウ



例:環境の準備とアプリケーションのデプロイ

オンラインストアを開設しようとしている場合を想像してください。サイトをできるだけ早く公開したいと考えています。注文を処理するアプリケーションはまだ開発中ですが、QAに移行する準備はほぼできています。

アプリケーションには、Apache 2.2.9 WebサーバーとOracle 11gデータベースの2つの階層が含まれています。アプリケーションは、ソースコード管理システムに保管された複数のコンポーネントで構成されています。

チームは次のような手順に従って、アプリケーションデプロイメントを使用して新しいアプリケーションをテスト用のQA環境に移行します。

1. アプリケーションデプロイメント管理者が必要なアクセス権を設定します。たとえば、
   - アプリケーションエキスパートには、アプリケーションを編集し、アプリケーションを開発環境にデプロイするアクセス権が必要です。
   - デプロイメントスペシャリストには、アプリケーションを表示し (編集はしない)、アプリケーションをQA環境にデプロイするアクセス権が必要です。
2. アプリケーションエキスパートが、関連するコンポーネントを作成し、これらをWebサーバー階層とデータベース階層に割り当てて、アプリケーションアーキテクチャーを定義します。
3. QA環境所有者がアプリケーションアーキテクチャーをレビューし、このアプリケーションに必要なQA環境を準備します。
   a. 最初に「Store Orders」というターゲットを作成します。ターゲットには、アプリケーションアーキテクチャーで指定されたものと同じ階層が含まれている必要があります。
   b. 次に、各階層に属するサーバーを指定します。これらのサーバーはすでにSAで管理されている必要があります。これでオンラインストアのQA環境の準備ができます。
4. ソフトウェア技術チームがアプリケーションの初期バージョンをテストする準備ができたと判断すると、デプロイメント用のバージョンを作成します。次に、アプリケーションをテストする準備ができたことをQAに通知します。
5. デプロイメントスペシャリストがアプリケーションをQA環境にデプロイし、QAエンジニアがテストを開始します。

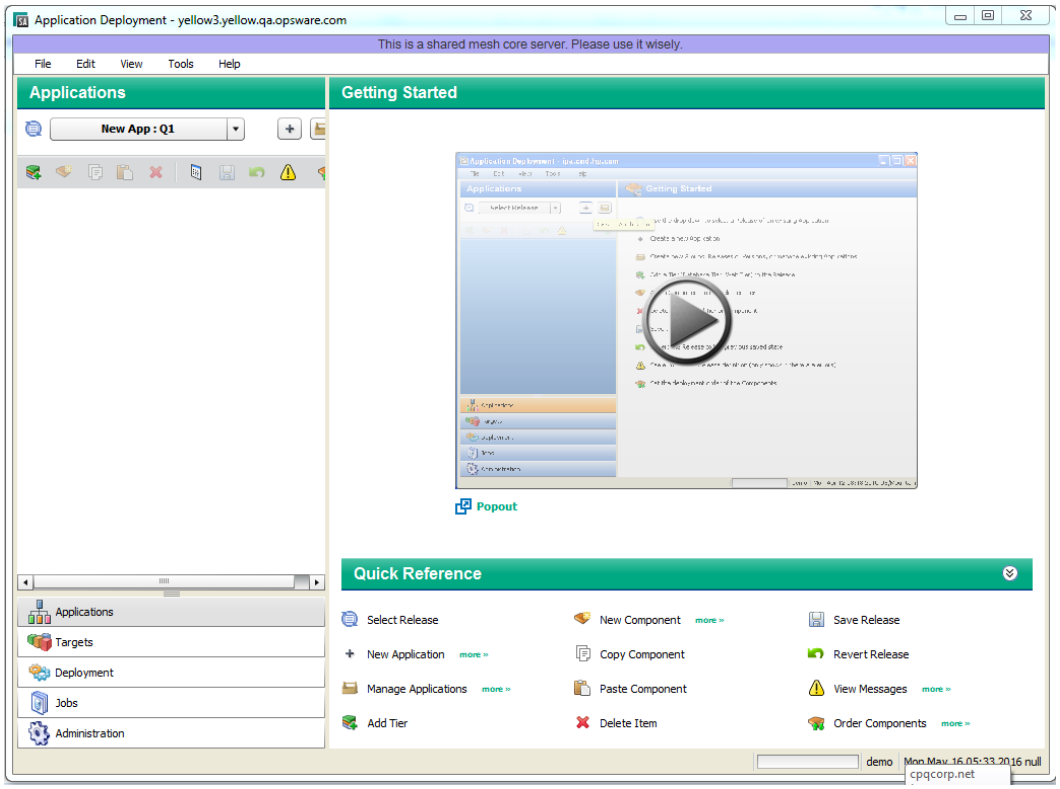
ここでの目的は、開発とQAの間を行き来する回数を最小限に抑えてテストプロセスを進めることです。デプロイメントを自動化することで、反復回数が少なくなります。

アプリケーションがQAサイクルを完了し、運用前フェーズに移行して合格し、最終的に運用フェーズに移行すると、オンラインストアをオープンすることができます。

アプリケーションデプロイメントのユーザーインタフェースについて

アプリケーションデプロイメントを初めて開いたとき、および新しいアプリケーションを作成するときには、次のようなユーザーインタフェースが表示されます。

アプリケーションデプロイメントのインタフェース – 初期表示



この初期表示には、アプリケーションデプロイメントのユーザーインタフェースを理解するために役立つ2つのツールが存在します。

- [作業の開始] 領域には、この画面の機能について説明した短いデモビデオが表示されます。

  ボタンをクリックすると、ビデオが始まります。別のウィンドウでビデオを再生する場合は、Popout をクリックします。

- [クリックリファレンス] 領域には、この画面で使用できるツールバーボタンの機能が表示されます。

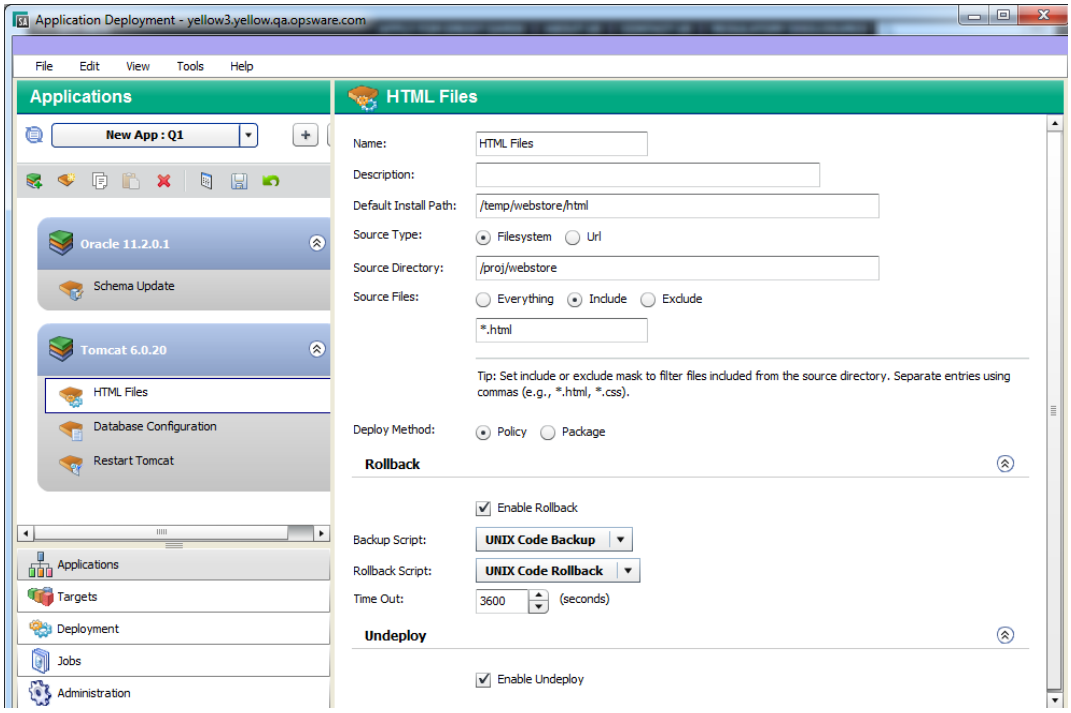
  各機能の上にマウスポインターを置くと、ツールヒントが表示されます。[詳細>>] リンクをクリックすると、関連するオンラインヘルプトピックが別のブラウザーウィンドウで開きます。

左下にあるボタンを使用すると、アプリケーションデプロイメントユーザーインタフェースの別の機能領域が開きます。強調表示されているボタンは、作業中の領域を示しています。別の領域に移動する際には、変更を保存するか、変更内容を元に戻す必要があります。

> **注:** アプリケーションデプロイメントには、すぐに使用できるサンプルアプリケーションとサンプルターゲットが付属しています。

アプリケーションに階層を追加し、階層内にコンポーネントを作成すると、インタフェース表示は次のようになります。

アプリケーションデプロイメントのインタフェース – 2つの階層を含むアプリケーション



左側のペインには、構造ビューとナビゲーションコントロールが表示されます。

右側のペインには、左側で選択したアイテムの内容が表示されます。この例では、Tomcat 6.0.20階層のHTML Filesコンポーネントが選択されています。

命名規則

次の規則は、すべてのアイテムの名前プロパティに適用されます。

- 名前の先頭と末尾にスペースを使用しないこと。
- 名前の先頭に句読点を使用しないこと(つまり、必ず英字または数字で始まる名前にすること)。
- 名前に次のいずれかの文字を含めないこと(改行、タブ、スラッシュ (/)、バックスラッシュ (\))。

アプリケーションデプロイメントのユーザーインタフェースでは、これらの命名規則に違反することはできません。名前がこれらのルールに準拠していない場合、[名前] ボックスの周りに赤いボックスが表示されます。

テキストボックス内の情報の変更

テキストボックス(右側のペインまたはダイアログ)の内容を変更する際には、次の点に注意してください。

- テキストボックスを選択すると、テキストボックスの周りに青い枠が表示されます。
- 「命名規則」(438ページ)に違反すると、テキストボックスの周りに赤いボックスが表示されます。

Ctrl+C、Ctrl+X、およびCtrl+Vキーを使用すると、編集可能なテキストボックスでコピー、切り取り、および貼り付けを行うことができます。また、[編集]メニューの対応するアイテムを使用することもできます。

変更の保存および変更を元に戻す

アプリケーションデプロイメントユーザーインタフェースでアイテムを変更すると、[保存]および[元に戻す]ボタンがアクティブになります。

- [**保存**]ボタン  をクリックすると、変更内容がアプリケーションデプロイメントデータベースに保存されます。
- [**元に戻す**]ボタン  をクリックすると、変更内容が破棄され、直近に保存された値がデータベースからリロードされます。

別の画面に移動する際には(アプリケーション画面からターゲット画面など)、変更を保存するか、変更を元に戻す必要があります。

アプリケーションおよびターゲットの管理

アプリケーション画面およびターゲット画面には、アプリケーションおよびターゲットの現在の状態をすばやく評価し、各種タスクを実行するために使用できる「マネージャー」が存在します。アプリケーションマネージャーでは、たとえば、新しいリリースの作成やアプリケーションの名前の変更などを行うことができます。

マネージャーでは、複数のアイテムを選択し、そのすべてで同じ操作を実行できます。たとえば、複数のアプリケーションを選択し、それらを削除することができます。

マネージャーを開くには、左上のドロップダウンリストの右側にある  ボタンをクリックします。

詳細については、「アプリケーションの管理」(460ページ)および「ターゲットの管理」(527ページ)を参照してください。

検証メッセージ

検証エラーが発生すると、ツールバーに「警告」を示す  が表示されます。これはアイテムの構成が不完全な場合にも表示されます。

「警告」を示す  をクリックすると、検出されたエラーの詳細が表示されます。

たとえば、新しいアプリケーションを作成して、まだコンポーネントを追加していない場合は、次の検証エラーがレポートされます。

“No components in release”

検証エラーメッセージは、正しく機能しない操作を実行しようとしたときにも表示されます。たとえば、コンポーネントが不完全なアプリケーションのバージョンを作成しようとすると、次のようなメッセージが表示されます。

“Cannot create version from this release.Please correct errors below: Script Component ‘Sample Script Component’: No content.

Code Component ‘Sample Code component’: Source Directory must be set.

Code Component ‘Sample Code component’: Default Install Path must be set.”

クイックスタート

アプリケーションデプロイメントツールを使用することで、アプリケーション開発者とオペレーションチームがシームレスに連携して、ミッションクリティカルなビジネスアプリケーションのエンジニアリングから運用への移行を管理することができます。アプリケーションデプロイメントを使用すると、複雑な多階層のJava EEおよび.NETアプリケーションを定義してデプロイすることができます。

SAクライアントからアプリケーションデプロイメントツールを実行するには、[**ツール(T)**] > [**アプリケーションデプロイメント**] を選択します。

アプリケーションを定義する際には、次のことが可能です。

- アプリケーションの定義をすばやく作成し、複数の同時リリースを管理できます。
- 階層を使用してアプリケーションの構造を定義できます。

- ソフトウェアポリシーを使用して、アプリケーションに必要なミドルウェアがアプリケーションとともにすでにデプロイされていることを保証できます。

新しいコンポーネントタイプ、ソフトウェアライブラリオブジェクト、および(オプションで)HPE Operations Orchestration (OO)フローを使用して、アプリケーションの内容を制御できます。

OOフローおよびその他のコンポーネントタイプに対するアプリケーションデフォルトパラメーターを指定できます。これらのパラメーターは、必要な場合、デプロイメント時に異なる環境に合わせて調整することができます。たとえば、アプリケーションの適切なデータベース接続設定は、QAと運用で異なる可能性があります。

ターゲットを定義する際には、次のことが可能です。

特定のアプリケーションをサポートするために連携して機能するサーバー、インスタンス、データベースの集合を定義できます。

アプリケーションと同じ階層の集合を使用してターゲットの構造を指定できます。

ソフトウェアポリシーを使用して、アプリケーションに必要なミドルウェアがアプリケーションとともにすでにデプロイされていることを保証できます。

ターゲット階層に管理対象サーバー、インスタンス、データベースを設定することができます。

ターゲット階層にOOフローを関連付けることができます。

>このクイックスタートでは、シンプルなアプリケーションの作成とデプロイの方法を示します。

「概要」(441ページ)

「クイックスタート」(440ページ)

「クイックスタート」(440ページ)

「クイックスタート」(440ページ)

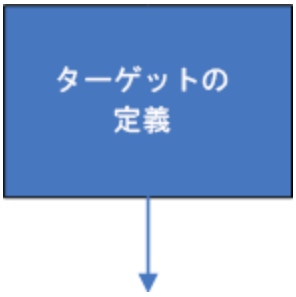
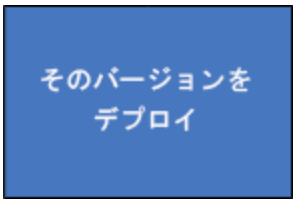
「クイックスタート」(440ページ)

このクイックスタートでは、Server Automation (SA)がインストールされており、SAクライアントを実行できることを前提としています。詳細については、『SA 10.50インストールガイド』を参照してください。

概要

アプリケーションデプロイメントは、アプリケーションのデプロイのプロセスを自動化します。このクイックスタートチュートリアルでは、このツールを初めて使用する場合に役立つ情報を提供します。

基本的なステップは次のとおりです。

	アプリケーションには、1つ以上の**リリース**があります。 各リリースは、**階層**から構成されています。たとえば、アプリケーションサーバー階層、Webサーバー階層、データベース階層などです。 各階層には、1つ以上の**コンポーネント**があります。これらは、アプリケーションの内容を表し、その機能を実装します。 詳細については「ステップ1: アプリケーションの定義 - 階層とコンポーネント」(442ページ)を参照してください
	ユーザーは、リリースの**バージョン**を作成し、そのバージョンをデプロイします。 各バージョンは、リリースの変更不可能なスナップショットです。 バージョンが変更不可能なのは、QAでテストされたのと同じコンポーネントが運用にデプロイされることを保証するためです。 詳細については「ステップ2: デプロイするバージョンの作成」(447ページ)を参照してください
	ターゲットは、アプリケーションの**階層**(データベース階層、アプリケーションサーバー階層、またはWebサーバー階層など)をホストする**サーバー**のグループです。 1つの階層が複数のサーバー上で動作する場合もあります。 詳細については「ステップ3: ターゲットの定義」(449ページ)を参照してください
	アプリケーションの各階層のコンポーネントは、対応する階層を持つ環境内の選択されたターゲットにデプロイされます。 詳細については「ステップ4: ターゲットへのバージョンのデプロイ」(452ページ)を参照してください

上記のリンクをクリックすると、基本的な手順が表示されます。本書のこの後の部分では、これらの作業についてさらに詳しく説明します。

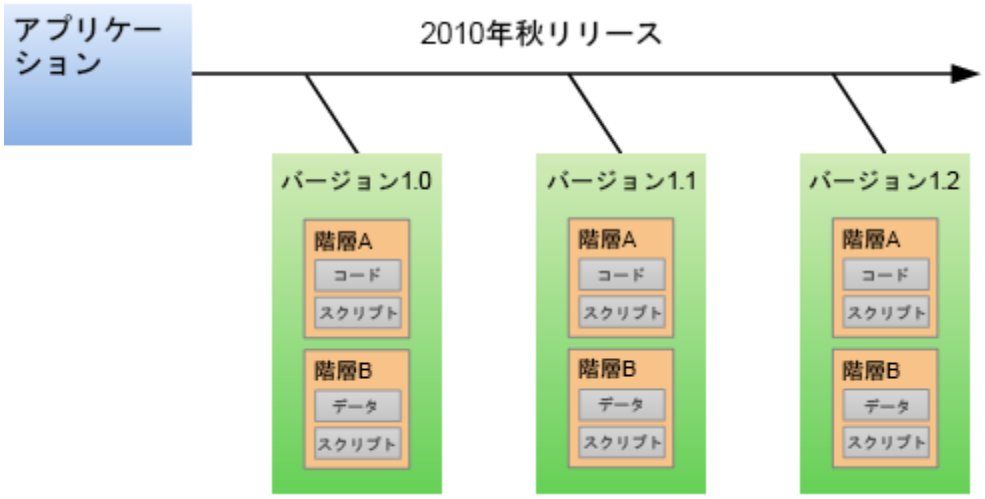
ステップ1: アプリケーションの定義 - 階層とコンポーネント

アプリケーションは、1つ以上の**階層**から構成されます。各階層には、アプリケーションの機能を実装する**コンポーネント**が含まれます。

アプリケーションは、1つ以上の**リリース**の形で存在します。各リリースには、複数の**バージョン**が存在できます。アプリケーションをデプロイする際には、各階層のコンポーネントから構成されるバージョンを作成します。バージョンとは、特定の時点でのリリースの変更不可能な「スナップショット」です。

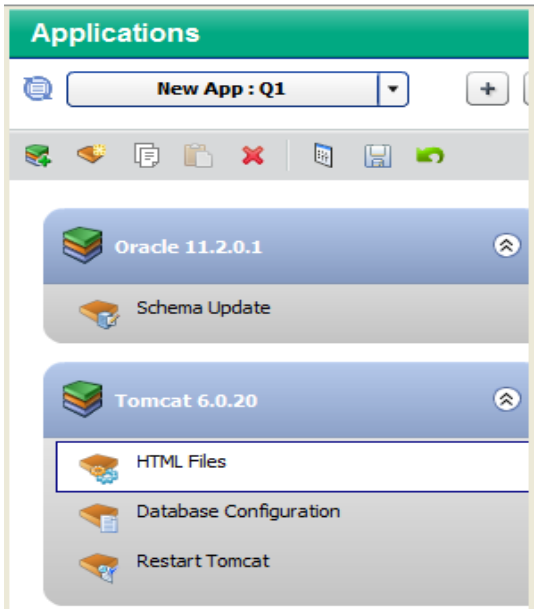
次の図に示すアプリケーションは、2つの階層から構成され、Fall 2010リリースの3つのバージョンがあります。

アプリケーションの構成



たとえば、次のアプリケーションには、Oracle 11.2.0.1データベース階層と、Tomcat 6.0.20アプリケーションサーバー階層があります。Tomcat階層のコンポーネントは、HTMLファイル、構成ファイル、再起動スクリプトです。Oracle階層のコンポーネントはデータベーススクリプトです。

2階層のアプリケーションの例

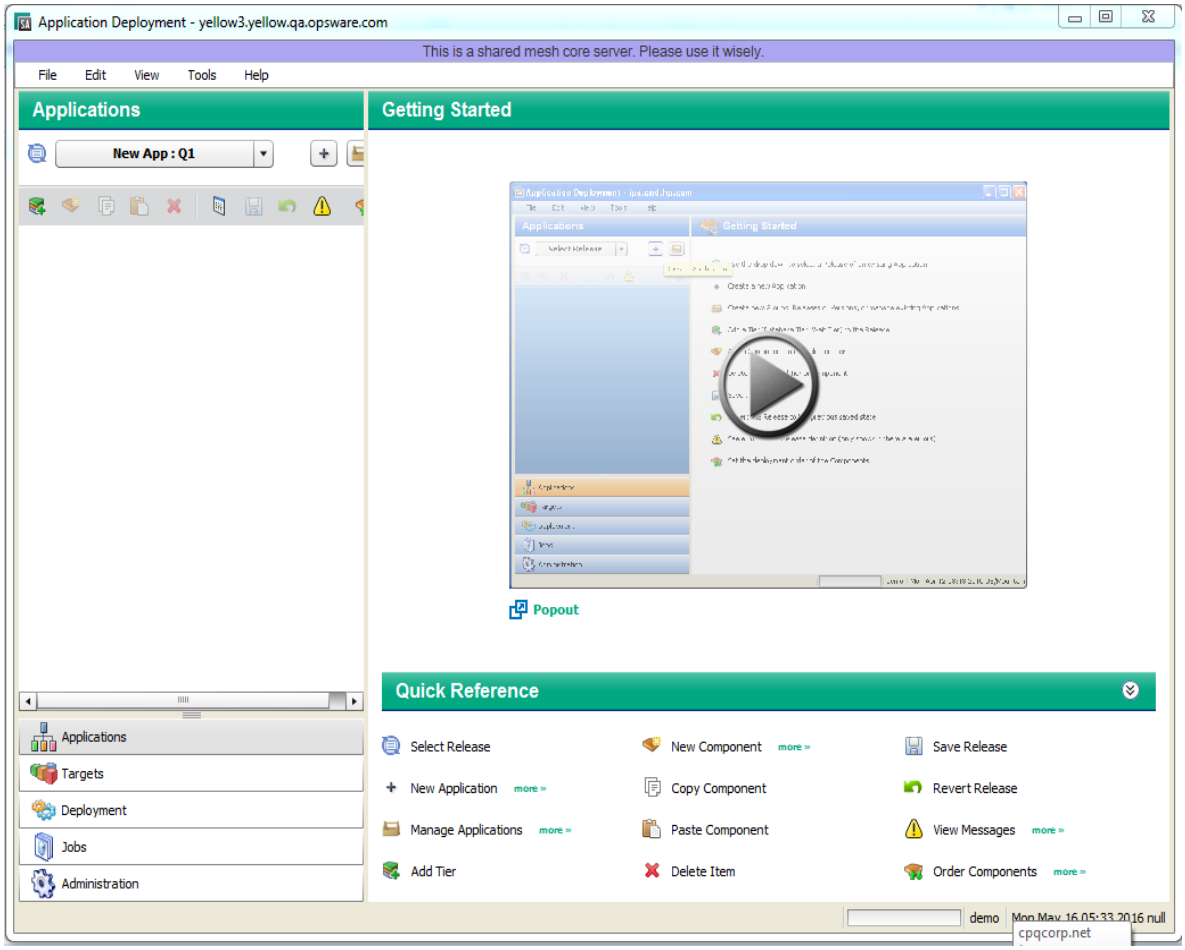


以下の手順では、新しいアプリケーションを作成する場合を仮定しています。既存のアプリケーションを操作する場合は、「アプリケーションの概要」(454ページ)を参照してください。

アプリケーションを作成するには、SA管理者から、アプリケーションデプロイメントツールへのアクセス権とアプリケーション作成のアクセス権を付与されている必要があります。「アクセス権の設定」(586ページ)を参照してください。

アプリケーションデプロイメントツールを起動する方法

1. SAクライアントを起動します。詳細については、『SA 10.50ユーザーガイド』を参照してください。
2. SAクライアントで、[**ツール**] > [**アプリケーションデプロイメント**] メニュー項目を選択します。アプリケーションデプロイメントインタフェースが開き、図のように [Applications] 画面が表示されます。



ボタンをクリックすると、[Applications] 画面の機能を説明した短いビデオデモが始まります。別ウィンドウでビデオを再生するには、Popoutをクリックします。

[Quick Reference] 領域には、この画面で使用できるツールバーボタンの機能が表示されます。[more>>] リンクをクリックすると、関連するオンラインヘルプトピックが別のブラウザーウィンドウで開きます。

新しいアプリケーションを作成する方法

1. [Applications] 画面に移動します (左下の [**Applications**] をクリックします)。
2. [**Create a New Application**]  をクリックします。[Create New Application] ダイアログが開きます。アプリケーションを作成するには、SAの対応するアクション権限が、ユーザーが属するグループに割り当てられている必要があります (「アクセス権の設定」(586ページ)を参照)。
   a. 新しいアプリケーションの名前を入力します。
   b. オプション: 新しいアプリケーションを既存のアプリケーショングループに割り当てます。
   c. オプション: 新しいアプリケーションの説明を入力します。
   d. [**Release Name**] フィールドに、このアプリケーションで操作している現在のリリースの名前を入力します。

      たとえば、四半期ごとのリリースの場合には、Q3-2010、Q4-2010、Q1-2011、Q2-2011といったリリース名を使用できます。
   e. このリリースのライフサイクルを選択します。ライフサイクルは、アプリケーションが開発から運用まで移行する過程で通過するフェーズを指定します。

      たとえば、一般的なライフサイクルとして、開発 > QA > 運用前 > 運用が挙げられます。これはデフォルトの標準ライフサイクルです。
   f. [**OK**] をクリックしてアプリケーション情報を保存します。
3. [**Add Tier**]  をクリックして、1つ以上の階層をアプリケーションに追加します。
   a. 追加する階層のタイプを選択します。

      ソフトウェアには、標準の階層のグループが付属しています。必要な階層が利用できない場合は、SA管理者に追加を依頼してください。「アプリケーションデプロイメントの管理」(600ページ)を参照してください。
   b. [**Add**] をクリックします。
   c. 必要な追加の階層のそれぞれに対して、ステップ (a) と (b) を繰り返します。

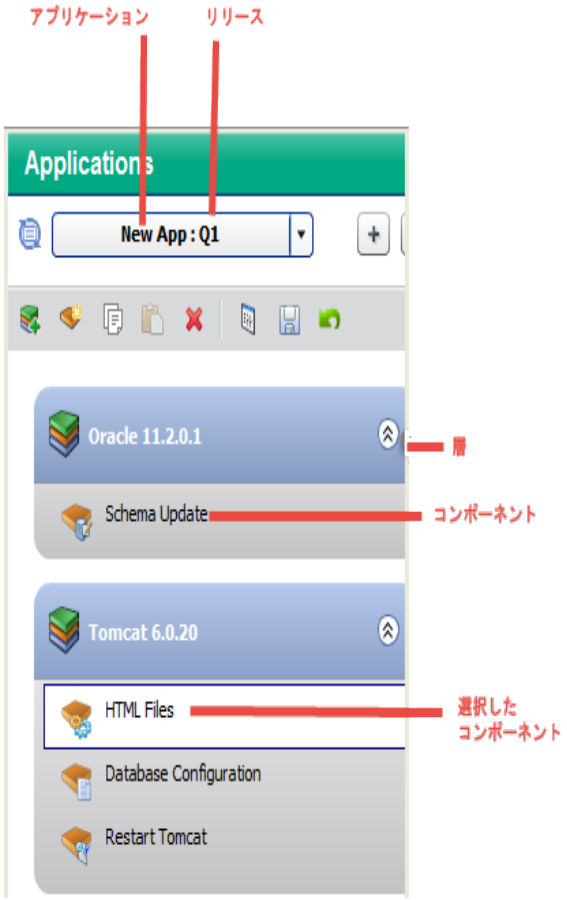
4. 階層にコンポーネントを追加するには、次の手順を実行します。
   a. 階層を選択します。
   b. [New Component] ボタンをクリックして、この階層にコンポーネントを追加します。
   c. 新しいコンポーネントの名前を入力します。
   d. コンポーネントタイプを選択します。詳細については、「コンポーネント」(469ページ)を参照してください。
   e. [**OK**] をクリックします。
   f. 必要な情報を入力して、新しいコンポーネントの場所と動作を完全に指定します。

      入力する必要がある情報は、追加するコンポーネントのタイプに応じて異なります。コンポーネントのタイプによっては、ロールバックおよびアンデプロイのシナリオのための手順が必要です。詳細については、「コンポーネントのタイプ」(470ページ)を参照してください。
   g. 上記のステップ (b) から (f) までを繰り返して、その他のコンポーネントをこの階層に追加します。
   h. [**Save Release**] をクリックして変更内容を保存します。
5. 上記のステップ3～4を繰り返して、その他の階層を追加し、各階層にその他のコンポーネントを追加します。
6. [**Edit component deployment order**] をクリックします。
7. 緑の矢印キーを使用して、コンポーネントを正しいデプロイメント順序に並べます。詳細については、「コンポーネントのデプロイメント順序の変更」(499ページ)を参照してください。

すべての階層とコンポーネントを追加したら、アプリケーションの定義は完成です。アプリケーションは、次の図に示す例のようになります。

ツールバーに「警告」記号が表示される場合、コンポーネントの定義が不完全であるか正しくないことを示します。たとえば、コードコンポーネントのソースの場所が、存在しないディレクトリを指しているような場合です。
をクリックして、必要な情報または対処について調べます。

3つの階層と7個のコンポーネントを持つアプリケーションの例

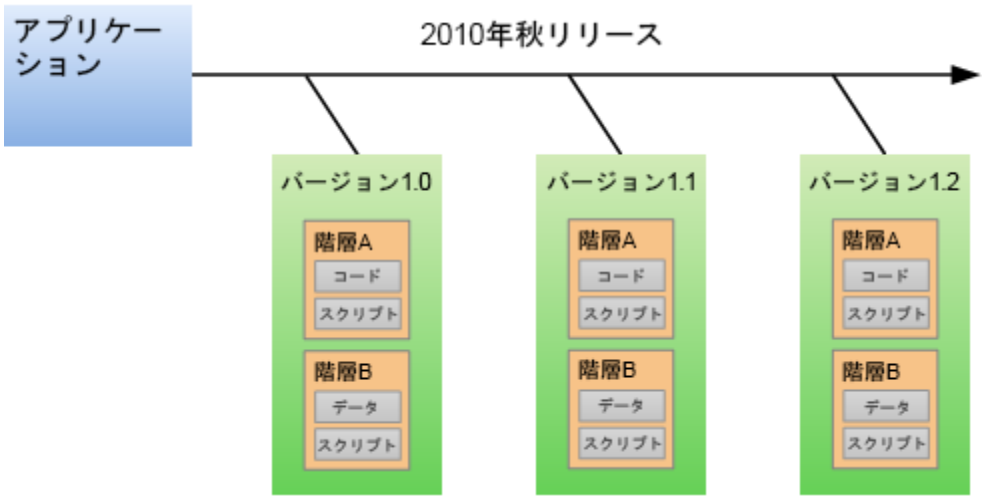


ステップ2: デプロイするバージョンの作成

アプリケーションを定義したら、アプリケーションの新しい**バージョン**を作成する必要があります。その後、そのバージョンを構成するコンポーネントをターゲットにデプロイします。

下の図は、“Fall 2010” という名前のリリースに対して開発中のアプリケーションを示します。このアプリケーションに関しては、バージョン1.0、1.1、1.2の3つのバージョンがビルドされています。これら3つのうち任意のバージョンをターゲットにデプロイできます。

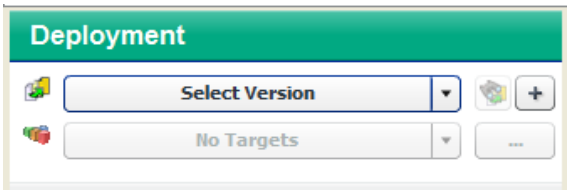
アプリケーション、リリース、バージョン



アプリケーションをデプロイするためには、アプリケーションのバージョンが正常に作成されている必要があります。

アプリケーションの新しいバージョンを作成する方法

1. [Deployment] 画面に移動します (左下で [**Deployment**] をクリックします)。
2. [Select Version] ドロップダウンフィールドで、対象のアプリケーションに移動します。



3. アプリケーションの適切なリリースの下で、[**Create New Version**] リンクをクリックします。
4. [Create New Version] ウィンドウで、バージョン名 (または番号) を入力します。

   ここに指定した文字列がバージョン名になります。このリリースの前のバージョンに番号が含まれている場合は、アプリケーションデプロイメントが自動的にその番号を進めます。たとえば、前のバージョンが "1" だった場合は、[Version] フィールドに "2" が表示されます。前のバージョンが "V 1.2.1" だった場合は、[Version] フィールドに "V 1.2.2" という文字列が表示されます。
5. アプリケーションデプロイメントが [Version] フィールドに入力した内容に関わらず、ユーザーは任意のバージョン名を指定できます。
6. オプション: [Description] フィールドに、このバージョンに対して指定する追加情報を入力します。

7. オプション: コードコンポーネントに含まれるファイルのリストを表示するには、**[Show Code Component Changes]** を選択します。
8. **[Create]** をクリックします。これにより、このアプリケーションのコンポーネントであるすべてのアーティファクトが収集され、アプリケーションの新しいバージョンが作成されます。
9. **[Close]** をクリックします。アプリケーションデプロイメントツールによって、アプリケーションの新しいバージョンが作成され、デプロイメントの準備ができました。

ステップ3: ターゲットの定義

ターゲットとは、アプリケーションのデプロイ先となるサーバーのセットです。各ターゲットには、アプリケーションに必要な**階層**が存在する必要があります。

3つの階層を持つターゲット

ターゲットをセットアップするユーザーには、2つのレベルのアクセス権が必要です。SA管理者が、関連する環境に対する書き込みアクセス権をユーザーに付与する必要があります(「環境のアクセス権」(599ペー

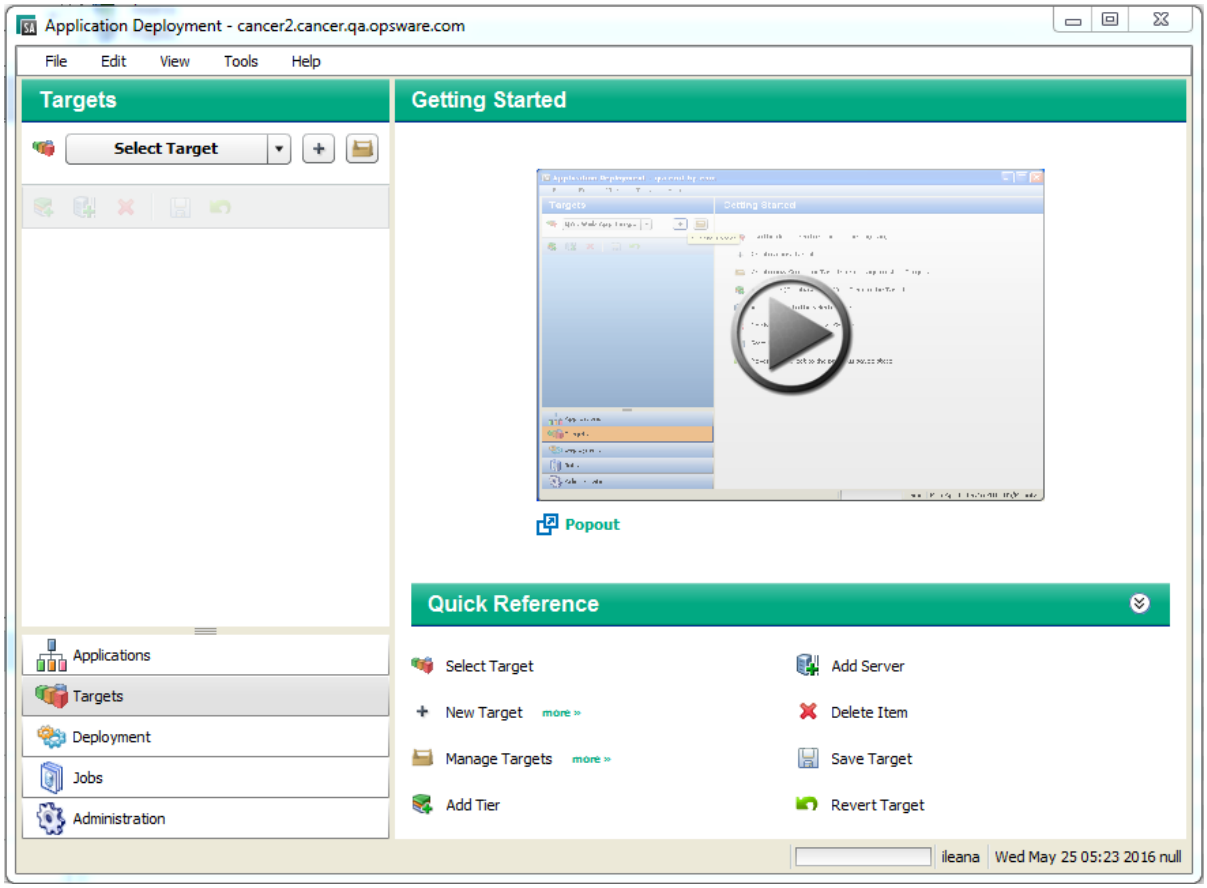
ジ)を参照)。

階層にサーバーを追加するには、対象のサーバーに対するアクセス権も必要です。これはSAクライアントで管理されます(「アクションのアクセス権」(591ページ)を参照)。

ターゲットは、アプリケーションデプロイメントツールの[Targets]画面で定義されます。

初めて[Targets]画面にアクセスする場合、画面は次の図のようになります。

[Targets]画面



ボタンをクリックすると、[Targets]画面の機能を説明した短いビデオデモが始まります。別ウィンドウでビデオを再生するには、Popoutをクリックします。

[Quick Reference]領域には、この画面で使用できるツールバーボタンの機能が表示されます。[more>>]リンクをクリックすると、関連するオンラインヘルプトピックが別のブラウザーウィンドウで開きます。

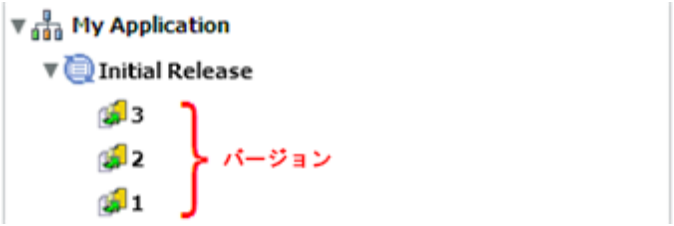
以下の手順では、新しいターゲットを作成する場合を仮定しています。また、このクイックスタートに示す手順では、ターゲットにはサーバーだけが含まれると仮定されています。

新しいターゲットを作成して構成する方法

1. [Targets] 画面に移動します (左下で [**Targets**] をクリックします)。
2. [Create a New Target] ボタンをクリックします。 [Create New Target] ダイアログが開きます。
3. ターゲット名を入力します。ターゲット名は、環境内で一意である必要があります。
4. [Location] ドロップダウンフィールドで、このターゲットが置かれる環境を選択します。

   ターゲットに階層を設定するには、2つの方法があります。手動で追加する方法 (ステップ6を参照) と、自動的に作成する方法です。

   既存のアプリケーションで使用されているのと同じ階層のセットをターゲットに自動的に作成するには、[Use Application's Tiers] を選択し、アプリケーションのバージョンを選択します。



   > **注:** [Select Version] ドロップダウンリストに表示されるためには、アプリケーションの特定のリリースの特定のバージョンがあらかじめ作成されている必要があります。

5. [**OK**] をクリックします。
6. 次の手順で、アプリケーションに必要な各階層を追加します。
   a. [**Add Tier**] ボタンをクリックします。
   b. 追加する階層のタイプを選択します。アプリケーションで使用されるのと同じ階層のセット (またはスーパーセット) を作成する必要があります。

      ソフトウェアには、標準の階層のグループが付属しています。必要な階層が利用できない場合は、アプリケーションデプロイメント管理者に追加を依頼してください。
   c. [**Add**] ボタンをクリックします。
7. ステップ6で追加した階層のそれぞれに対して、次の手順を実行して、この階層を実行するサーバー (1つまたは複数) を指定します。

a. 対象とする階層を選択します。

b. [**Add Server**] ボタンを選択して、この階層にサーバーを追加します。

使用可能なサーバーのリストは、プラットフォームによってフィルターされます。階層に対して適切なサーバーを選択してください。例：階層のコンポーネントがRed Hat Enterprise Linux 5を対象としている場合、Red Hat Enterprise Linux 5サーバーのみを選択します。

c. [Add Servers to Target] ダイアログで、追加する管理対象サーバーを見つけて選択します。複数サーバーの階層をセットアップする場合は、CTRLキーを使用して複数のサーバーを選択します。

d. [**OK**] をクリックします。

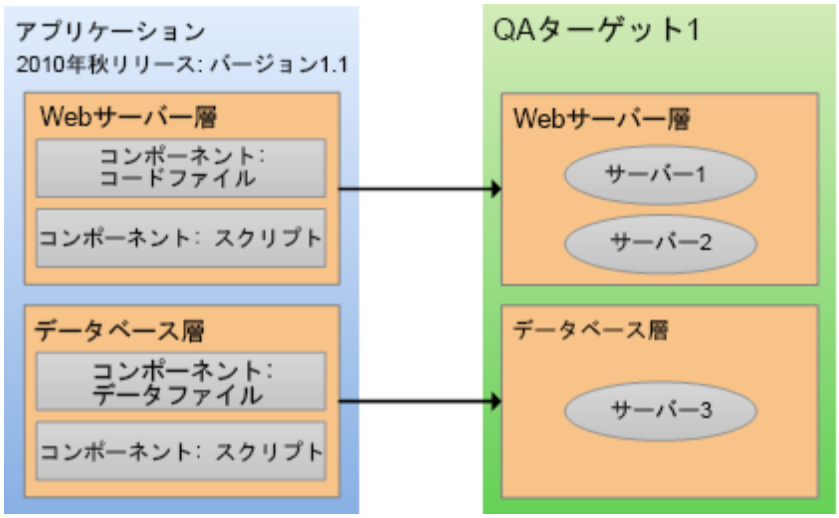
8. ステップ7を繰り返して、残りのすべての階層にサーバーを追加します。

9. [**Save Target**] ボタン をクリックして、変更を保存します。

ステップ4: ターゲットへのバージョンのデプロイ

ユーザーは、アプリケーションの**バージョン**を、**ターゲット**にデプロイします。これにより、アプリケーションファイルがターゲットサーバーにコピーされ、アプリケーションに指定されたスクリプトが実行されます。

ターゲットへのアプリケーションのデプロイ



バージョンを作成したら、そのバージョンをターゲットにデプロイできます。

バージョンをデプロイする方法

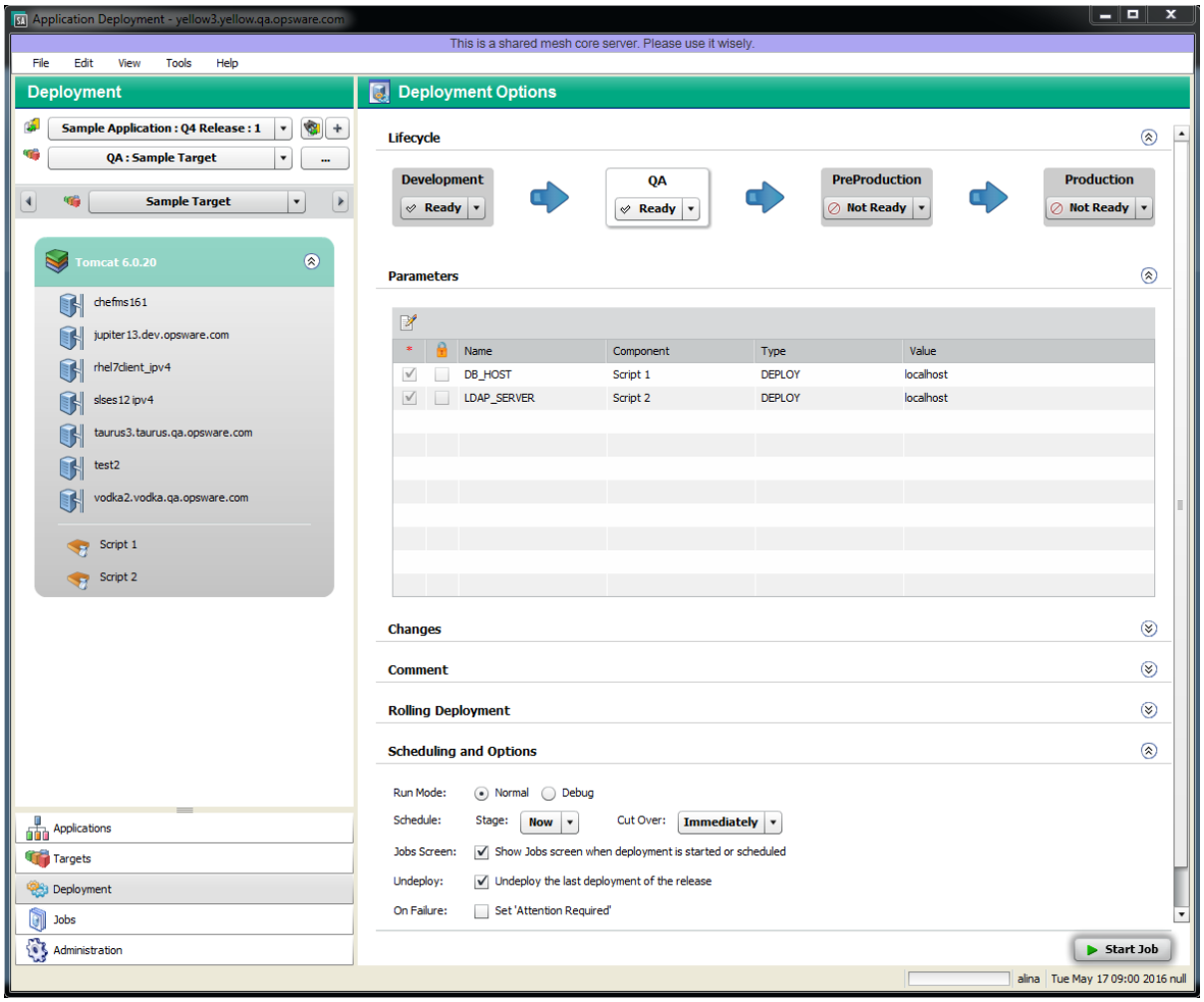
1. [Deployment] 画面に移動します (左下で [**Deployment**] をクリックします)。
2. [Select Version] ドロップダウンリストで、作成したバージョンを選択します。作成したばかりのバージョンの場合は、すでに選択されています。
3. [Select Target] ドロップダウンリストで、先に作成したターゲットを選択します。これにより、アプリケーションをデプロイするターゲットサーバーが指定されます。

   ターゲットがリストに表示されない場合は、次の条件が満たされていることを確認します。
   - ターゲットの階層がアプリケーションの階層と一致する (またはそのスーパーセットである) こと。
   - この環境に対するデプロイアクセス権があること。
   - 各階層に少なくとも1つのサーバーが存在すること。
   - このリリースのライフサイクルに、ターゲットが存在する環境が含まれていること。
   - 環境がライフサイクルで「Not Ready」とマークされていないこと。

その他のターゲットオプションについては、[Select Target] ドロップダウンリストの右側にある  ボタンをクリックしてください。これらのオプションの詳細については、「アプリケーションのデプロイ」(539ページ)を参照してください。

バージョンとターゲットを選択すると、[Deployment] 画面は次のようになります。左側のパネルにはデプロイ内容のプレビューが、右側のパネルにはデプロイメントジョブの設定が表示されます。

4. オプション: 右側のパネルのデプロイメントジョブの設定を変更します。詳細については、「アプリケーションのデプロイ」(539ページ)を参照してください。



5. [**Start Job**] をクリックします。これにより、デプロイメントジョブが起動されて、アプリケーションコンポーネントがターゲットサーバーにコピーされ、必要なコードとスクリプトが実行されます。

   デプロイメントのステータスは、[Jobs] 画面で見ることができます。

アプリケーションの概要

アプリケーションは、アプリケーションデプロイメントの基本的な構成要素の1つです。アプリケーションには、**内容**と**構造**の2つの基本的性質があります。アプリケーションの内容は、その機能を決定します。アプリケーションの構造は、その構成を反映します。アプリケーションデプロイメントのコンテキストにおいて、構造は、アプリケーションを、類似した構造の適切なアプリケーションデプロイメントターゲットに関連付ける役割を果たします。

また、構造は、**コンポーネント**がデプロイされる順序も指定します。アプリケーションデプロイメントでは、コンポーネントはアプリケーションの内容を実装します。次のタイプのコンポーネントが使用可能です。

コードコンポーネント (例: HTMLファイル、JARファイル、WARファイル、.NETアセンブリ、データベース)

- アドホックスクリプト
- 構成ファイル
- アプリケーション構成
- ソフトウェアポリシー
- パッケージ
- HPE Operations Orchestration (OO) フロー
- レジストリコンポーネント (Windowsターゲットのみ)

階層とは、アプリケーションの構造を表すコンテナーです。階層には、アプリケーションの特定の部分に必要なコンポーネントが含まれます。階層の例としては、Webサーバー、アプリケーションサーバー、データベースなどが挙げられます。**ターゲット**も階層から構成されます。アプリケーションをデプロイする場合、各アプリケーション階層のコンポーネントが、ターゲットサーバーの対応する階層にデプロイされます。

2つの階層を持つアプリケーション

1つのアプリケーションには複数の**リリース**が存在できます。また、通常、1つのリリースには複数の**バージョン**が存在します。バージョンとはある時点におけるリリースの不変の「スナップショット」です。アプリケーションを環境にデプロイする場合は、特定のバージョンをデプロイします。

リリースには、**フル**リリースと**デルタ**リリースの2つのタイプがあります。その違いは、コードコンポーネントの扱いにあります。フルリリースの場合、すべてのコードコンポーネントに属するすべてのファイルが対象となります。デルタリリースの場合、リリースの特定のバージョンが作成された後で追加または変更されたファイルだけが対象となります。削除されたファイルは、デプロイメント中にターゲットサーバー上で削除されます。

リリースに含まれるコンポーネントを指定する際には、各コンポーネントの**ロールバック**手順と**アンデプロイ**手順の両方を指定できます。ロールバック手順は、デプロイメント中に失敗が発生した場合に使用されます。この場合、アプリケーションデプロイメントはステップを逆にたどり、指定したロールバック手順に基づいてデプロイメントの各ステップを元に戻します。アンデプロイ手順は、アプリケーションをアンインストールする際に使用されます。

ロールバックとアンデプロイがどの程度自動化されるかは、コンポーネントのタイプによって異なります。たとえば、コードコンポーネントの場合、アプリケーションデプロイメント管理者によって管理される標準のスクリプトが使用されます。これに対して、スクリプトおよびOOフローコンポーネントの場合は、ロールバックおよびアンデプロイ手順を明示的に指定する必要があります。

アプリケーション**グループ**を作成することで、複数のアプリケーションを最も便利な方法で整理することができます。これはもっぱらユーザーの便宜のために提供されている整理ツールであり、アプリケーションのデプロイメントには影響しません。

新しいアプリケーションの定義プロセスには、次の3つの主要なステップがあります。

- アプリケーションとリリースの作成
- リリースに含まれる階層の指定
- 各階層のコンポーネントの指定

前提条件

アプリケーションを定義するためには、次の条件が満たされている必要があります。

このアプリケーションに必要な階層が、アプリケーションデプロイメントユーザーインタフェースで使用可能であること。

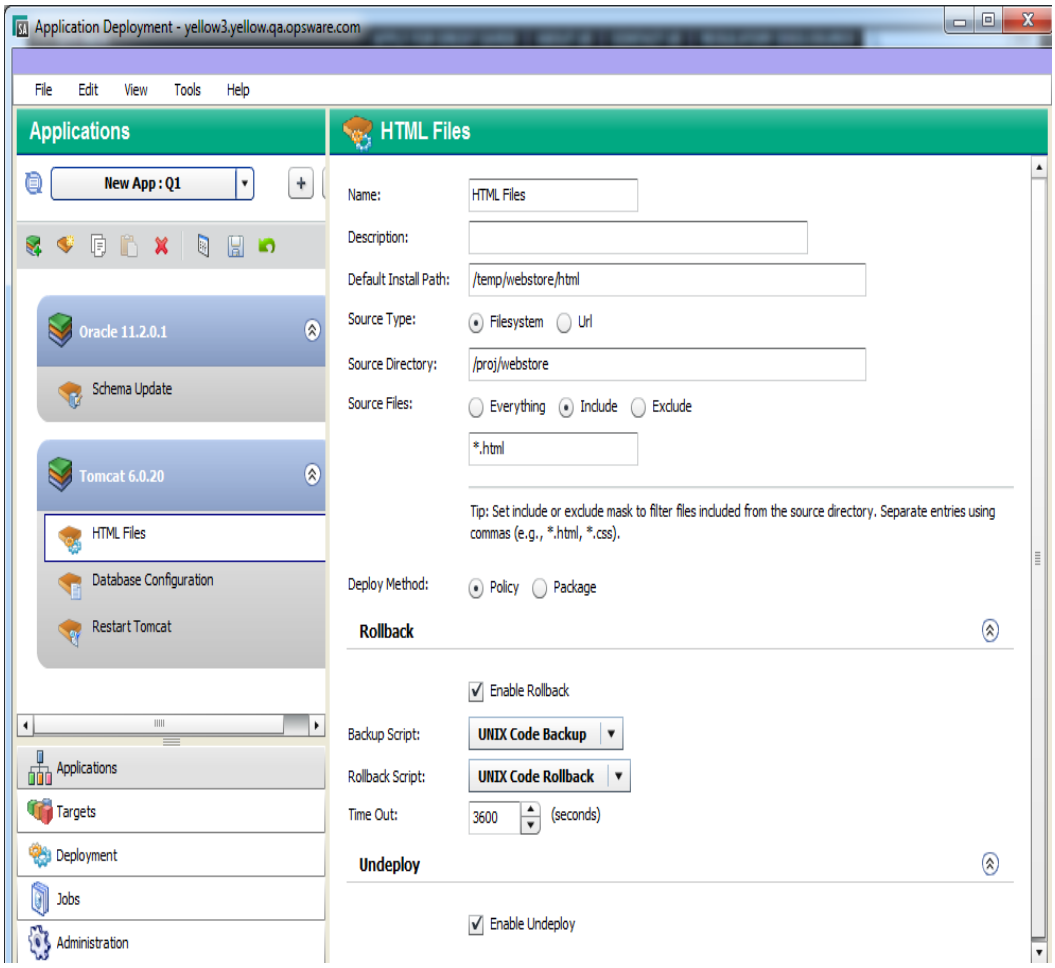
ユーザーにアプリケーション作成のアクセス権があること。

詳細については、『SA 10.50管理ガイド』を参照してください。

[Applications] 画面について

アプリケーションが定義されると、[Applications] 画面は次のようになります。

既存のアプリケーションが表示された [Applications] タブ



注意すべき点:

- アプリケーション名は一意である必要があります。
- 左上のドロップダウンリストには、ユーザーが表示アクセス権を持つ定義済みのアプリケーションとリリースがすべて表示されます。この例では、New App: Q1が選択されています。
- 左側のペインには、アプリケーションの構造が示されます。
- 右側のペインには、左側のペインで選択されているアイテムに関する情報が表示されます。1つのコンポーネントを選択することも、階層全体を選択することもできます。アイテムを選択すると、そのアイテムの背景色が変化します。この例では、HTMLファイルコンポーネントが選択されています。
- 階層の間でコンポーネントをドラッグアンドドロップすることができます(プラットフォームが一致する場合)。
- コンポーネントのデプロイメント順序は変更できます(「コンポーネントのデプロイメント順序の変更」(499ページ)を参照)。

階層について

階層とは、アプリケーションの**構造**を表すために使用されるコンテナーです。階層には、アプリケーションの機能を実装するコンポーネントが含まれます。アプリケーションデプロイメント管理者は、ユーザーが使用できる階層を定義します(『SA 10.50管理ガイド』を参照)。一般的に使用される階層の集合が標準で定義されています。管理者は、必要に応じて環境に適合するように階層を変更するか、新しい階層を追加します。

単純な例については、「クイックスタート」(440ページ)を参照してください。

アプリケーションの操作

新しいアプリケーションの作成

このトピックでは、新しいアプリケーションを作成するための基本的な手順を説明します。アプリケーションを作成するために必要なHPE Server Automationアクセス権をユーザーが持っていることが仮定されます。詳細については、「アクセス権の設定」(586ページ)を参照してください。

新しいアプリケーションを作成して構成するには、次の手順を実行します。

1. [Applications] 画面に移動します(左下の[**Applications**]をクリックします)。
2. [**Create a New Application**]をクリックします。[Create New Application] ダイアログが開きます。アプリケーションを作成するには、SAの対応するアクション権限が、ユーザーが属するグループに割り当てられている必要があります(「アクセス権の設定」(586ページ)を参照)。
   a. 新しいアプリケーションの名前を入力します(「命名規則」(438ページ)を参照)。
   b. オプション: 新しいアプリケーションを既存のアプリケーショングループに割り当てます。
   c. オプション: 新しいアプリケーションの説明を入力します。
   d. [**Release Name**] フィールドに、このアプリケーションで操作している現在のリリースの名前を入力します。

      たとえば、四半期ごとのリリースの場合には、Q3-2010、Q4-2010、Q1-2011、Q2-2011といったリリース名を使用できます。

e. このリリースのライフサイクルを選択します。ライフサイクルは、アプリケーションが開発から運用まで移行する過程で通過するフェーズを指定します。

   たとえば、一般的なライフサイクルとして、開発 > QA > 運用前 > 運用が挙げられます。これはデフォルトの標準ライフサイクルです。

f. [**OK**] をクリックしてアプリケーション情報を保存します。

3. [**Add Tier**] をクリックして、1つ以上の階層をアプリケーションに追加します。
   a. 追加する階層のタイプを選択します。

      ソフトウェアには、標準の階層のグループが付属しています。必要な階層が利用できない場合は、アプリケーションデプロイメント管理者に追加を依頼してください。『SA 10.50管理ガイド』を参照してください。

   b. [**Add**] をクリックします。
   c. 必要な追加の階層のそれぞれに対して、ステップ (a) と (b) を繰り返します。
4. 階層にコンポーネントを追加するには、次の手順を実行します。
   a. 階層を選択します。
   b. [**New Component**] をクリックして、この階層にコンポーネントを追加します。
   c. 新しいコンポーネントの名前を入力します。
   d. コンポーネントタイプを選択します。詳細については、「コンポーネントのタイプ」(470ページ)を参照してください。
   e. [**OK**] をクリックします。
   f. 必要な情報を入力して、新しいコンポーネントの場所と動作を完全に指定します。

      入力する必要がある情報は、追加するコンポーネントのタイプに応じて異なります。コンポーネントのタイプによっては、ロールバックおよびアンデプロイのシナリオのための手順が必要です。

   g. 上記のステップ (b) から (f) までを繰り返して、その他のコンポーネントをこの階層に追加します。
   h. [**Save Release**] をクリックして変更内容を保存します。
5. 上記のステップ3～4を繰り返して、その他の階層を追加し、各階層にその他のコンポーネントを追加します。
6. [**Edit component deployment order**] をクリックします。
7. 緑の矢印キーを使用して、コンポーネントを正しいデプロイメント順序に並べます (「コンポーネントのデプロイメント順序の変更」(499ページ)を参照)。

すべての階層とコンポーネントを追加したら、アプリケーションの定義は完成です。アプリケーションは、「クイックスタート」(440ページ)に示した例のようになります。

ツールバーに「警告」記号  が表示される場合、コンポーネントの定義が不完全であるか正しくないことを示します。たとえば、コードコンポーネントのソースの場所が、存在しないディレクトリを指しているような場合です。
 アイコンをクリックして、必要な情報について調べます。

アプリケーションの管理

必要なアクセス権があるユーザーは(「概要」(586ページ)を参照)、Manage Applicationsツールを使用して、いくつかの作業を実行できます。このツールでは、ターゲットにデプロイ可能なバージョンを作成できます。また、アプリケーションとリリースに関するきわめて詳細な情報を指定できます。

Manage Applicationsツールを開くには、[Applications] 画面の  ボタンをクリックします。アプリケーションを管理するアクセス権がない場合は(「アクセス権の設定」(586ページ)を参照)、このボタンは使用できません。

Manage Applicationsツールには、ここに示すアイテムを表示したテーブルが含まれます。

 アプリケーショングループ(ネスト可能)

 アプリケーション

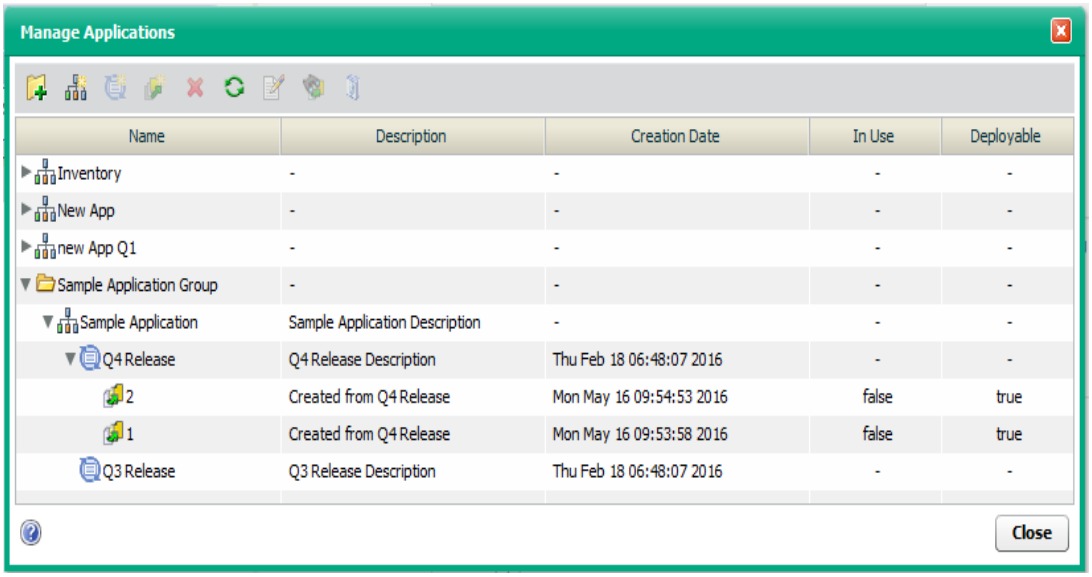
 リリース

 バージョン

表示アクセス権があるアプリケーションだけが表示されます。

[Name] 列と[Description] 列には、アイテムの作成時(または最新の編集時)に指定された情報が表示されます。[Creation Date] は、リリースとバージョンに対してのみ表示されます。[In Use] 列には、バージョンがデプロイ済み(かつアンデプロイまたはロールバックされていない)かどうかが示されます。
[Deployable] 列には、このバージョンが正常に作成されたかどうかが示されます。

Manage Applicationsツール



特定のアイテムを編集するには、そのアイテムを含む行を選択して、ツールバーの ボタンをクリックするか、単にアイテムをダブルクリックします。

バージョンは、特定の時点でのリリースの変更不可能な「スナップショット」なので、バージョンを編集することはできません。

既存のリリースのバージョンの作成

リリースをデプロイするには、そのリリースのバージョンを作成する必要があります。バージョンとは、リリースと、バージョン作成時にそのリリースに関連付けられているライフサイクルの「スナップショット」です。リリースはその後も進化することができますが、バージョンは変化しません。これにより、リリースの特定のバージョンを繰り返しデプロイすることができます。

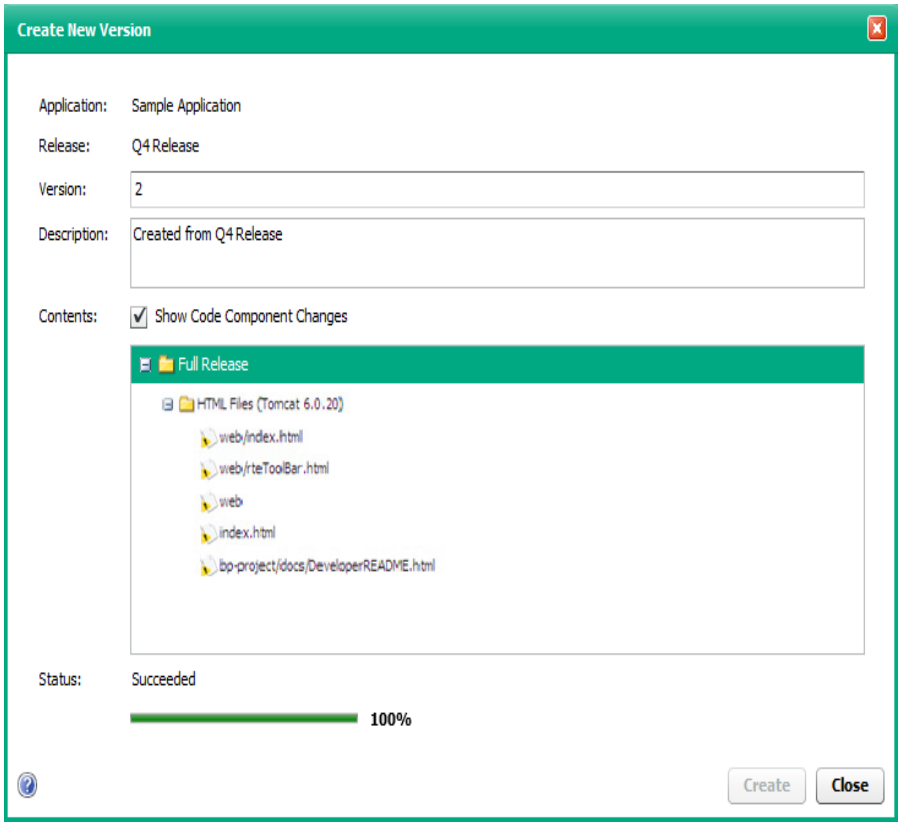
適切なアクセス権があるユーザーは (「概要」(586ページ)を参照)、リリースのバージョンを作成することができます。

バージョンは [Deployment] 画面で作成することもできます。詳細については、「アプリケーションのデプロイ」(539ページ)を参照してください。

既存のリリースのバージョンを作成するには、次の手順を実行します。

1. Manage Applicationsツールを開きます (「アプリケーションの管理」(460ページ)を参照)。
2. テーブルで、操作するリリースを含む行を選択します。
3. [Create Version] ボタンをクリックします。[Create New Version] ダイアログが開きます。

4. 次の情報を指定します。
   a. [Version] に表示される内容は、前のバージョンの名前 (多くは番号) に基づいています。これは必要に応じて変更できます。
   b. オプション: [Description] の初期値は、このバージョンが基づいているリリースを表します。これも必要に応じて変更できます。
   c. オプション: コードコンポーネントファイルのリストを表示するには、[Show Code Component Changes] チェックボックスをオンにします。フルリリースの場合、すべてのファイルがリストに表示されます。デルタリリースの場合、「ベースライン」バージョンの後で変更されたファイルだけが表示されます。例:



5. [**Create**] をクリックします。アプリケーションデプロイメントがバージョンを作成します。
6. ステータスに「Succeeded」と表示されたら、[**Close**] をクリックします。
7. ジョブが失敗した場合、[**Close**] をクリックし、[Jobs] 画面で理由を調べます。

新しいアプリケーショングループの作成

アプリケーショングループとは、アプリケーションを階層構造で整理するためのコンテナーです。適切なアクセス権があるユーザーは (「概要」(586ページ)を参照)、アプリケーショングループを作成することができます。

アプリケーショングループを作成するには、次の手順を実行します。

1. Manage Applicationsツールを開きます (「アプリケーションの管理」(460ページ)を参照)。
2. トップレベルにアプリケーショングループを作成する場合は、何も選択しません。ネストしたアプリケーショングループを作成するには、新しいグループを中に作成するグループを選択します。
3. [**Create Application Group**] をクリックします。
4. 新しいグループの名前を指定します (「命名規則」(438ページ)を参照)。この名前は、アプリケーション階層の同じレベルで一意である必要があります。
5. [**OK**] をクリックします。新しいグループがテーブルに表示されます。グループとアプリケーションは、階層のレベルごとにアルファベット順で表示されます。

Manage Applicationsツールで、グループの間でアプリケーションをドラッグアンドドロップすることができます。

新しいアプリケーションの作成

適切なアクセス権があるユーザーは (「概要」(586ページ)を参照)、Manage Applicationsツールを使用して新しいアプリケーションを作成することができます。これは、[Applications] メイン画面でアプリケーションを作成する代わりに使用できる方法です。これはたとえば、階層やコンポーネントを追加する前に、アプリケーションへのアクセス権を設定する場合に便利です。

Manage Applicationsツールで新しいアプリケーションを作成するには、次の手順を実行します。

1. Manage Applicationsツールを開きます (「アプリケーションの管理」(460ページ)を参照)。
2. トップレベルにアプリケーションを作成する場合は、何も選択しません。既存のアプリケーショングループ内にアプリケーションを作成するには、新しいアプリケーションを含むグループを選択します。
3. [**Create Application**] をクリックします。
4. 次の情報を指定します。
   a. 新しいアプリケーションの一意の名前を指定します (「命名規則」(438ページ)を参照)。
   b. オプション: 新しいアプリケーションの説明を入力します。

c. [Release Name] フィールドに、このアプリケーションで操作している現在のリリースの名前を入力します。デフォルトでは、これは「Initial Release」です。

d. このリリースのライフサイクルを選択します。ライフサイクルは、アプリケーションが開発から運用まで移行する過程で通過するフェーズを指定します。

   たとえば、一般的なライフサイクルとして、開発 → QA → 運用前 → 運用が挙げられます。これはデフォルトの標準ライフサイクルです。

5. [**OK**] をクリックします。新しいアプリケーションがテーブルに表示されます。アプリケーションとグループは、階層のレベルごとにアルファベット順で表示されます。

次のステップ:

このアプリケーションに対するアクセス権を指定するには、Manage Applicationsツールを開いたままにして、「アクセス権の設定」(586ページ)を参照してください。

アプリケーションに対して階層とコンポーネントを指定するには、Manage Applicationsツールを閉じ、[Applications] メイン画面で、作成したリリースを選択します。

既存のアプリケーションの新しいリリースの作成

適切なアクセス権があるユーザーは (「概要」(586ページ)を参照)、Manage Applicationsツールを使用して、既存のアプリケーションの新しいリリースを作成することができます。新しいリリースは、既存のリリースから複製されます。リリースは、フルリリースまたはデルタリリースのどちらかです。フルリリースには、すべてのコードコンポーネントに含まれるすべてのファイルが含まれます。デルタリリースには、指定した「ベースライン」バージョンの作成後に追加または変更されたコードコンポーネントファイルだけが含まれます。

詳細については、「アプリケーションの管理」(460ページ)を参照してください。

既存のアプリケーションの新しいリリースを作成するには、次の手順を実行します。

1. Manage Applicationsツールを開きます (「アプリケーションの管理」(460ページ)を参照)。
2. テーブルで、操作するアプリケーションを含む行を選択します。
3. [Create Release] ボタンをクリックします。  . [Create New Release] ダイアログが開きます。
4. 次の情報を指定します。
   a. [Clone Release] ドロップダウンリストで、複製するリリースを選択します。新しいアプリケーションには、選択したリリースと同じ階層およびコンポーネントが含まれます。

b. このリリースの名前を入力します(「命名規則」(438ページ)を参照)。名前は、このアプリケーション内で一意である必要があります。

c. オプション: このリリースの説明を[Description]に入力します。この情報はレポートに表示され、Manage Applicationsツールにも表示されます。

d. オプション: このリリースの変更要求番号を[Change Request]に指定します。この情報もレポートに表示されます。

e. オプション: [ROI]ボックスに、各ターゲットマシンで実現されると予想される投資収益率の見積もりを入力します。これは、このリリースをデプロイすることでどのくらいの価値が得られるかに関するユーザーの評価を表します。このフィールドのスケールと意味は、ユーザーが定義します。これは、HPE Server AutomationROIレポートで使用されます。任意の正の整数が使用できます。

f. オプション: [Planned Release Date]に予定リリース日を指定します。この情報もレポートに表示されます。

g. これがデルタリリースの場合、[Delta]チェックボックスをオンにし、「ベースライン」バージョンをドロップダウンリストから選択します。各コードコンポーネントに指定されたソースディレクトリにあるファイルのうち、ベースラインバージョンの作成後に追加または変更されたものだけがデプロイされます。

これがフルリリースの場合、[Delta]チェックボックスがオフになっていることを確認します。

h. [Lifecycle]ドロップダウンリストで適切なライフサイクルを選択します。

5. [**OK**]をクリックします。

既存のリリースのプロパティの変更

適切なアクセス権があるユーザーは(「概要」(586ページ)を参照)、任意のリリースのプロパティを変更することができます。

既存のリリースのプロパティを変更するには、次の手順を実行します。

1. Manage Applicationsツールを開きます(「アプリケーションの管理」(460ページ)を参照)。
2. テーブルで、変更するリリースを含む行を選択します。
3. [Edit Properties]ボタン:  をクリックします。
4. リリースのプロパティを変更します。プロパティの説明については、「アプリケーションの概要」(454ページ)を参照してください。
5. [**OK**]をクリックします。

アプリケーショングループ、アプリケーション、リリース、またはバージョンの削除

適切なアクセス権があるユーザーは (「概要」(586ページ)を参照)、Manage Applicationsツールを使用して、アプリケーション、グループ、リリース、またはバージョンを削除することができます。

既存のアプリケーション、グループ、リリース、またはバージョンを削除するには、次の手順を実行します。

1. Manage Applicationsツールを開きます (「アプリケーションの管理」(460ページ)を参照)。
2. テーブルで、削除するアイテムを含む行を選択します。次の点に注意してください。
   - 複数のアイテムを選択するには、CtrlまたはShiftキーを使用します。ただし、複数のアイテムを削除する場合は、同じタイプでなければなりません。たとえば、複数のアプリケーションや複数のリリースなどです。
   - アプリケーションを削除すると、そのアプリケーションのリリースとバージョンもすべて削除され、スケジュールされてまだ実行されていないジョブも削除されます。
   - 使用中のバージョンは削除できません。

     先に [Jobs] 画面からそのバージョンを「アンデプロイ」する必要があります (「アプリケーションデプロイメントジョブの管理」(562ページ)を参照)。その後に削除することができます。

     [**Show Deployment Jobs**] をクリックすると、このバージョンを削除するためにアンデプロイまたはロールバックする必要があるジョブの一覧が表示されます。

     ただし、アプリケーションデプロイメント管理者以外のユーザーは、自分が開始したデプロイメントしかアンデプロイまたはロールバックすることができません。
   - アプリケーショングループの一部は削除できません。
3. [**Delete**] をクリックします。

この操作を元に戻すことはできません。アイテムを削除してよいかどうかの確認が表示されます。[**Yes**] をクリックして削除します。

アプリケーションまたはアプリケーショングループの名前変更

適切なアクセス権があるユーザーは(「概要」(586ページ)を参照)、Manage Applicationsツールを使用して、アプリケーションまたはアプリケーショングループの名前を変更することができます。

アプリケーションまたはグループの名前を変更するには、次の手順を実行します。

1. Manage Applicationsツールを開きます(「アプリケーションの管理」(460ページ)を参照)。
2. テーブルで、名前を変更するアプリケーションまたはグループを含む行を選択します。
3. [**Edit Properties**] をクリックします。
4. アプリケーションまたはグループの新しい一意の名前を入力します(「命名規則」(438ページ)を参照)。
5. [**OK**] をクリックします。

アプリケーションに対するアクセス権の付与または取り消し

アプリケーションを作成する際に、そのアプリケーションの表示、編集、デプロイを実行できるSAユーザーおよびユーザーグループを指定できます。これらのアクセス権を明示的に指定しなかった場合は、作成したユーザーとアプリケーションデプロイメント管理者だけが、アプリケーションにアクセスできます。詳細については、「概要」(586ページ)を参照してください。

アプリケーションに対するアクセス権を設定するには、次の手順を実行します。

1. Manage Applicationsツールを開きます(「アプリケーションの管理」(460ページ)を参照)。
2. テーブルで、操作するアプリケーションを含む行を選択します。
3. [**Edit Properties**] をクリックします。
4. [Choose User] メニューで、ユーザーまたはユーザーグループを選択します。
5. [**Add User**] をクリックします。

6. 付与するアクセス権を選択します。
   a. [View] アクセス権を持つユーザーは、[Applications] 画面の [Select Release] ドロップダウンリストでこのアプリケーションを表示できます。
   b. [Edit] アクセス権を持つユーザーは、アプリケーションの内容 (コンポーネント)、構造 (階層)、またはプロパティ (名前、説明、リリース名など) を変更できます。
   c. [Deploy] アクセス権を持つユーザーは、アプリケーションを1つ以上のターゲットにデプロイできます。

   これらのアクセス権の任意の組み合わせを選択または拒否することができます。[Edit] または [Deploy] を選択すると、[View] が自動的に選択されます。
7. アクセス権のリストからユーザー (またはグループ) を削除するには、そのユーザー (またはグループ) を含む行を選択して、[**Delete**] をクリックします。
8. [**OK**] をクリックして [Edit Applications] ダイアログを閉じ、変更内容を保存します。[**Cancel**] ボタンをクリックすると、変更内容が破棄されます。

特定のバージョンのプロパティの表示

適切なアクセス権があるユーザーは (「概要」(586ページ)を参照)、Manage Applicationsツールからバージョンビューアーを開くことができます (「バージョンのプロパティの表示」(555ページ)を参照)。

Manage Applicationsツールからバージョンビューアーを開くには、次の手順を実行します。

1. Manage Applicationsツールを開きます (「アプリケーションの管理」(460ページ)を参照)。
2. テーブルで、表示するバージョンを選択します。
3. ツールバーの [**View version**] をクリックします。バージョンビューアーが開きます。

特定のバージョンを使用しているデプロイメントジョブの表示

適切なアクセス権があるユーザーは (「概要」(586ページ)を参照)、リリースの最も新しいバージョンを使用しているデプロイメントジョブの一覧を表示できます。

特定のバージョンを使用しているデプロイメントジョブを表示するには、次の手順を実行します。

1. Manage Applicationsツールを開きます(「アプリケーションの管理」(460ページ)を参照)。
2. テーブルで、表示するバージョンを選択します。
3. ツールバーの [**Show Deployment Jobs**] をクリックします。[Jobs] 画面が開き、このバージョンを使用するすべてのデプロイメントジョブのリストが表示されます。

コンポーネント

概要

自分が編集アクセス権を持っているアプリケーションで、コンポーネントを作成して変更することができます(「アクセス権の設定」(586ページ)を参照)。

単純なコンポーネントを持つアプリケーションの例については、「クイックスタート」(440ページ)を参照してください。

コンポーネントについて

コンポーネントは、アプリケーションの**内容**を表します。コンポーネントは、アプリケーションのさまざまな機能を実装します。コンポーネントは特定の階層と関連付けられます。コンポーネントには8つのタイプがあり、それらをアプリケーションの階層内に入れることができます。

コンポーネントタイプ	アイコン	目的
コード		ターゲットサーバーに**ファイルをコピー**します。たとえば、Webストアを実現するHTMLファイルのセットなどです。「コードコンポーネント」(471ページ)を参照してください。
スクリプト		アプリケーションの**デプロイメント中に**ターゲットサーバー上でスクリプトを実行します。たとえば、デプロイメント前にすべてのアプリケーションをシャットダウンするスクリプトと、デプロイメント後にもう一度アプリケーションを起動するスクリプトを用意することができます。「スクリプトコンポーネント」(475ページ)を参照してください。
構成ファイル		ターゲットサーバー上に新しい構成ファイルを作成します。これらのファイルには、特定のアプリケーションに関

コンポーネントタイプ	アイコン	目的
		する**きわめて具体的な情報**が含まれるのが普通です。たとえば、データベースに関する情報を指定する構成ファイルの作成が必要な場合があります。「構成ファイルコンポーネント」(477ページ)を参照してください。
アプリケーション構成		既存の**アプリケーション構成**をSAライブラリからターゲットサーバーにデプロイします。「アプリケーション構成コンポーネント」(481ページ)を参照してください。
ソフトウェアポリシー		**ソフトウェアポリシー**をSAライブラリからターゲットサーバーにアタッチします。別の方法として、ポリシーをアタッチする代わりにインストールするように指定することもできます。「ソフトウェアポリシーコンポーネント」(485ページ)を参照してください。
パッケージ		**ソフトウェアパッケージ**をSAライブラリからターゲットサーバーにインストールします。別の方法として、パッケージをポリシーとしてアタッチするように指定することもできます。「パッケージコンポーネント」(487ページ)を参照してください。
OOフロー		**HPE Operations Orchestration (OO)** ワークフローを開始します。たとえば、特定のアプリケーションがデプロイされたときに、製品の所有者に電子メールが送信されるようなワークフローを使用できます。「OOフローコンポーネント」(491ページ)を参照してください。
DMAフロー		**HPE Database and Middleware Automation (DMA)** ワークフローを、ターゲットサーバー、インスタンス、またはデータベース上で開始します。たとえば、DMAワークフローを使用して、データベースの重要なパッチをインストールすることができます。
Windowsレジストリ		**Windowsレジストリ**のキーまたは値を追加または削除します。このタイプのコンポーネントは、Windows階層でのみ使用できます。「Windowsレジストリコンポーネント」(493ページ)を参照してください。

コンポーネントはサーバーに対して実行されます。デフォルトでは、コンポーネントは作成された順序でデプロイされます。この順序は、アプリケーションの編集アクセス権があれば変更することができます (「コンポーネントのデプロイメント順序の変更」(499ページ)を参照)。

コンポーネントのタイプ

以下では、アプリケーションデプロイメントコンポーネントのタイプについて説明します。

OOフローコンポーネントが使用できるのは、SAコアサーバーでHPE OOサーバーとの統合が設定されている場合に限ります。

Windowsレジストリコンポーネントが使用できるのは、Windows階層だけです。

コードコンポーネント

コードコンポーネントは、管理対象サーバーにファイルをデプロイするために使用されます。コードコンポーネントは単に、指定されたリポジトリからアプリケーションデプロイメントが作成するファイルのパッケージです。リポジトリとしては、次のいずれかが使用できます。

- SAコアサーバーのファイルシステム。
- URLでアクセスされる1つのファイル
- ソースコード管理システム

コードコンポーネントは動的です。バージョンが作成されるたびに、ディレクトリ(またはURL)のその時点の内容が反映されます。これに対して、ソフトウェアポリシーコンポーネントやパッケージコンポーネントでは、SAライブラリからの比較的静的なアイテムが毎回使用されます。

コードコンポーネントに含めることができるファイルは最大100,000個です。

コードコンポーネントを作成する際には、次の情報を指定します。

コードコンポーネントのプロパティ

プロパティ	必須	目的
Name	はい	コンポーネントの名前。この名前は階層に表示されます。これはアプリケーション内で一意である必要があります。

コードコンポーネントのプロパティ (続き)

プロパティ	必須	目的
Description	いいえ	レポートに表示される説明。
Base Path	いいえ	[Base Path] は、[Default Install Path] と結合されて、フルインストールパス (管理対象サーバー上でファイルが置かれる場所) を決定します。[Base Path] と [Full Install Path] は、[Sandboxing] が有効になっている場合のみ表示されます (「アプリケーション設定の管理」を参照)。 表の下の注を参照してください。
Default Install Path	はい	ターゲット上でファイルが配置されるディレクトリ ([Sandboxing] が有効な場合は [Base Path] 内)。
Full Install Path	読み取り専用	[Base Path] + [Install Path] ([Sandboxing] が有効な場合のみ表示)。
Source Type	はい	ソースファイルが存在する場所。 Filesystem – SAコアサーバーのファイルシステム内でファイルを探します。これには、SambaおよびNFSマウントポイントが含まれます。 CVSまたはSubversion – 指定したソースコード管理システム内でファイルを探します (「ソースコード管理システムの使用」(475ページ)を参照)。 URL – URLに指定されたプロトコルに従って、1つのファイルを探します (「URLの使用」(475ページ)を参照)。 これらの場所は、アプリケーションデプロイメント管理者によって管理されます。詳細については、「コードコンポーネントソースタイプの管理」(611ページ)を参照してください。
Source Directory	はい	リポジトリ内でソースファイルが存在するディレクトリ。このディレクトリは、SAコアサーバーにマウントされている必要があります。 ソースタイプにURLを指定した場合は、[Source Directory] はありません。代わりに [URL] プロパティがあります。
Source Files	はい	Everything – ソースディレクトリ内のすべてのファイルを取得します。 Include – ソースディレクトリ内で、指定したパターンに一致する名前を持つすべてのファイルを取得します。 Exclude – ソースディレクトリ内で、指定したパターンに一致する名前を持つファイル以外のすべてのファイルを取得します。

コードコンポーネントのプロパティ (続き)

プロパティ	必須	目的
		「フィルターの使用」(474ページ)を参照してください。
Deploy Method	はい	新規バージョンを作成する場合、アプリケーションデプロイメントは、指定したファイルと、必要なスクリプトを含むパッケージを作成します。 Policy – アプリケーションデプロイメントは、新規バージョンを作成する際にポリシーも作成します。バージョンがデプロイされるときに、このポリシーが、ターゲットに含まれる各管理対象サーバーにアタッチされ、修復されます。 Package – バージョンがデプロイされるときに、アプリケーションデプロイメントは、ターゲットに含まれる各管理対象サーバーに対して、パッケージの「アドホック」インストールを実行します。
Backup	いいえ	使用可能なバックアップおよびロールバックスクリプトは、階層のプラットフォームによって異なります。これらのスクリプトは、アプリケーションデプロイメント管理者によって管理されます。詳細については、「スクリプトの管理」(609ページ)を参照してください。
Rollback	いいえ	バックアップおよびロールバックスクリプトの完了をSAが待つ最大時間 (秒) を指定します。スクリプトがこの時間内に完了しなかった場合、スクリプトは中止されます。デフォルトは3600秒 (1時間) です。
Time Out	はい	
Undeploy	いいえ	有効または無効。

> **注:** サンドボックスを有効にする場合、コードコンポーネントの作成時に [Base Path] を指定する必要があります。[Base Path] が指定されていないコードコンポーネントを含むリリースのバージョンを作成することはできません (サンドボックスが有効にされている場合)。

既存のリリースでコードコンポーネントによって使用されているベースディレクトリをアプリケーションデプロイメント管理者が削除した場合、そのリリースの新規バージョンを作成することはできなくなります。ただし、リリースの既存のバージョンは引き続きデプロイできます。

シンボリックリンクの使用

アプリケーションデプロイメントは、コードコンポーネント内のシンボリックリンクをサポートします。シンボリックリンクを使用する場合は、次の点に注意してください。

- シンボリックリンクが使用できるのはUNIXターゲットだけです。Windowsオペレーティングシステムでは使用できません。
- ソースディレクトリがシンボリックリンクの場合、SAはリンクをたどってコードコンポーネントのファイルを探します。
- ソースディレクトリが指しているディレクトリ構造の中にシンボリックリンクが見つかった場合、そのシンボリックリンクはたどられず、アーカイブされます。ターゲットサーバー上にファイルがインストールされる際に、元の値でシンボリックリンクが作成されます。
- シンボリックリンクは、含めるか除外するかの判定に関しては、ファイルと同様に扱われます。

フィルターの使用

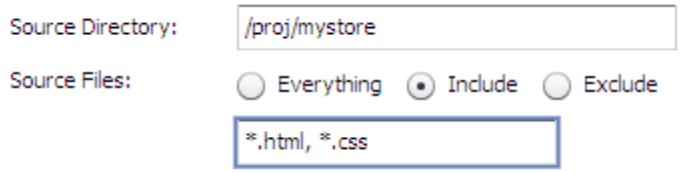
[Include]/[Exclude] オプションでフィルターを使用することで、コードコンポーネントに含めるファイルを指定することができます。

- フルリリースの場合、フィルター条件を満たすすべてのファイルとディレクトリが含められます。
- デルタリリースの場合、フィルター条件を満たし、タイムスタンプが関連するバージョンよりも後であるファイルだけが含められます。ファイルがファイルシステムから削除されている場合は、コンポーネントパッケージには含められません。

フィルターには、ワイルドカード文字として*が使用できます。アプリケーションデプロイメントは、ソースディレクトリのサブディレクトリを再帰的に検索して、指定したパターンに一致するファイルを探します。例:

- `*/src/*`は、`src`という名前のディレクトリがパスに含まれるすべてのファイルに一致します。
- `Fred*`は、名前の先頭が`Fred`であるすべてのファイルと、パス内の`Fred`という名前のディレクトリに一致します。

フィルターパターンは、次のようにカンマで区切ります。



このパターンは、たとえば、次のようにファイル名の拡張子が`*.html`または`*.css`であるすべてのファイルに一致します。

```
/proj/mystore/index.html

/proj/mystore/products/dogs/dachshunds.html

/proj/mystore/main.css
```

ソースコード管理システムの使用

ソースコード管理システムの例としては、CVSやSubversionが挙げられます。アプリケーションデプロイメントは、一般的なソースコード管理システムのほとんどと統合されます。

この場合、ソースディレクトリは、SAコアサーバー上に存在するソースリポジトリのクライアントコピーの場所を表します。[Include]/[Exclude] フィルターを使用することで、コンポーネントに含めるファイルを指定することができます。

ソースタイプとしてソースコード管理システムを使用することには、次のような利点があります。

- リリースのバージョンを作成するたびに、ソースコード管理システムを使用してクライアントコピー内のファイルが更新されます。このため、最新の更新がコードコンポーネントに含まれることを保証できます。
- デルタリリースの場合、ソースコード管理システムからファイルが削除されていると、新しいバージョンがデプロイされるときに、そのファイルはターゲットサーバーからも削除されます。

同じソースコード管理リポジトリにアクセスするコードコンポーネントが複数存在する場合、各コンポーネントに異なるソースディレクトリを指定してください。これは、アプリケーションのデプロイ時に同時実行エラーが発生するのを防ぐためです。

アプリケーションでこのソースタイプを使用するには、アプリケーションデプロイメント管理者がソースコード管理システムを統合している必要があります。詳細については、「コードコンポーネントソースタイプの管理」(611ページ)を参照してください。

URLの使用

URLソースタイプのコードコンポーネントを作成する際には、そのURLに指定されたプロトコル(通常はHTTP) でアクセス可能な1つのファイルを指すURLを指定します。たとえば、アプリケーションの最新の夜間ビルドを、自動ビルドシステムによって置かれた場所から取得することができます。

HTTPSプロトコルを使用する場合、必要な証明書が正しい場所にインストールされている必要があります。

スクリプトコンポーネント

スクリプトコンポーネントは、アプリケーションの**デプロイメント中**に、ターゲット上で実行されます。たとえば、デプロイメント前にすべてのアプリケーションをシャットダウンするスクリプトと、デプロイメント後にもう一度アプリケーションを起動するスクリプトを用意することができます。

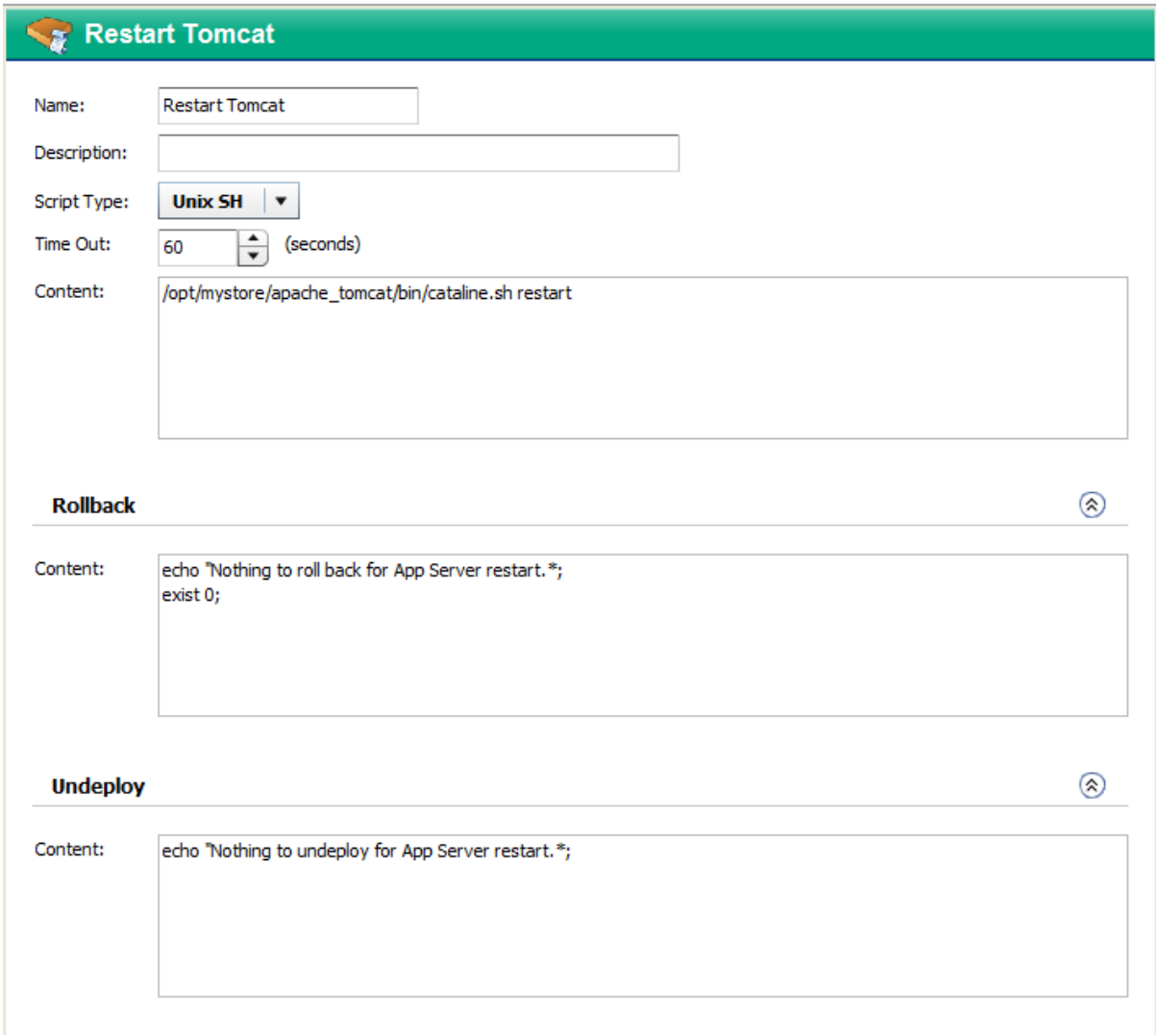
スクリプトコンポーネントを作成する際には、次の情報を指定します。

スクリプトコンポーネントのプロパティ

プロパティ	必須	目的
Name	はい	コンポーネントの名前。この名前は階層に表示されます。これはアプリケーション内で一意である必要があります。
Description	いいえ	他のアプリケーションデプロイメントユーザーのためのコンポーネントのドキュメントを記述するために使用できるフィールド。
Script Type	はい	スクリプトのタイプ。選択可能な値は、階層で要求されるプラットフォームによって異なります。
Time Out	はい	SAがスクリプトの完了を待つ時間 (秒)。
Content	はい	デプロイメント中にターゲット上で実行される実際の命令。
Rollback Content	いいえ	ロールバックまたはアンデプロイを有効にする場合は、各シナリオに対応する明示的な命令を指定する必要があります。スクリプト自体の内容と同じスクリプト言語を使用します。
Undeploy Content	いいえ	

次に示す単純な例は、ターゲット上のTomcatアプリケーションサーバーを再起動します。

UNIXターゲットに対するスクリプトコンポーネントの例



スクリプトコンポーネントでは、パラメーターを使用できます。詳細については、「パラメーターと特殊変数」(506ページ)を参照してください。

構成ファイルコンポーネント

構成ファイルコンポーネントは、ターゲット上に新しい構成ファイルを作成します。このファイルには、アプリケーションに必要な**固有の情報**が含まれます。たとえば、特定のアプリケーションが使用するデータベースを指定する構成ファイルの作成が必要な場合があります。

構成ファイルコンポーネントは、単純なキーと値から構成されるファイルの場合に最も適しています。

複雑なパラメーター化やアプリケーション固有の構成に対しては、アプリケーション構成コンポーネントを代わりに使用します(「アプリケーション構成コンポーネント」(481ページ)を参照)。

構成ファイルコンポーネントを作成する際には、次の情報を指定する必要があります。

構成ファイルコンポーネントのプロパティ

プロパティ	必須	目的
Name	はい	コンポーネントの名前。この名前は階層に表示されます。これはアプリケーション内で一意である必要があります。
Description	いいえ	レポートに表示される説明。
Base Path	いいえ	[Base Path] は、[Destination File] と結合されて、フルインストールパス(管理対象サーバー上で構成ファイルが置かれる場所)を決定します。 [Base Path] と[Full Install Path] は、[Sandboxing] が有効になっている場合のみ表示されます(「アプリケーション設定の管理」を参照)。 表の下の注を参照してください。
Destination File	はい	ターゲットサーバー上に作成される構成ファイルの名前。 [Sandboxing] が有効になっている場合、[Base Path] と [Destination File] の組み合わせによって [Full Install Path] が決定され、そこに構成ファイルが作成されます。 [Sandboxing] が有効になっていない場合、ターゲットサーバー上で構成ファイルが作成される場所のフル絶対パスを指定する必要があります。相対パスは指定しないでください。例: `/tmp/webstore/config/db.cfg`
Full Install Path	読み取り専用	[Base Path] + [Destination File] ([Sandboxing] が有効な場合のみ表示)。
Create Path	いいえ	[Create Path] を選択した場合、構成ファイルを正しい場所に配置するために必要なディレクトリ階層がターゲットサーバー上にすでに存在しないときには、階層が作成されます。
Content	はい	構成ファイルの内容。
Rollback	該当しない	使用可能なバックアップ、ロールバック、アンデプロイスクリプトは、階層のプラットフォームによって異なります。これらのスクリプトは、アプリケーションデプロイメント管理者によって管理されます。詳細については、「スクリプトの管理」(609ページ)を参照してください。
Undeploy	該当しない	

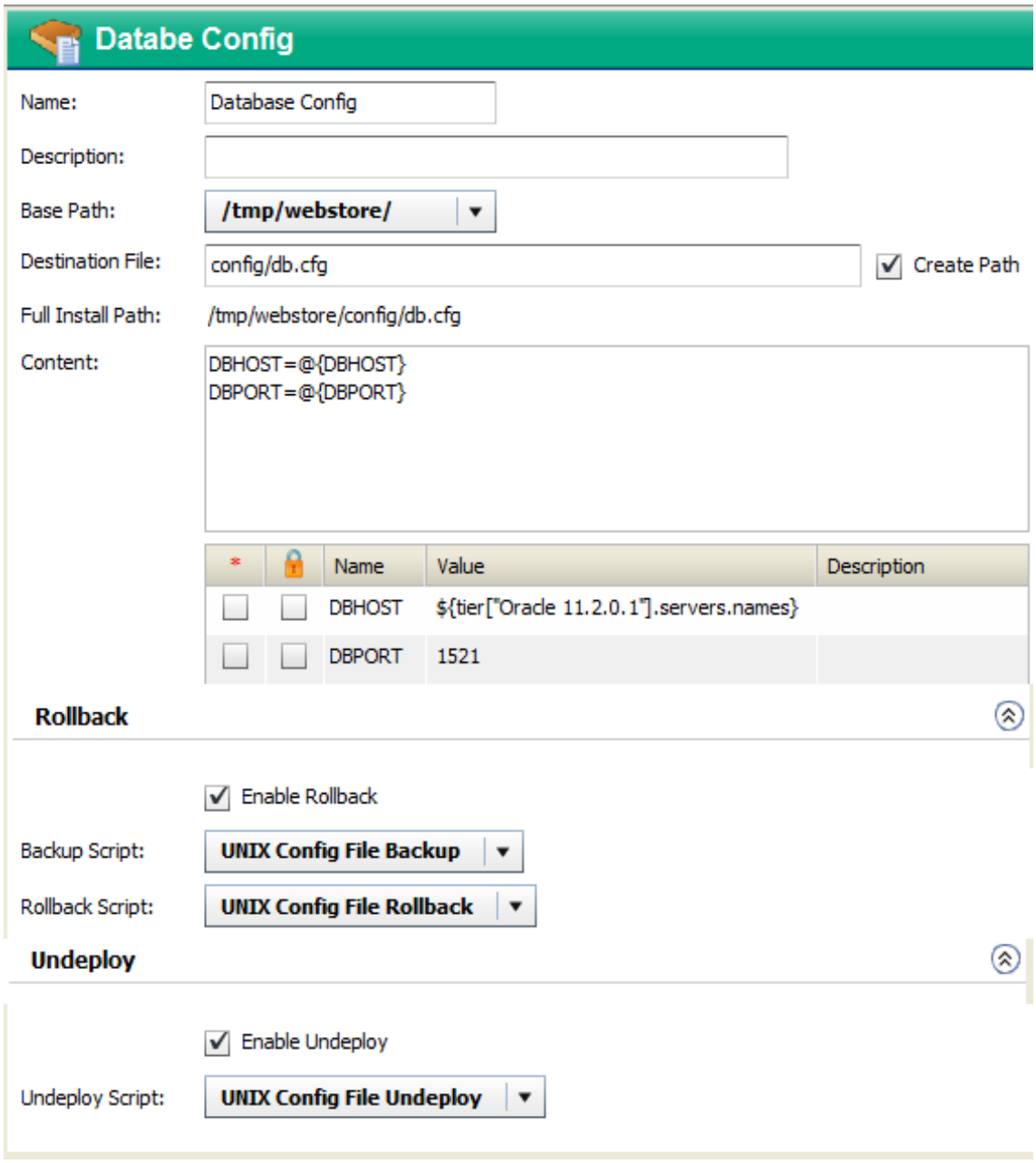
注: サンドボックスを有効にする場合、構成ファイルコンポーネントの作成時に [Base Path] を指定

> する必要があります。[Base Path] が指定されていない構成ファイルコンポーネントを含むリリースのバージョンを作成することはできません (サンドボックスが有効にされている場合)。

既存のリリースで構成ファイルコンポーネントによって使用されているベースディレクトリをアプリケーションデプロイメント管理者が削除した場合、そのリリースの新規バージョンを作成することはできなくなります。ただし、リリースの既存のバージョンは引き続きデプロイできます。

構成ファイルコンポーネントでは、パラメーターを使用して、その値をデプロイメント時に割り当てることができます。詳細については、「コンポーネントのタイプ」(470ページ)を参照してください。

UNIXターゲットに対する構成ファイルコンポーネントの例



この単純な例は、db.cfgという名前のファイルを作成します。このファイルには、データベースサーバーのホスト名とポートが記述されます。DBHOSTおよびDBPORTパラメーターの値は、コンポーネントがターゲットにデプロイされるときに割り当てられます。DBPORTの値は1521です。DBHOSTの値は、ターゲット内のOracle 11.2.0.1階層に含まれるサーバーの名前のカンマ区切りリストです。

アプリケーション構成コンポーネント

SAでは、構成ファイルを1か所でまとめて管理し、変更や更新をデータセンター内の複数のサーバーに容易に反映することができます。1つの構成ファイル(UNIXシステム上の/etc/hostsファイルなど)を管理することも、1つのアプリケーションに関連する複数の複雑な構成ファイル(Web LogicやWebSphereなどの大規模なビジネスアプリケーションに関連する構成ファイルなど)を管理することもできます。

SAで構成ファイルの管理に使用されるメカニズムは、**アプリケーション構成**と呼ばれています。これは2種類の要素から構成されます。1つ以上のテンプレートと、値のセットです。

アプリケーションデプロイメントでは、SAライブラリにある既存のアプリケーション構成を使用するコンポーネントを作成できます。値セット内のアイテムは、デプロイメント時に変更できます。たとえば、アプリケーションデプロイメントのターゲットに固有の情報 (データベースをホストしているサーバーの名前など) を記述したXMLファイルが必要な場合があります。

アプリケーション構成コンポーネントを作成する際には、次の情報を指定します。

アプリケーション構成コンポーネントのプロパティ

プロパティ	必須	目的
Name	はい	コンポーネントの名前。この名前は階層に表示されます。これはアプリケーション内で一意である必要があります。
Description	いいえ	レポートに表示される説明。
Application Configuration Name	はい	これら3つのプロパティは、SAライブラリ内のアプリケーション構成で設定されます。 アプリケーションデプロイメントユーザーインタフェースではこれらのプロパティを変更することはできません。ただし、アプリケーション構成が存在するフォルダーに対する適切なアクセス権があるユーザーは、SAの[アプリケーション構成]ウィンドウを開いて、そこでこれらのプロパティを変更できます。このウィンドウを開くには、このアプリケーション構成の[Name]リンクをクリックします。
Application Configuration Description	いいえ	デプロイメント時に変更が許可されているアプリケーション構成の値セットのアイテムが、テーブルに表示されます。 特定のアプリケーション構成に対して複数のテンプレートが
Instance Name	はい	
Parameters	いいえ	

アプリケーション構成コンポーネントのプロパティ (続き)

プロパティ	必須	目的
		存在する場合は、すべてのテンプレートにあるすべての編集可能なパラメーターが、アプリケーションデプロイメントユーザーインタフェースに表示されます。
Enable Rollback	該当しない	このアプリケーション構成のロールバックおよびアンデプロイを有効または無効にすることができます。実際のロールバックおよびアンデプロイ動作は、アプリケーション構成自体に指定されています。
Enable Undeploy	該当しない	

次の例に示すアプリケーション構成は、多数の異なるネットワークにデプロイされるアプリケーションを対象としています。各ネットワークには、それぞれ異なるデータベースサーバーが存在します。このアプリケーションは、接続情報を`dbinfo.xml`というXML構成ファイルから取得します。アプリケーションがデータベースに接続するためには、`dbinfo.xml`内の4つの値が必要です。

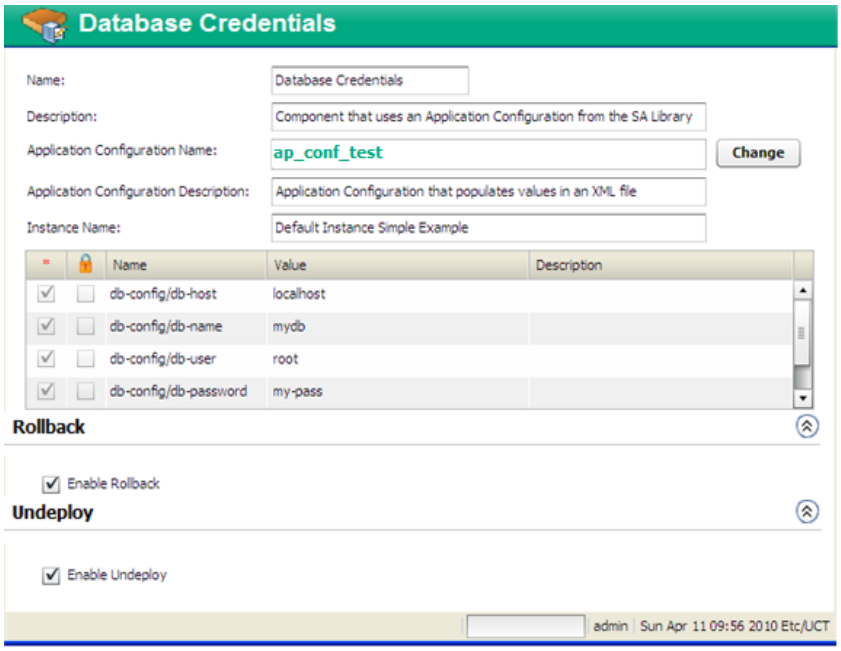
- **ホスト**: データベースがインストールされているサーバーのホスト名
- **名前**: ホストサーバー上のデータベースの名前
- **ユーザー**: データベースへの接続を開くために使用するユーザー名資格情報
- **パスワード**: データベースへの接続を開くために必要なパスワード

`dbinfo.xml`ファイルの例を次に示します。

```
<?xml version="1.0" ?>
<db-config>
<db-host>localhost</db-host>
<db-name>mydb</db-name>
<db-user>root</db-user>
<db-password>my-pass</db-password>
</db-config>
```

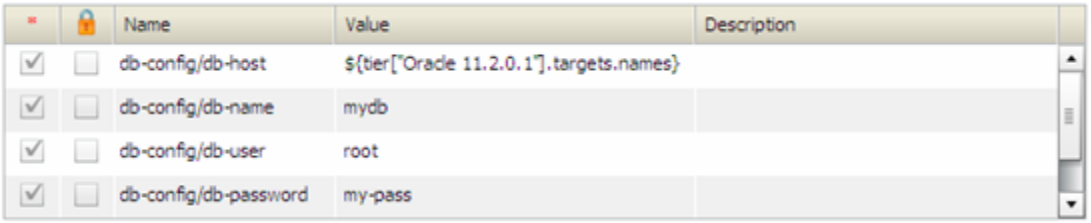
このアプリケーション構成に基づくコンポーネントを次に示します。この例では、`db-host`パラメーターが、ターゲットの名前を使用するようにデプロイメント時に変更されます。

アプリケーション構成コンポーネントの例



アプリケーション構成コンポーネントを設定するには、次の手順を実行します。

1. アプリケーション構成タイプのコンポーネントを新規作成します。
2. 使用するアプリケーション構成をSAライブラリから見つけて選択します。
3. パラメーターの値を変更するには、テーブルでそのパラメーターの [Value] セルをクリックします。定数値を指定することも、次に示すように特殊変数を選択して、デプロイメント時に値を割り当てることもできます。

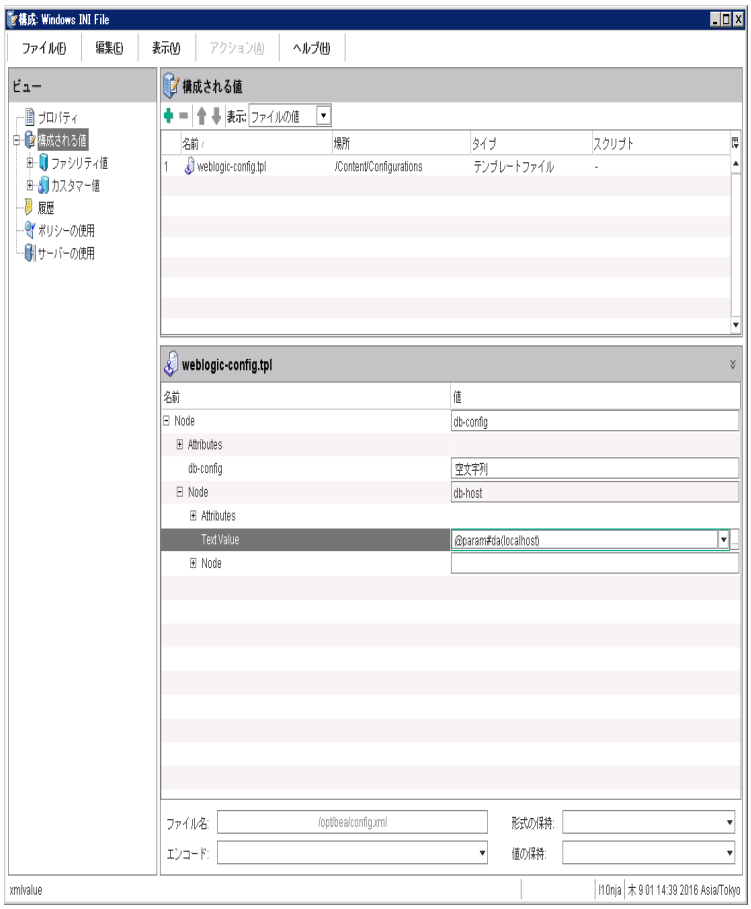


詳細については、「パラメーターと特殊変数」(506ページ)を参照してください。

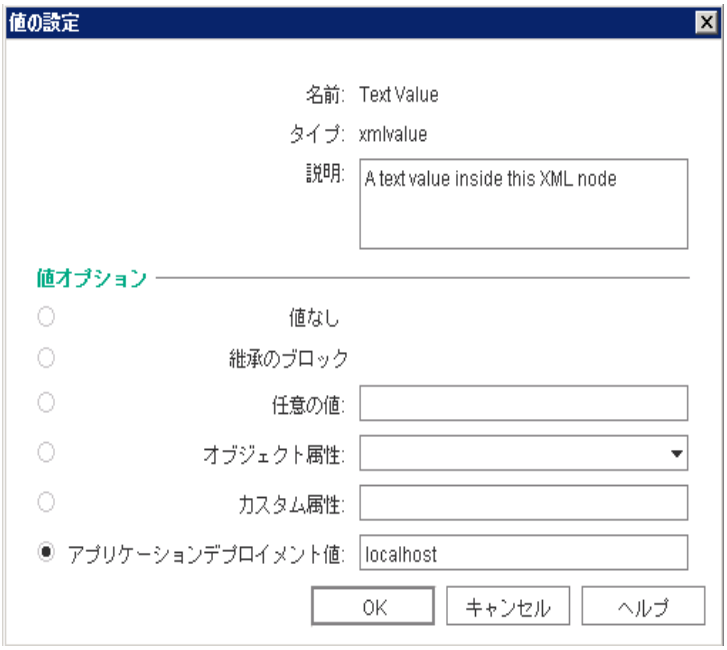
4. [Rollback] と [Undeploy] を有効または無効にします。

別の画面に移動する前に、必ずコンポーネントを保存してください。

アプリケーション構成パラメーターをアプリケーションデプロイメントユーザーインタフェースで利用可能にするには、次に示すように、SAサーバー上でそのパラメーターに対して [デプロイメント自動化の値] オプションを指定する必要があります。



このためには、パラメーターの [値] ボックスを選 択して、  ボタンをクリックします。[値 の設 定] ダイアログが開きます。



[デプロイメント自動化の値]オプションを選択します。デフォルト値を指定する場合は、その値をテキストボックスに入力します。

アプリケーション構成の詳細については、『SA 10.50ユーザーガイド』を参照してください。

ソフトウェアポリシーコンポーネント

Server Automation (SA) は、ソフトウェアポリシーを使用して、管理対象サーバー上でのソフトウェアのインストールとアプリケーションの構成のプロセスを自動化します。ソフトウェアポリシーには、パッケージ、RPMパッケージ、パッチ、アプリケーション構成、スクリプト、サーバーオブジェクトを含めることができます。また、OSシーケンスに関連付けることもできます。作成したソフトウェアポリシーは、サーバーまたはサーバーのグループにアタッチすることができます。これにより、管理対象サーバーが特定のポリシーに適合しているかどうかを評価できます。

SAライブラリにあるソフトウェアポリシーに基づくアプリケーションコンポーネントを作成できます。

ソフトウェアポリシーコンポーネントを作成する際には、次の情報を指定します。

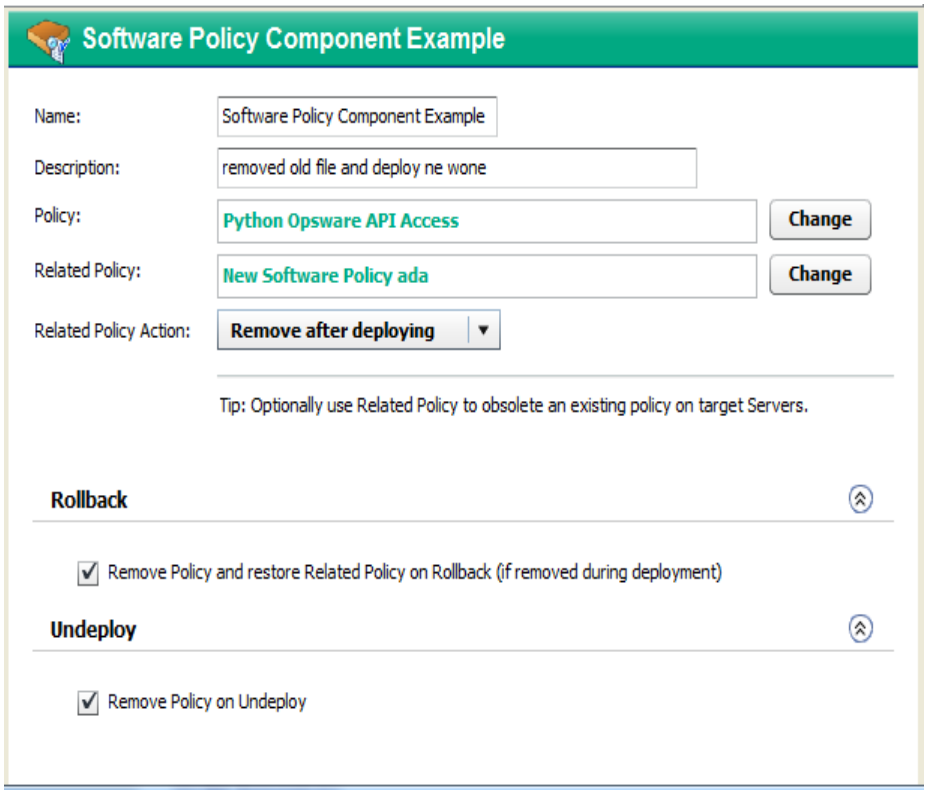
ソフトウェアポリシーコンポーネントのプロパティ

プロパティ	必須	目的
Name	はい	コンポーネントの名前。この名前は階層に表示されます。これはアプリケーション内で一意である必要があります。

ソフトウェアポリシーコンポーネントのプロパティ (続き)

プロパティ	必須	目的
Description	いいえ	レポートに表示される説明。
Policy	はい	アプリケーションのデプロイ時にターゲットサーバーにアタッチされて修復されるポリシー。
Related Policy	いいえ	コンポーネントに指定されたポリシーによって置き換えられる既存の (すでにサーバーに関連付けられている) ポリシー。
Related Policy Action	[Related Policy] が指定された場合	[Remove after deploying] (デフォルト)、[remove before deploying]、[do not remove] のいずれか。[Do Not Remove] を指定した場合、[Related Policy] はロールバック時だけに使用されます。
Rollback	該当しない	[Rollback] ボックスを選択した場合、アプリケーションデプロイメントがロールバックされると、以下のことが起こります。 [Policy] ボックスに指定されたソフトウェアポリシーが削除されます。 [Related Policy] ボックスに指定されたポリシーがデプロイメント中に削除された場合、このポリシーが復元されます。
Undeploy	該当しない	[Undeploy] ボックスを選択した場合、このアプリケーションがアンデプロイされるときに、[Policy] ボックスに指定されたソフトウェアポリシーが削除されます。

ソフトウェアポリシーコンポーネントの例



ソフトウェアポリシーコンポーネントを設定するには、次の手順を実行します。

1. ソフトウェアポリシータイプのコンポーネントを新規作成します。
2. 使用するポリシーと関連ポリシーをSAライブラリから見つけて選択します。
3. 必要な関連ポリシーアクションを選択します。
4. ロールバックおよびアンデプロイ動作を指定します。

別の画面に移動する前に、必ずコンポーネントを保存してください。

ソフトウェアポリシーの詳細については、SAオンラインヘルプの「ソフトウェアポリシーの作成と管理」を参照してください。

パッケージコンポーネント

SAソフトウェアリポジトリ内のアプリケーションは、パッケージ単位で構成されています。SAでパッケージをインストールするには、そのパッケージをソフトウェアポリシーに追加し、そのポリシーを1つ以上の管理対象サーバーにアタッチして、サーバーの修復を実行します。

SAライブラリにある1つ以上のパッケージと、オプションの関連ポリシーを含むアプリケーションデプロイメントコンポーネントを作成できます。また、新規パッケージをその場で作成して、そのパッケージをSAライブラリ

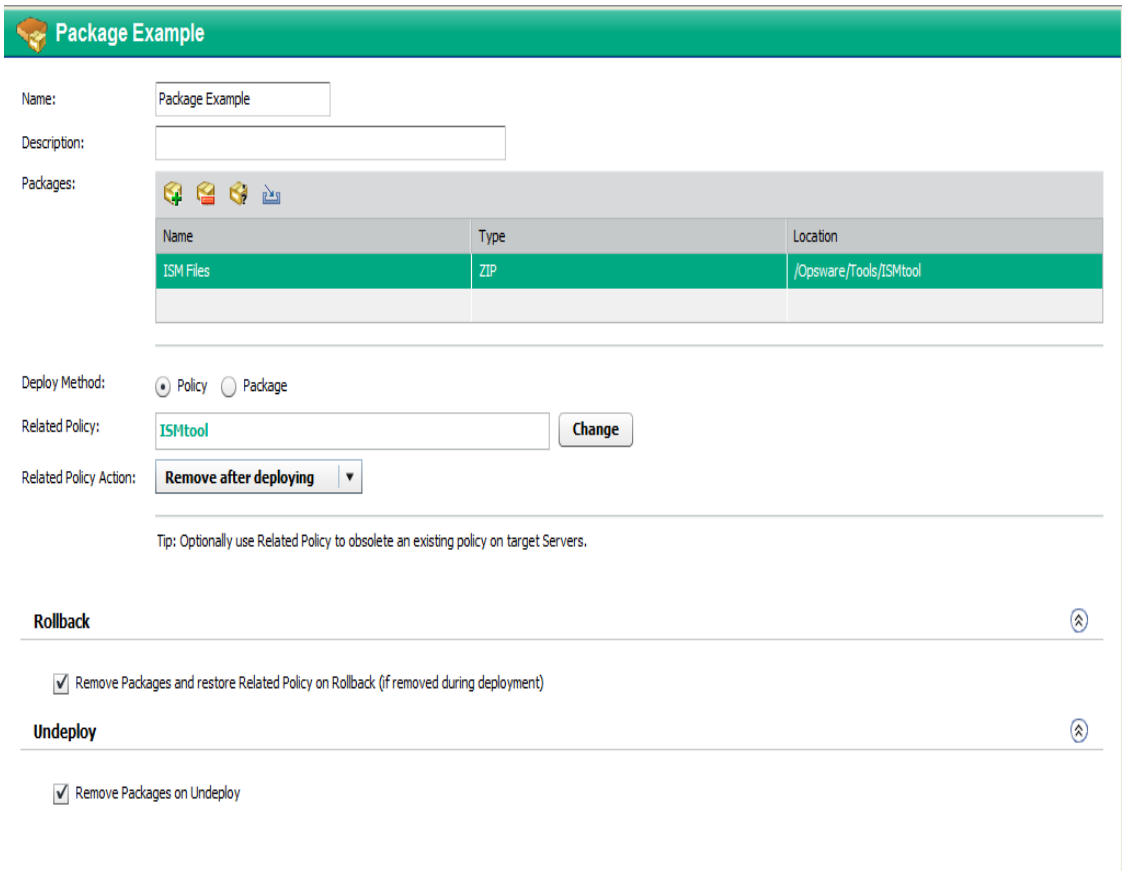
にインポートすることもできます。ポリシーをアタッチして通常のSAのサーバー修復ワークフローを使用することも、ポリシーをアタッチせずに単にパッケージをインストールすることもできます。

パッケージコンポーネントを作成する際には、次の情報を指定します。

パッケージコンポーネントのプロパティ

プロパティ	必須	目的
Name	はい	コンポーネントの名前。この名前は階層に表示されます。これはアプリケーション内で一意である必要があります。
Description	いいえ	レポートに表示される説明。
パッケージ	はい	SAライブラリにあるパッケージのリスト。
Deploy Method	はい	新規バージョンを作成する場合、アプリケーションデプロイメントは、指定したパッケージと、必要なスクリプトを含むパッケージを作成します。 Policy – アプリケーションデプロイメントは、新規バージョンを作成する際にポリシーも作成します。バージョンがデプロイされるときに、このポリシーが、各ターゲットサーバーにアタッチされ、修復されます。 Package – バージョンがデプロイされるときに、アプリケーションデプロイメントは、各ターゲットサーバーに対して、パッケージの「アドホック」インストールを実行します。
Related Policy Related Packages	いいえ	コンポーネントに指定されたパッケージによって置き換えられる既存のポリシーまたはパッケージ。
Related Packages Action	[Related Packages] を指定した場合	[Remove after deploying] (デフォルト)、[remove before deploying]、[do not remove] のいずれか。[Do Not Remove] を指定した場合、[Related Packages] はロールバック時だけに使用されます。
Rollback	該当しない	[Rollback] ボックスを選択した場合、アプリケーションデプロイメントがロールバックされると、以下のことが起こります。 パッケージが削除されます。 指定された [Related Policy] が削除された場合は、復元されます。指定された [Related Packages] は常に復元されます。
Undeploy	該当しない	[Undeploy] ボックスを選択した場合、このアプリケーションがアンデプロイされるときに、[Packages] テーブルに含まれるパッケージが削除されます。

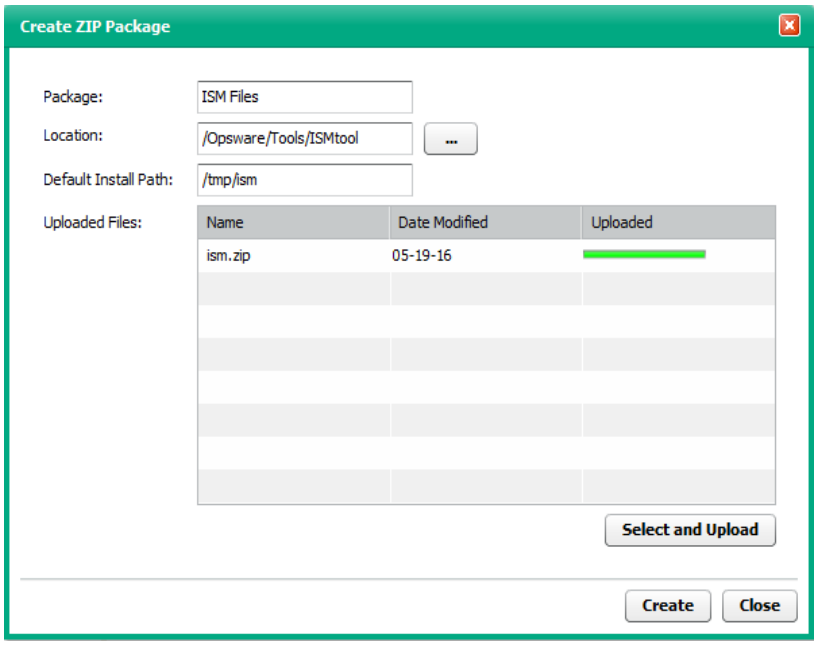
パッケージコンポーネントの例



次のツールを使用して、コンポーネント内のパッケージを指定することができます。

- [Add Package] ボタンをクリックして、SAライブラリからパッケージを選択します。これにより、[Select Package] ダイアログが開きます。「コンポーネントの操作」(496ページ)を参照してください。
- 検索文字列に一致するパッケージをすべて選択するには、ワイルドカード文字を使用します。このためには、次の手順を実行します。
  a. [Packages] テーブルでパッケージを選択します。
  b. [Wildcard Package Name] ボタンをクリックします。[Set Wildcard] ダイアログが開きます。
  c. [Name] ボックスのテキストを変更して、ワイルドカード文字 *および?を使用して検索文字列を作成します。
  d. [**OK**] をクリックします。

  新しいバージョンを作成する際には、SAライブラリ内の同じ場所にある、検索文字列とパッケージタイプに一致する最新のパッケージが含められます。
- 新しいZIPパッケージを作成するには、[**Create ZIP Package from Files**] をクリックします。これにより、[Create ZIP Package] ダイアログが開きます。



[Location] は、新しいパッケージが置かれるSAライブラリ内のフォルダーです。

[Default Install Path] は、アプリケーションがデプロイされるときにパッケージがインストールされるターゲット上の場所です。

新しいパッケージを作成するには、次の手順を実行します。

1. [**Select and Upload**] をクリックします。
2. パッケージに含めるローカルシステム上のファイルを参照します。
3. [**Create**] をクリックしてZIPパッケージを作成し、SAライブラリにインポートします。

これで、SAライブラリの他のパッケージと同様に、パッケージをデプロイできるようになります。

パッケージコンポーネントを設定するには、次の手順を実行します。

1. パッケージタイプのコンポーネントを新規作成します。
2. [**Add Package**] ツールまたは [**Import Package**] ツールを使用して、含めるパッケージを指定します。
3. デプロイ方法を指定します (「ソフトウェアポリシーコンポーネントのプロパティ」(485ページ)を参照)。
4. [**Add Package**] ツールまたは [**Import Package**] ツールを使用して、含める関連パッケージを指定し、[**Related Policy Action**] を選択します。
5. ロールバックおよびアンデプロイ動作を指定します。

別の画面に移動する前に、必ずコンポーネントを保存してください。

ソフトウェアパッケージの詳細については、SAオンラインヘルプの「ソフトウェアパッケージの管理」を参照してください。

OOフローコンポーネント

このタイプのコンポーネントは、**HPE Operations Orchestration (OO)** ワークフローを開始します。これは、他のアプリケーションと統合する場合に有用です。例:

- サーバーの特定のグループが更新されたことを示すHPE Service Managerトラブルチケットの発行
- デプロイメントについて知るためのロードバランサーの設定
- アプリケーション監視システムとの通信

SAコアサーバーがHPE OOサーバーと統合するように設定されている場合、HPE OOワークフローを含むアプリケーションデプロイメントコンポーネントを作成することができます。

また、特定の環境にHPE OOワークフローを関連付けることもできます。この方法を使用すると、次のタイプのシナリオを作成できます。

1. 環境レベルで、「デプロイメント前」OOワークフローを開始します。
2. アプリケーションをデプロイします。
3. 環境レベルで、「デプロイメント後」OOワークフローを開始します。

例: 監視を10分間停止→アプリケーションをデプロイ→監視を再開。

QA環境では、たとえば、夜間ビルドが終了するたびにスモークテスト (検証) を実行することができます。その後、製品所有者に電子メールを送信して、テスト結果が許容できるかどうかを確認できます。電子メールで回答を受け取ったら、新規ビルドをデプロイできます。

詳細については、「環境の管理」(604ページ)を参照してください。

OOフローコンポーネントを作成する際には、次の情報を指定します。

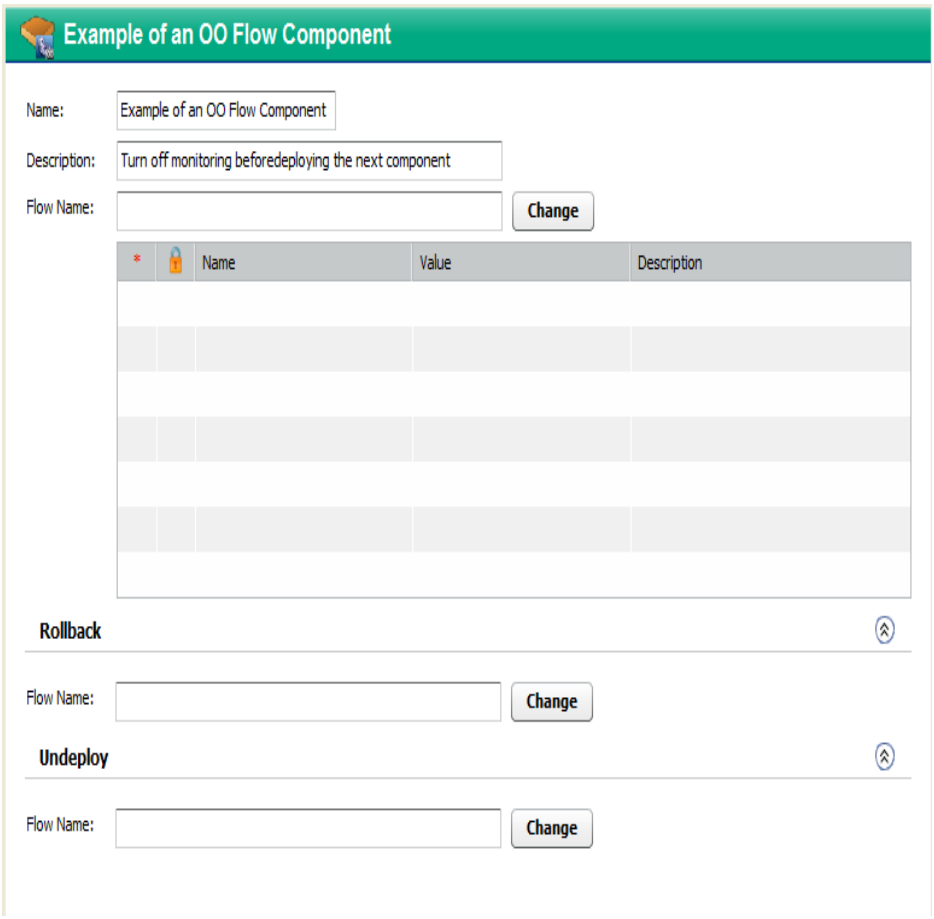
OOフローコンポーネントのプロパティ

プロパティ	必須	目的
Name	はい	コンポーネントの名前。この名前は階層に表示されます。これはアプリケーション内で一意である必要があります。
Description	いいえ	レポートに表示される説明。
Flow Name	はい	HPE OOワークフローへの参照。
Flow Parameters	選択したフローに	選択したHPE OOワークフローに必要なパラメーター。

OOフローコンポーネントのプロパティ (続き)

プロパティ	必須	目的
	依存	同じ名前のパラメーターは、機能、ロールバック、アンデプロイの各セクションで共通に使用されます。
Rollback Flow	いいえ	デプロイメントが失敗した場合にロールバックに使用するHPE OOワークフロー。
Undeploy Flow	いいえ	アプリケーションがアンデプロイされるときに使用されるHPE OOワークフロー。

OOフローコンポーネントの例



OOフローコンポーネントを設定するには、次の手順を実行します。

1. OOフロータイプのコンポーネントを新規作成します。
2. HPE OOライブラリからHPE OOワークフローを選択します。

   OOフローを検索する場合、テキストボックスに単語全体を指定することで、検索を大幅に高速化

できます。OOフローの検索は、単語全体だけに一致します。たとえば、このコンテキストでは “start” は “restart” には一致しません。

3. パラメーターの値を変更するには、テーブルでそのパラメーターの [Value] セルをクリックします。定数値を指定することも、特殊変数を選択して、デプロイメント時に値を割り当てることもできます (「パラメーターと特殊変数」(506ページ)を参照)。
4. ロールバックおよびアンデプロイ動作を指定します。

別の画面に移動する前に、必ずコンポーネントを保存してください。

Windowsレジストリコンポーネント

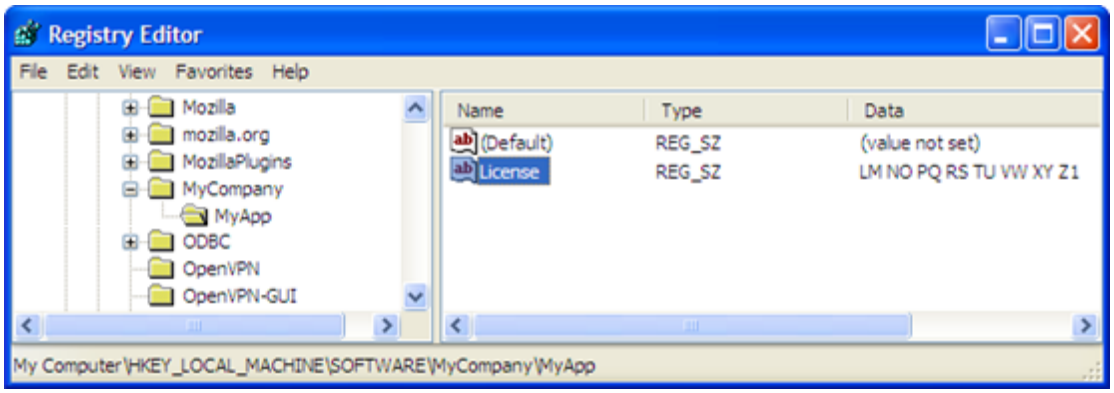
このタイプのコンポーネントは、**Windowsレジストリ**のキーまたは値を追加または削除するために使用します。このタイプのコンポーネントは、Windowsアプリケーションでのみ使用できます。

Windowsレジストリコンポーネントを作成する際には、次の情報を指定します。

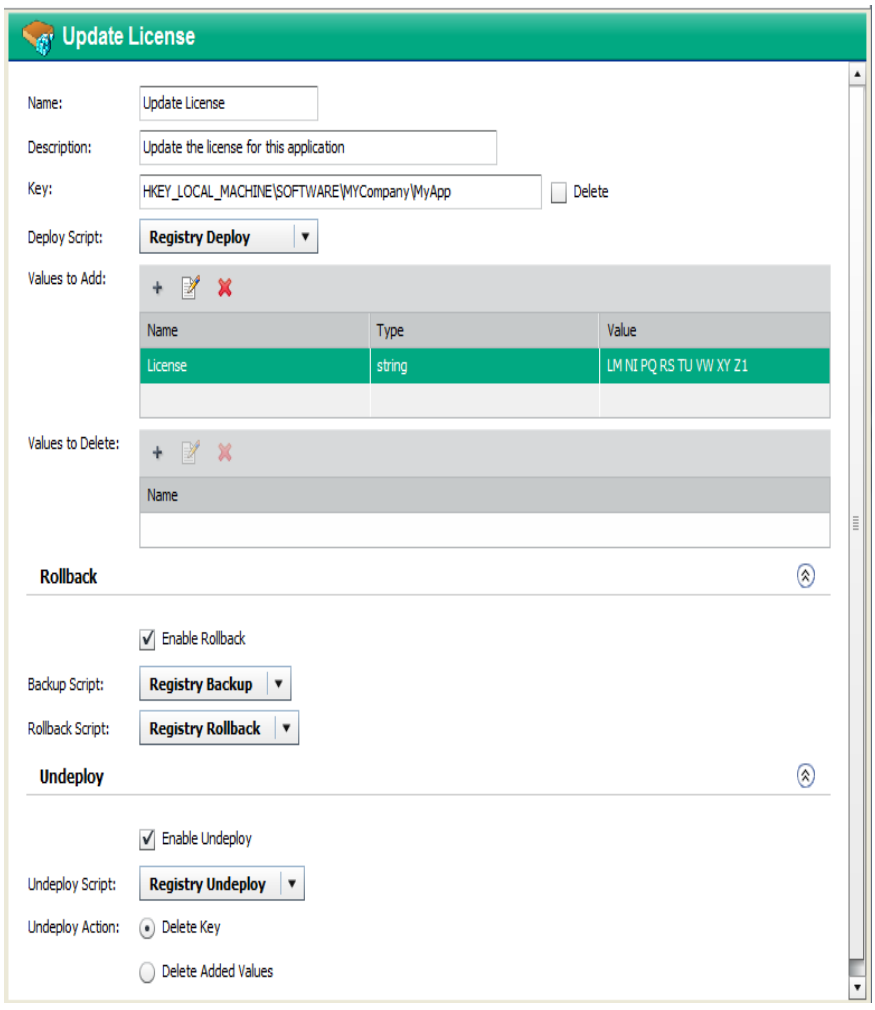
Windowsレジストリコンポーネントのプロパティ

プロパティ	必須	目的
Name	はい	コンポーネントの名前。この名前は階層に表示されます。これはアプリケーション内で一意である必要があります。
Description	いいえ	レポートに表示される説明。
Key	はい	変更するレジストリキー。
Values	はい	指定したキーに対して追加または削除する値。キーを削除する場合は、値は不要です。
Backup Script	ロールバックが有効な場合	バックアップ、ロールバック、アンデプロイスクリプトは、アプリケーションデプロイメント管理者が管理します。詳細については、「スクリプトの管理」(609ページ)を参照してください。
Rollback Script		
Undeploy Script	アンデプロイが有効な場合	
Undeploy Action	アンデプロイが有効な場合	キー全体とすべての値を削除するか、追加された値だけを削除します。

次のレジストリコンポーネントは、このライセンスキーの値をレジストリに追加します。



Windowsレジストリコンポーネントの例



キー名と値は正しい形式で指定することが重要です。次の点に注意してください。

- [Key] フィールドの先頭は有効なハイブ名である必要があります。次のハイブ名は有効です。HKEY_CLASSES_ROOT、HKCR、HKEY_CURRENT_USER、HKCU、HKEY_LOCAL_MACHINE、HKLM、HKEY_USERS、HKU、HKEY_CURRENT_CONFIG、HKEY_PERFORMANCE_DATA、HKEY_DYN_DATA。
- キーコンポーネントの区切り文字としては、バックスラッシュだけが使用できます。例:
- HKEY_LOCAL_MACHINE\SOFTWARE\Hewlett-Packard\HP Virtual Rooms 8.0
- Binaryタイプのキー値の [Value] フィールドは、スペースで区切られた2桁の16進数の列で指定する必要があります。例: 01 A2 B3 C4
- Dwordタイプのキー値の [Value] フィールドは、0xで始まる16進数か、正の10進数で指定する必要があります。

Windowsレジストリコンポーネントを設定するには、次の手順を実行します。

1. Windowsレジストリタイプのコンポーネントを新規作成します。
2. 新しいキー値を追加するには、次の手順を実行します。
   a. [Values to Add] テーブルで [**Add**] をクリックします。
   b. [Add Value] ダイアログで、[**Name**]、[**Type**]、[**Value**] を指定します。

      [**Value**] に値を入力することも、特殊変数を選択して、デプロイメント時に値を割り当てることもできます (「コンポーネントのタイプ」(470ページ)を参照)。
   c. [**OK**] をクリックします。
3. 既存のキー値を削除するには、次の手順を実行します。
   a. [Values to Delete] テーブルで [**Add**] をクリックします。
   b. [Add Value to Delete] ダイアログで、[**Name**] に名前を指定します。
   c. [**OK**] をクリックします。
4. どちらかのテーブルの値を変更するには、その値を選択して、[**Edit Properties**] をクリックします。
5. どちらかのテーブルから値を削除するには、その値を選択して、[**Delete**] をクリックします。
6. ロールバックおよびアンデプロイ動作を指定します。

別の画面に移動する前に、必ずコンポーネントを保存してください。

コンポーネントの操作

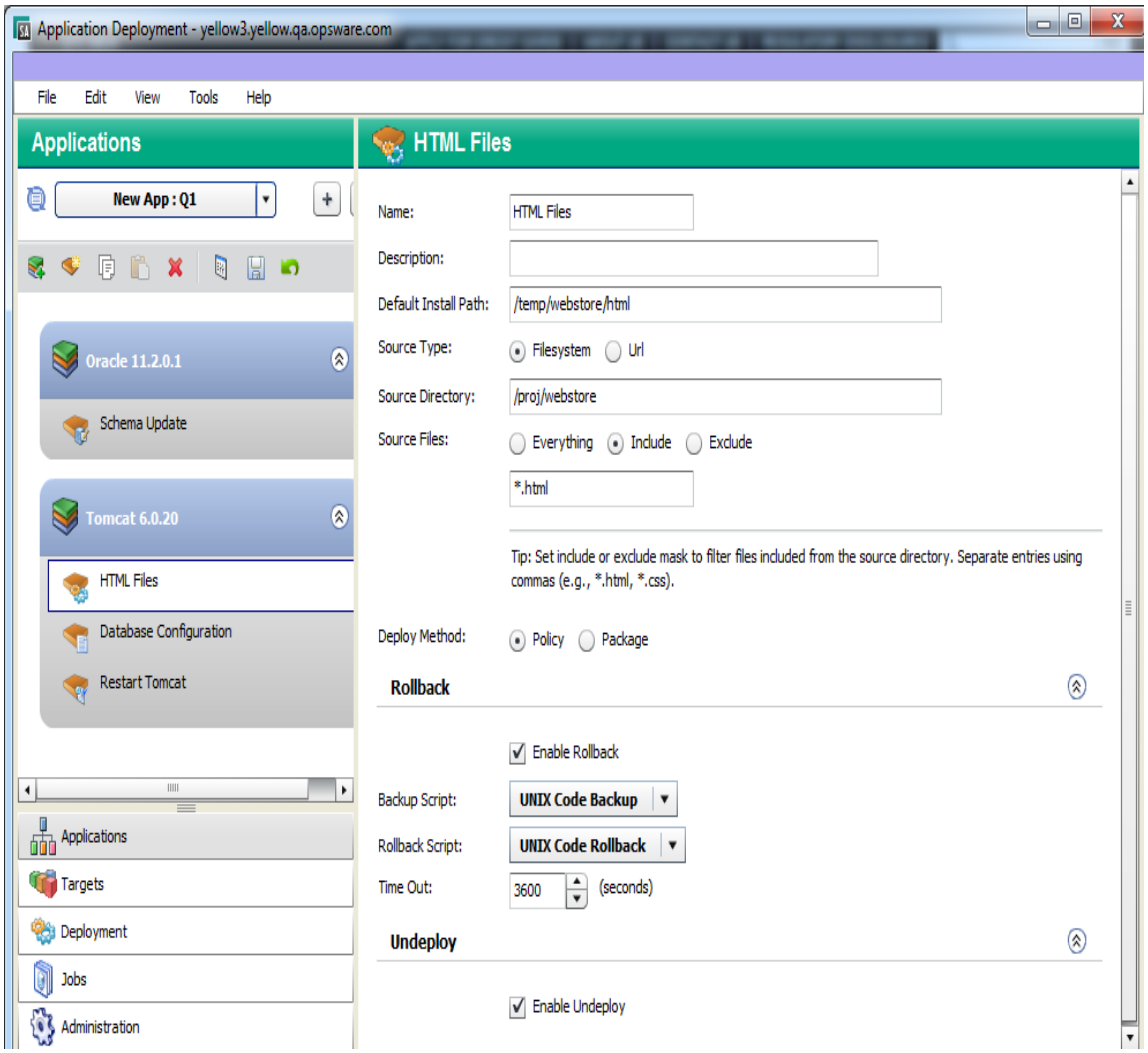
コンポーネントは、アプリケーションの**内容**を表します。各コンポーネントは、アプリケーションの一部をデプロイします。

コンポーネントは、[Applications] 画面の右側のペインで設定します。コンポーネントは特定の階層に属します。階層はアプリケーションの**構造**を表現します。

設定するコンポーネントのタイプに応じてプロパティは異なりますが、コンポーネントの作成とアクセスのプロセスはほぼ共通です。

コンポーネントを追加して変更することができるのは、アプリケーションを編集するアクセス権がある場合に限ります。

コードコンポーネントの例



すべてのコンポーネントには、プロパティのセットが3つあります。

機能プロパティ(ペインの一番上)は、コンポーネントを一意に識別し、その動作を指定し、このコンポーネントに含まれるアイテムがどこにあるかをアプリケーションデプロイメントに知らせる役割を果たします。

ロールバックプロパティは、デプロイメントが失敗し、秩序あるロールバックが必要になった場合に、このコンポーネントをどう処理するかをアプリケーションデプロイメントに知らせる役割を果たします。

特定のタイプのコンポーネントでは、ロールバックを有効にする場合にロールバックアクションを指定する必要があります。これは、コンポーネントのタイプに応じて、スクリプト、フロー、または明示的命令の集合です。また、バックアップアクションの指定が必要な場合もあります。この場合、バックアップアクションは、コンポーネントがデプロイされる前に実行されます。ロールバックアクションは、デプロイメントが失敗した場合に、バックアップされたアイテムを復元します。

アンデプロイプロパティは、アプリケーションをアンデプロイする指示が与えられた場合に、このコンポーネントをアンインストールする方法をアプリケーションデプロイメントに知らせる役割を果たします。

ロールバックおよびアンデプロイスクリプトは、アプリケーションデプロイメント管理者によって管理されます(「スクリプトの管理」(609ページ)を参照)。

新しいコンポーネントの作成

新しいアプリケーションコンポーネントを作成するには、[New Component] ダイアログを使用します。新しいコンポーネントを作成できるのは、編集アクセス権を持っているアプリケーションに対してです(「アプリケーションのアクセス権」(597ページ)を参照)。

コンポーネントは特定の階層と関連付けられます。新しいコンポーネントを作成できるのは、階層が選択されている場合だけです。階層に対応するプラットフォームによって、作成できるコンポーネントのタイプが決まります。

新しいコンポーネントを作成するには、次の手順を実行します。

1. 新しいコンポーネントを含む階層を選択します。階層を選択するには、次の2つの方法があります。
   - 階層名をクリックします。
   - その階層内の任意のコンポーネントを選択します。
2. ツールバーの [**New Component**] をクリックします。
3. [New Component] ダイアログで、[**Component Name**] と [**Component Type**] を指定します。コンポーネント名は、アプリケーション内で一意である必要があります。
4. [**OK**] をクリックします。
5. 新しいコンポーネントのプロパティに、適切な値を入力します。詳細については、「コンポーネントのタイプ」(470ページ)を参照してください。
6. [**Save**] をクリックして変更内容を保存します。

変更を保存せずに [Applications] 画面を終了しようとすると、変更を保存するか破棄するかを尋ねられます。

既存のコンポーネントの変更

アプリケーションの編集アクセス権を持つユーザーは、アプリケーションのコンポーネントを変更することができます(「アプリケーションデプロイメントのアクセス権」(597ページ)を参照)。左側のペインでコンポーネントを選択し、右側のペインでそのプロパティを変更できます。実行できる変更の種類は、コンポーネントのタイプに応じて異なります。

既存のコンポーネントを変更するには、次の手順を実行します。

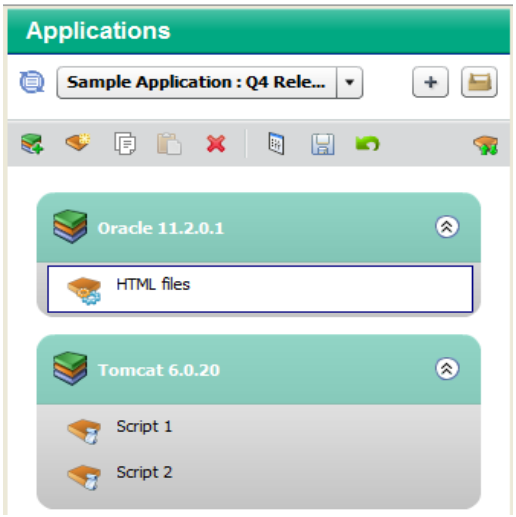
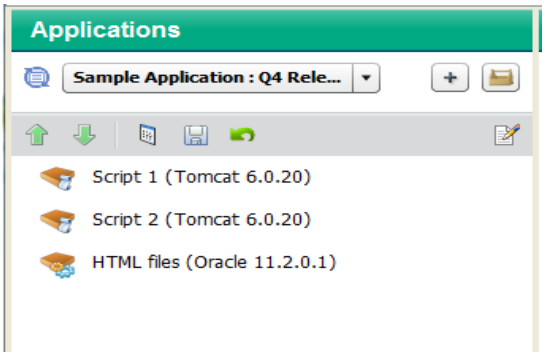
1. 階層内でコンポーネントを選択します。
2. 右側のペインのツールを使用してプロパティを変更します。
3. [**Save**] をクリックして変更内容を保存します。

変更を保存せずに [Applications] 画面を終了しようとすると、変更を保存するか破棄するかを尋ねられます。

コンポーネントのデプロイメント順序の変更

アプリケーションの編集アクセス権を持つユーザーは、コンポーネントがターゲットにデプロイされる順序を変更できます (「アプリケーションのアクセス権」(597ページ)を参照)。このためには、左側のペインを、階層ビュー (デフォルト) から、次に示すデプロイメント順序ビューに変更する必要があります。

コンポーネントビュー

階層ビュー	デプロイメント順序ビュー
	

たとえば、アプリケーションのファイルがデプロイされた後で、アプリケーションを開始することが必要な場合があります。この場合、アプリケーションを開始するスクリプトコンポーネントが、ファイルをデプロイするコードコンポーネントよりも、デプロイメント順序で後に来るようにする必要があります。

順序を変更しない限り、コンポーネントは作成された順序でデプロイされます。

コンポーネントのデプロイメント順序を変更するには、次の手順を実行します。

1. [**Edit Component Deployment Order**] をクリックします。

左のペインが変化して、現在のアプリケーションに関連付けられたすべてのコンポーネントが表示されます。リストの順序はデプロイされる順序に一致します。

2. コンポーネントを選択し、緑の矢印 ( と )を使用して、リスト内でコンポーネントを上または下に移動します。このビューでは、階層はコンポーネント名の後の括弧の中に示されています。矢印ボタンでコンポーネントを移動しても、階層は元のままで変わりません。
3. 階層ビューに戻るには、[Edit Properties] ボタンをクリックします。 

階層ビューでは、コンポーネントを別の階層にドラッグアンドドロップすることができます。一部のコンポーネントは、特定のオペレーティングシステム向けの階層でのみ使用できます。たとえば、レジストリコンポーネントが使用できるのはWindows階層だけです。

コンポーネントを別の階層にドラッグアンドドロップする場合、デプロイメント順序に注意してください。デフォルトでは、コンポーネントは階層の最後に配置されます。

コンポーネントのコピーと貼り付け

既存のコンポーネントをコピーして、同じアプリケーションまたは別のアプリケーションの互換性のある階層に貼り付けることができます。互換性のある階層とは、同じOSプラットフォーム上で動作するものです。

コンポーネントをコピーして貼り付けた場合、既存のコンポーネントのすべてのプロパティが新しいコンポーネントにコピーされます。

コピーして貼り付けられるコンポーネントは一度に1つだけであり、コンポーネントをコピーして貼り付けるためには、対象のアプリケーションに対する編集アクセス権が必要です (「アプリケーションのアクセス権」(597ページ)を参照)。

コンポーネントをコピーして貼り付けるには、次の手順を実行します。

1. コピーするコンポーネントを選択します。

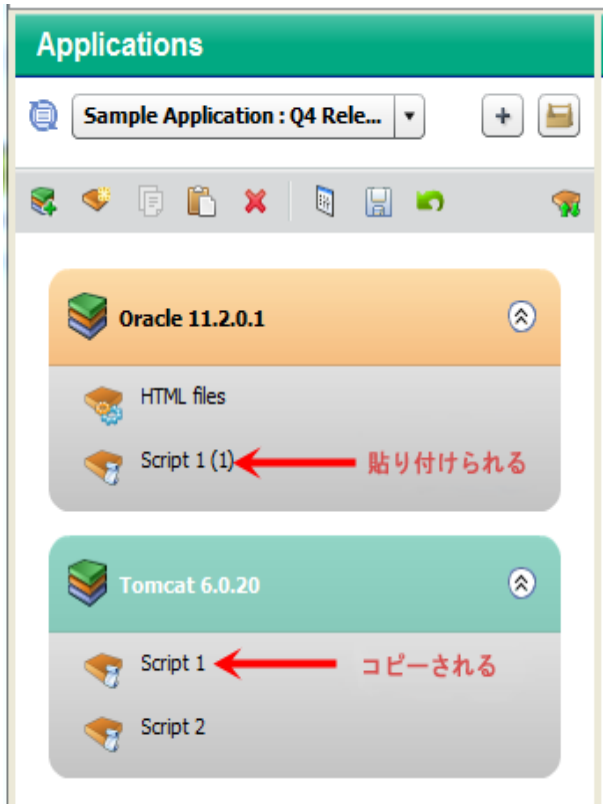
   そのコンポーネントを含む階層がアクティブになり、ツールバーの [**Copy**] ボタン  が使用可能になります。

   [**Copy**] ボタンが使用できない場合は、このアプリケーションに対する編集アクセス権がありません。
2. ツールバーの [**Copy**] をクリックします。
3. オプション: 別のアプリケーションにコンポーネントを貼り付けるには、そのアプリケーションを編集のために開きます。

4. コンポーネントを貼り付ける階層を選択します。

   コピーを実行したのと同じ階層に貼り付ける場合は、選択の必要はありません。

5. ツールバーの [**Paste**] をクリックします。貼り付けられたコンポーネントの名前には、次の図に示すように番号が追加されます。



   [**Paste**] が使用できない場合は、このアプリケーションに対する編集アクセス権がありません。

6. [**Save**] をクリックして変更内容を保存します。

コンポーネントの移動

「ドラッグアンドドロップ」を使用して、コンポーネントを、1つの階層から、同じアプリケーション内の互換性のある別の階層に移動することができます。互換性のある階層とは、同じOSプラットフォーム上で動作するものです。

移動できるコンポーネントは一度に1つだけであり、コンポーネントを移動するためには、対象のアプリケーションに対する編集アクセス権が必要です (「アプリケーションのアクセス権」(597ページ)を参照)。

ドラッグアンドドロップを使用してコンポーネントを移動するには、次の手順を実行します。

1. 移動するコンポーネントを、互換性のある階層のコンポーネント領域にドラッグします。

新しい階層が元の階層と互換性がある場合は、緑のプラス記号が表示されます。
新しい階層に互換性がないか、同じ階層にコンポーネントをドロップしようとした場合は、赤いXが表示されます。

2. 新しい階層にコンポーネントをドロップします。
3. [**Save**] をクリックして変更内容を保存します。

次の例は、HTML Filesコンポーネントを Tomcat 6.0.20階層からOracle 11.2.0.1階層に移動する操作を示します。

コンポーネントを別の階層に移動する例

実行前:


Oracle 11.2.0.1
HTML files
Tomcat 6.0.20
Script 1
Script 2
スクリプト1
コンポーネントが
中にあります

実行中:


Oracle 11.2.0.1
HTML files
Script 1
Tomcat 6.0.20
Script 1
Script 2
この互換階層まで
ドラッグします

実行後:


Oracle 11.2.0.1
Script 1
HTML files
Tomcat 6.0.20
Script 2
この階層の中に
入りました

コンポーネントの削除

自分が編集アクセス権を持っているアプリケーションに属するコンポーネントを削除することができます(「アプリケーションのアクセス権」(597ページ)を参照)。

コンポーネントを削除するには、次の手順を実行します。

1. コンポーネントを選択します。
2. ツールバーの [**Delete Selected Item**] をクリックします。
3. [**Yes**] をクリックして確認します。

一度に削除できるコンポーネントは1つだけです。

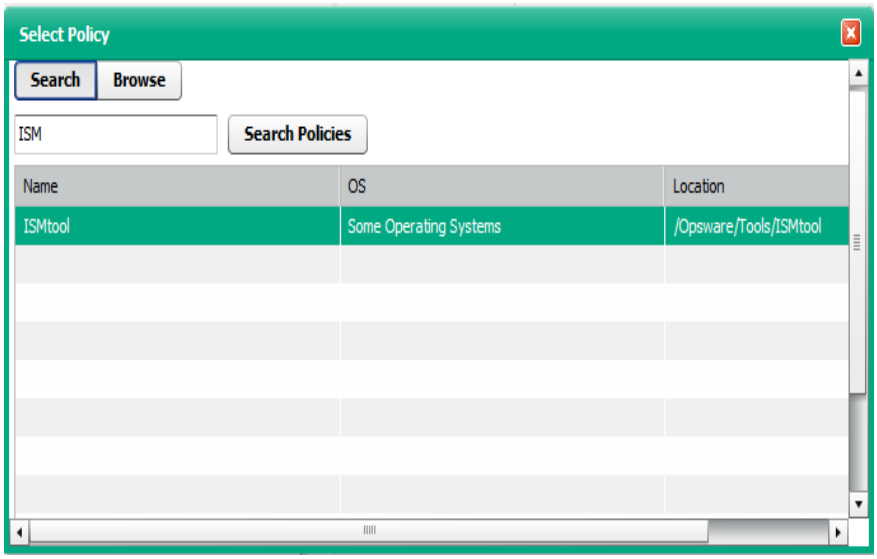
ライブラリ内のアイテムの検索と選択

ここに記す情報は、次のタイプのコンポーネントに当てはまります。

- 「アプリケーション構成コンポーネント」(481ページ)
- 「コンポーネントの操作」(496ページ)
- 「コンポーネントの操作」(496ページ)
- 「コンポーネントの操作」(496ページ)

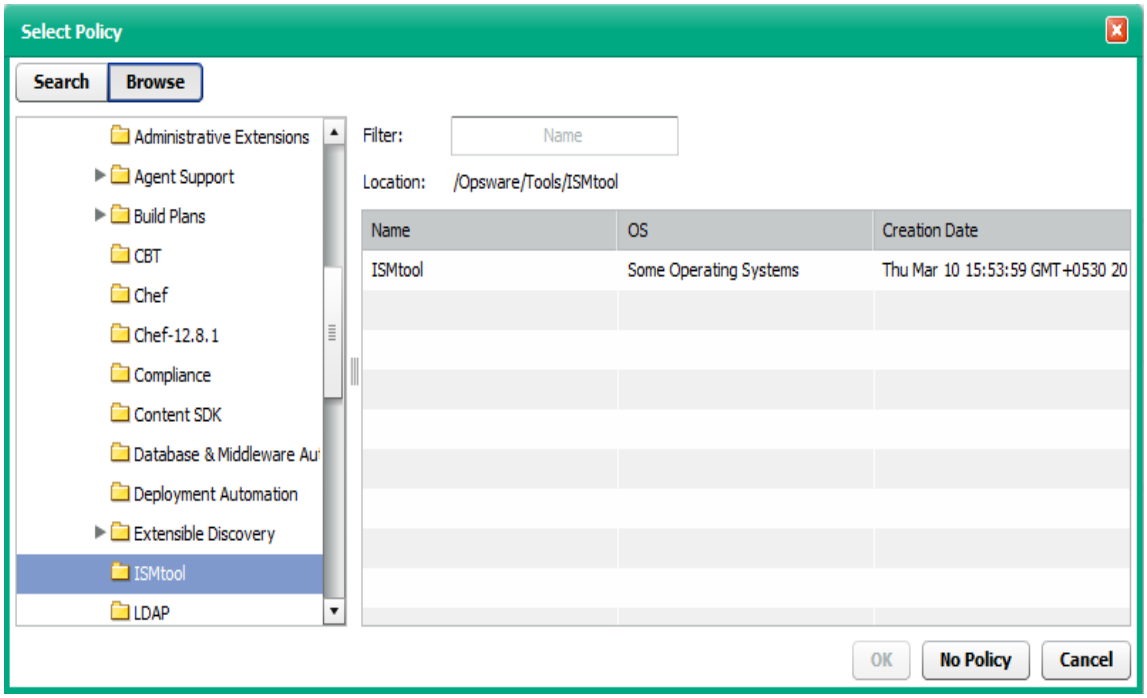
どの場合でも、次のようなダイアログを使用して、特定のアイテムに関連するライブラリ(SA OO)を検索します。

ライブラリアイテムの選択ウィンドウ - [Search] タブ



この例では、SAライブラリで、名前に“ISM”が含まれるポリシーを検索しています。上記のようにテーブルビューでライブラリを検索するほかに、ライブラリのフォルダー階層内を参照することもできます。

ライブラリアイテムの選択ウィンドウ - [Browse] タブ



この例では、`/Opsware/Tools/ISMtool`フォルダー内で、名前に“Linux”が含まれるすべてのポリシーが検索に一致しています。

以下の手順では、[Applications] 画面でリリースが選択されていることを前提としています。“Item”はプレースホルダーで、先に示したライブラリアイテムのどれかを示します。

ライブラリ内のアイテムを検索して選択するには、次の手順を実行します。

1. 左のペインで、編集 (または新規コンポーネントを作成) するコンポーネントのタイプを1つ選択します。
2. アイテムに対して [**Change**] をクリックします (または、パッケージコンポーネントを対象とする場合は  アイコンをクリックします)。

   [Select Item] ダイアログが開きます。
3. オプション: パッケージを指定する場合、パッケージのタイプをリストから選択します。RPM、ZIP、MSI (Windows階層のみ)、またはAll Typesが選択できます。
4. テーブルビューを使用して検索するには、次の手順を実行します。
   a. テキストボックスに検索文字列を入力します。ライブラリ内のすべてのアイテムを表示するには、テキストボックスを空にしておきます。

      OOフローを検索する場合、テキストボックスに単語全体を指定することで、検索を大幅に高速化できます。検索は単語全体だけに一致します。たとえば、このコンテキストでは “start” は “restart” には一致しません。
   b. [**Search Item**] をクリックします。

   フォルダー階層を使用して検索するには、次の手順を実行します。
   a. [**Browse**] ボタンをクリックします。表示アクセス権があるすべてのフォルダーが表示されます。
   b. オプション: フィルターを指定します。
5. テーブルで、目的のアイテムを選択します。
6. [**OK**] をクリックします。

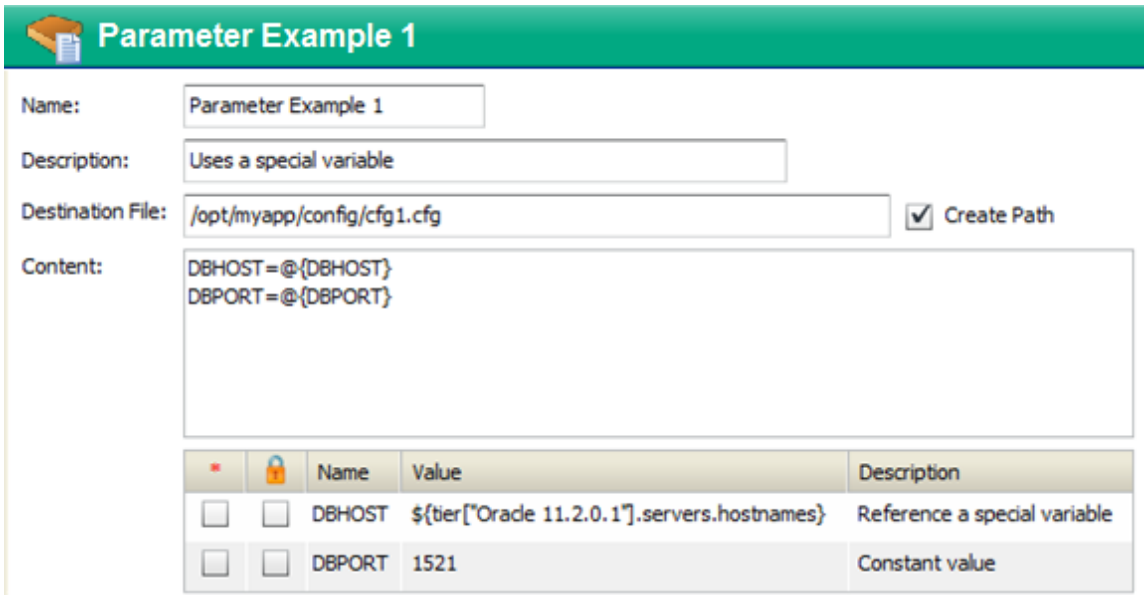
既存のコンポーネントからライブラリアイテムを削除するには、[**No Item**] ボタンをクリックします。このボタンは、ソフトウェアポリシーおよびOOフローコンポーネントに対して使用できます。たとえば、以前に指定したポリシーを削除するには、[**No Policy**] をクリックします。

パラメーターと特殊変数

いくつかのタイプのコンポーネントでは、**コンポーネントパラメーター**を使用して、アプリケーション、リリース、バージョン、ターゲット、階層、または環境によって異なる情報を表すことができます。

次に示す構成ファイルコンポーネントでは、DBHOSTおよびDBPORTパラメーターを使用して、データベースサーバーのホスト名とデータベースポートを表しています。

パラメーターの例



パラメーター参照の記法は次のとおりです。`@{parameter_name}`

[Content] ボックスでこの記法を使用してパラメーターを参照する場合、パラメーターの定義は、[Content] ボックスのすぐ下にあるテーブルにあります。

- パラメーターの名前は、[Content] ボックスで中括弧の中にあるものです。
- コンポーネントパラメーター名には、スペース、\、@、{、}は使用できません。
- * (必須) 列のチェックボックスがオンになっている場合、そのパラメーターはコンポーネントのデプロイ時に値を持つ必要があります。
- (暗号化) 列のチェックボックスがオンになっている場合、そのパラメーターは暗号化され、値はアプリケーションデプロイメントユーザーインタフェースのどこにも表示されません。

パラメーターの値は、この例のDBPORTパラメーターのように単純な数値または文字列の場合も、**変数**の場合もあります。変数の値は、デプロイメント時に割り当てられます。この例のDBHOSTパラメーターは、変数を参照しています。この変数は、指定されたターゲットのOracle 11.2.0.1階層に含まれるサーバーのホスト名のカンマ区切りリストを表します。

パラメーター値の指定には、特殊変数、リリースパラメーター、グローバルパラメーターの3つのタイプの変数が使用できます。それぞれのタイプには、固有の目的と機能があります (「変数のタイプ」(508ページ)を参照)。

コンポーネント定義でパラメーターの値を指定する場合、その値はパラメーターのデフォルト値になります。この値は、デプロイメント時に変更できます。これはたとえば、特定のターゲットや環境に対してデフォルト値をオーバーライドする場合に便利です (「デプロイメントパラメーターのカスタマイズ」(551ページ)を参照)。

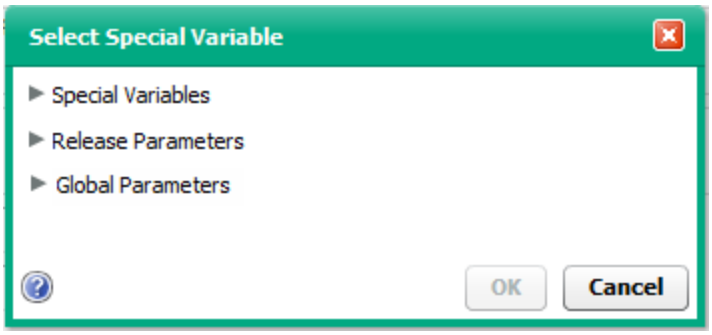
パラメーターの作成、管理、参照の詳細については、「パラメーターと変数の操作」(513ページ)を参照してください。パラメーターに関するルールと制約については、「特別な考慮事項」(518ページ)を参照してください。

変数のタイプ

パラメーター定義で参照できる変数には、次の3つのタイプがあります。

タイプ	説明
特殊変数	特殊変数は、アプリケーションデプロイメントに組み込まれています。この変数は、ユーザーが[Deployment]画面でバージョンとターゲットを指定するまで決定されない情報を表します。 特殊変数は、デプロイされるもの(アプリケーション、リリース、またはバージョン)、デプロイ先(ターゲットまたは環境)、デプロイするユーザー(SAユーザー)のいずれかに関連します。 「特殊変数」(510ページ)を参照してください。
リリースパラメーター	リリースパラメーターは、リリース内のすべてのコンポーネントから使用できます。作成と管理はアプリケーションの所有者が行い、アプリケーションの編集アクセス権を持つすべてのユーザーが変更することができます。リリースパラメーターの値は、明示的に変更しない限り変化しません。 リリースパラメーターは、リリースの特定のバージョン内で同じ値を持ちます。 「リリースパラメーター」(511ページ)を参照してください。
グローバルパラメーター	グローバルパラメーターは、すべてのアプリケーションから使用できます。作成と管理はアプリケーションデプロイメント管理者が行います。グローバルパラメーターの値の指定は、管理者が明示的に変更しない限り変化しません。 グローバルパラメーターの実際の値は、環境とターゲットに応じて変わることがあります。 「グローバルパラメーター」(512ページ)を参照してください。

3つのタイプの変数は、すべて[Select Special Variable]ダイアログから参照できます。



このダイアログには、アプリケーションデプロイメントプロセスのさまざまな時点でアクセスできます(「[Select Special Variable] ダイアログへのアクセス」(518ページ)を参照)。

デプロイメント時に、アプリケーションデプロイメントは、特殊変数、リリースパラメーター、グローバルパラメーターへの参照を再帰的にたどることにより、各コンポーネントパラメーターの値を解決します。

たとえば、LogDirという名前のコンポーネントパラメーターが次のように定義されているとします。

		Name	Value	Description
☐	☐	LogDir	${release=>LogDir}	References release parameter LogDir

リリースパラメーターAppDirとLogDirは、次のように定義されています。

Release Parameters:

		Name	Value	Description
☐	☐	AppDir	${global=>AppBaseDir}/ThisApp	References global parameter AppBaseDir
☐	☐	LogDir	${release=>AppDir}/log	References release parameter AppDir

グローバルパラメーターAppBaseDirは/optに定義されています。

Global Parameters

		Name	Value	Description
☐	☐	AppBaseDir	/opt	All applications are stored under /opt

バージョンのデフォルトのパラメーターは次のようになります。

	Name	Component	Type	Value
	AppBaseDir	-	GLOBAL	/opt
	AppDir	-	VERSION	${global=>AppBaseDir}/ThisApp
	LogDir	-	VERSION	${release=>AppDir}/log
	LogDir	Show Recursive Parameter Resolution	DEPLOY	${release=>LogDir}

ターゲットまたは環境のオーバーライドがない場合、コンポーネントのデプロイ時のコンポーネントパラメーターLogDirの最終的な値は、/opt/ThisApp/logになります。

Parameters:	Name	Component	Type	Value
	LogDir	Show Recursive Parameter Resolution	DEPLOY	/opt/ThisApp/log

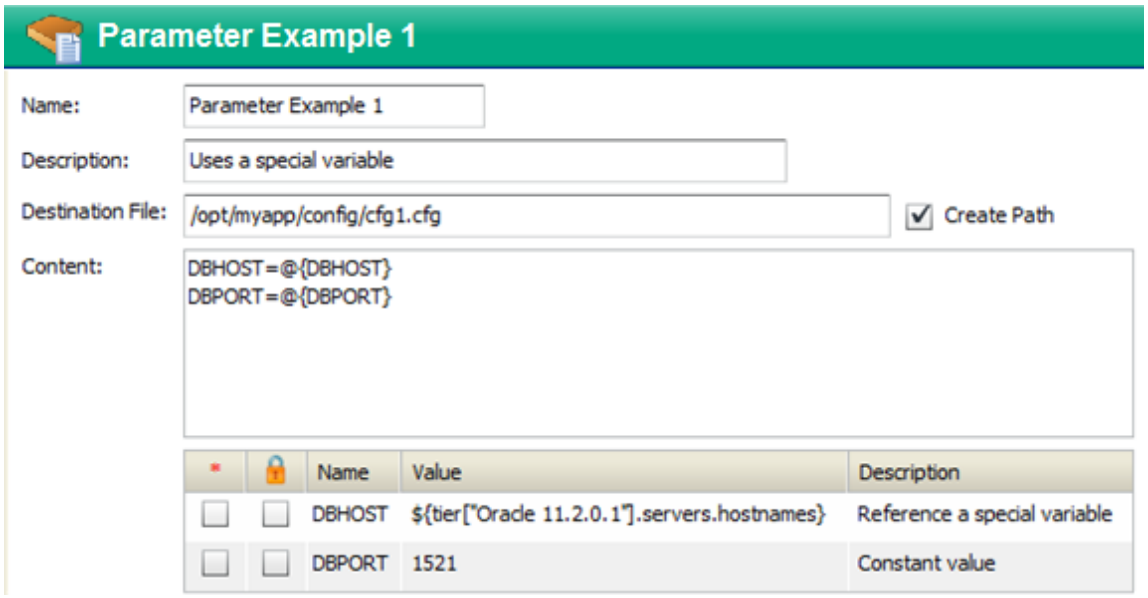
特殊変数

特殊変数は、次のタイプのアイテムに関する情報を表します。

- コンポーネントがデプロイされるサーバー
- そのサーバーを含む階層
- コンポーネントが属するアプリケーション
- デプロイされているリリースとバージョン
- ターゲットが存在する環境
- デプロイを開始したSAユーザー

次の例で、DBHOSTパラメーターは、変数を参照しています。この変数は、Oracle 11.2.0.1階層に含まれるサーバーのホスト名のカンマ区切りリストを表します。

特殊変数を参照するパラメーター



一部の特殊変数の値は前もって知られていることもありますが、特殊変数の値の割り当て(バインド)はデプロイメント時まで行われません。

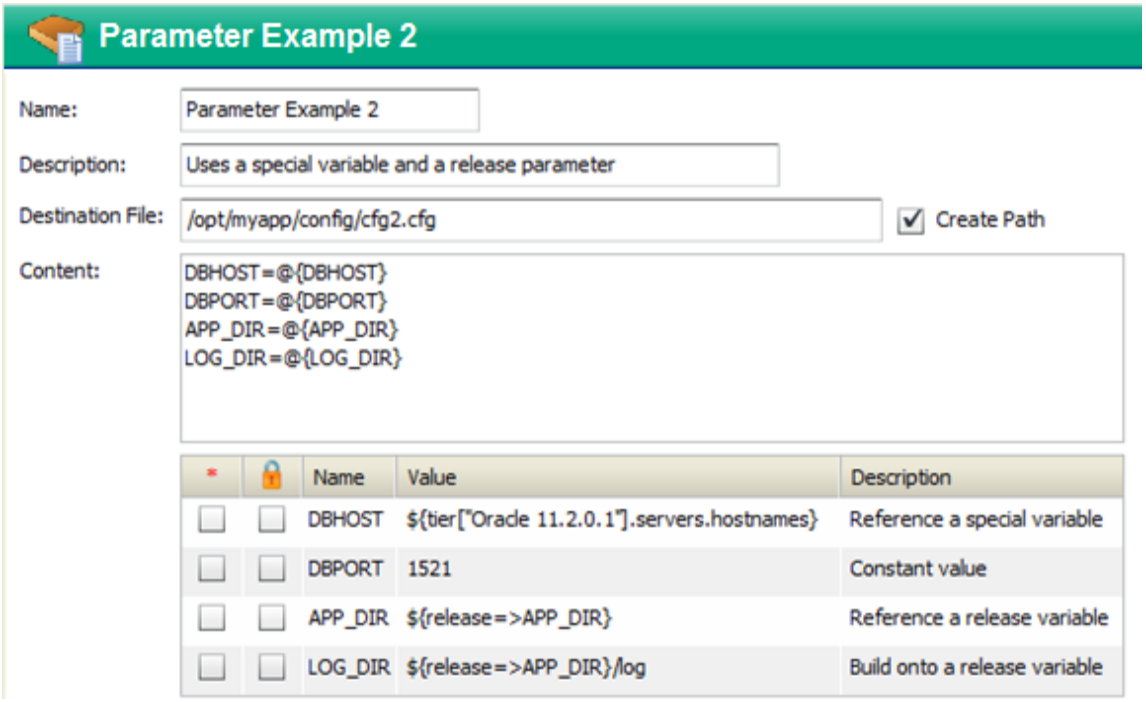
「サーバー名」特殊変数は、SA内のサーバーの名前を参照します。この名前は、実際のサーバーホスト名と無関係に、SAで変更できます。「ホスト名」特殊変数は、サーバーの実際のホスト名を参照します。SAのサーバー名は、ホスト名と一致する場合もしない場合もあります。

パラメーター定義での特殊変数の参照については、「パラメーターの値の指定」(515ページ)を参照してください。

リリースパラメーター

リリースパラメーターの値は、リリースの特定のバージョン内で同一です。これらは、リリースの任意のコンポーネントから参照できます。次の例では、APP_DIRリリースパラメーターを使用して、アプリケーションのインストールディレクトリを表しています。

リリースパラメーターの例



リリースパラメーターへの参照の前後に情報を追加することで、パラメーター値を構築することができます。この例では、LOG_DIRパラメーターの値で、"/log"というテキストが、APP_DIRリリースパラメーターの値に追加されています。

リリースパラメーターは、グローバルパラメーター、特殊変数、および他のリリースパラメーター(同じリリース内のもの)を参照することができます。

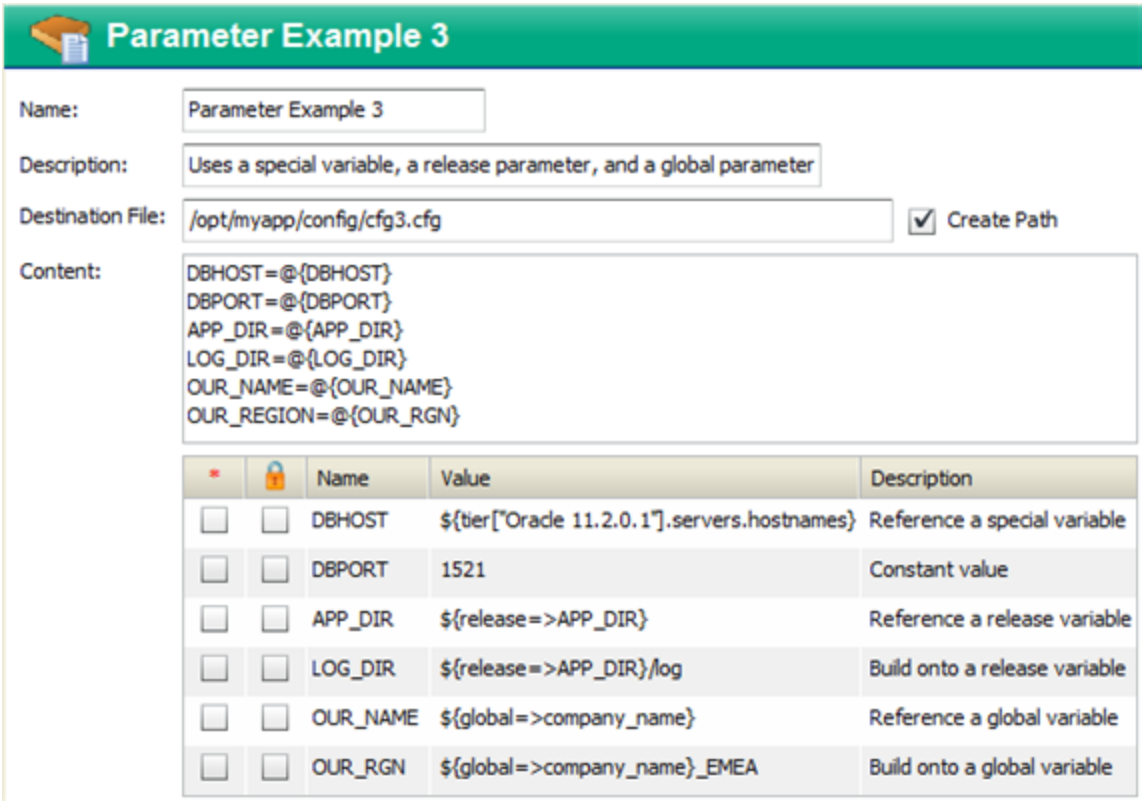
アプリケーションの編集アクセス権を持つユーザーは、そのアプリケーションの任意のリリースのリリースパラメーターを定義または変更できます(「リリースパラメーターの定義」(516ページ)を参照)。

新しいリリースが既存のリリースから複製される場合、リリースパラメーターも複製されます。

グローバルパラメーター

グローバルパラメーターは、すべてのアプリケーションで同一です。これらは、任意のアプリケーションの任意のコンポーネントから参照できます。また、リリースパラメーターからも参照できます。次の例では、OUR_NAMEグローバルパラメーターを使用して、会社の名前を表しています。

グローバルパラメーターを使用した例



リリースパラメーターの場合と同様に、グローバルパラメーターへの参照の前後に情報を追加することで、パラメーター値を構築することができます。この例では、OUR_RGNパラメーターの値で、"_EMEA"というテキストが、company_nameグローバルパラメーターの値に追加されています。

グローバルパラメーターは、特殊変数や他のグローバルパラメーターを参照することができます。

グローバルパラメーターの管理は、アプリケーションデプロイメント管理者が行います(詳細については「アプリケーション設定の管理」(616ページ)を参照)。

パラメーターと変数の操作

このセクションのトピックでは、次の作業の実行方法を説明します。

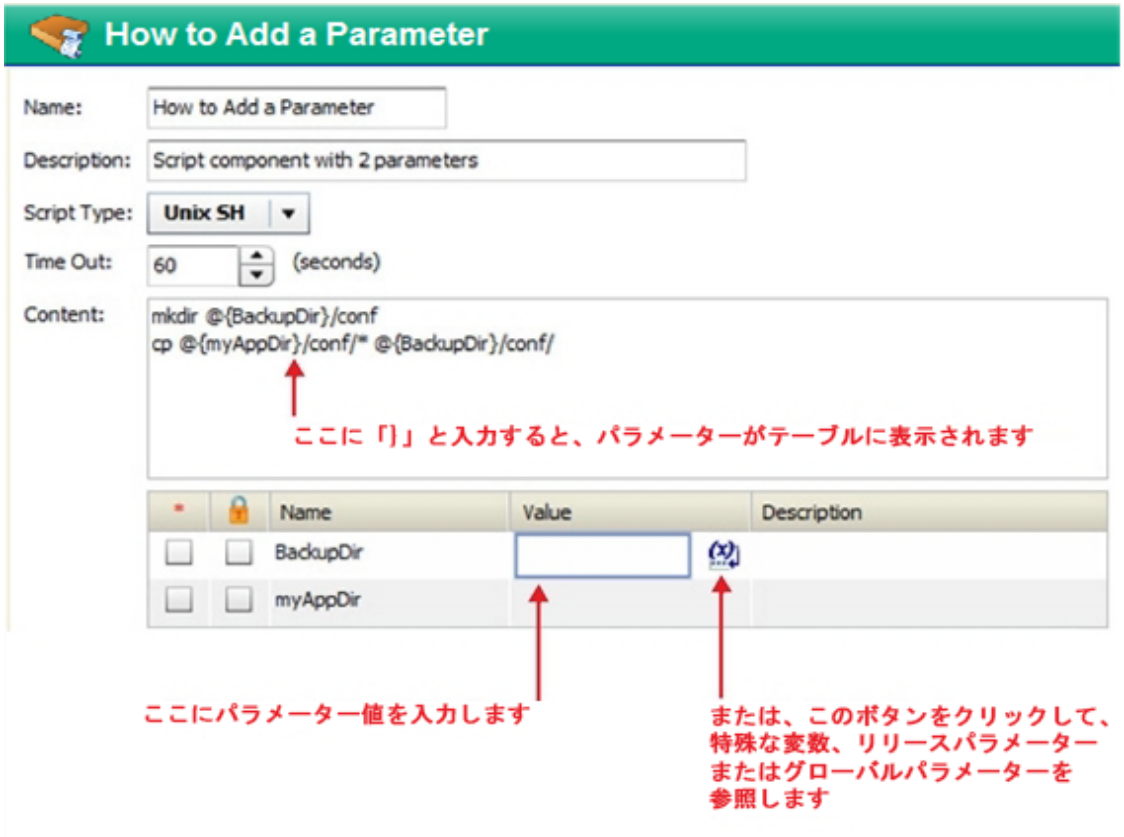
- 「コンポーネントパラメーターの指定」(514ページ)
- 「パラメーターの値の指定」(515ページ)
- 「[Select Special Variable] ダイアログへのアクセス」(518ページ)
- 「リリースパラメーターの定義」(516ページ)
- 「リリースパラメーターの削除」(517ページ)

コンポーネントパラメーターの指定

アプリケーションの編集 (または作成) アクセス権を持つユーザーは、そのアプリケーションに含まれるスクリプトまたは構成ファイルコンポーネントの内容に、パラメーターを追加することができます。

OOフローおよびアプリケーション構成コンポーネントでも、パラメーターを使用できます。これらのコンポーネントタイプのパラメーターの値を変更することはできますが、パラメーター自体を追加したり削除したりすることはできません。

スクリプトコンポーネントでのパラメーター参照の例



スクリプトまたは構成ファイルコンポーネントの内容にパラメーターを追加するには、次の手順を実行します。

1. [Content] ボックスに、次の記法を使用してパラメーターの名前を入力します。
   `@{parameter_name}`

   パラメーター名には、スペース、@、\、{、}は使用できません。

2. オプション: コンポーネントのデプロイ時にパラメーターが値を持つ必要がある場合は、* (必須) 列のチェックボックスをオンにします。
3. パラメーター値を指定します。定数値を指定するか、変数を参照するか、その両方を使用することができます (手順については「パラメーターの値の指定」(515ページ)を参照)。
4. オプション: パラメーターを暗号化するには、 (暗号化) 列のチェックボックスをオンにします。
5. ツールバーの [Save Release] ボタンをクリックして、変更内容を保存します。

パラメーターの値の指定

アプリケーションの編集アクセス権を持つユーザーは、コンポーネントパラメーターのデフォルト値を変更することができます。また、リリースパラメーターの値も変更できます。定数値 (テキストまたは数値) を指定するか、次の表に示す任意のアイテムを参照できます。

変数参照の形式

変数タイプ	形式	例
特殊変数	`${special_variable_name}`	`${application.name}`
リリースパラメーター	`${release=>parameter_name}`	`${release=>APP_DIR}`
グローバルパラメーター	`${global=>parameter_name}`	`${global=>APP_BASE_DIR}`

[Select Special Variable] ダイアログを使用すると、関連する特殊変数、リリースパラメーター、グローバルパラメーターのリストから変数を選択できます。

次の例に示すように、テキストと変数参照を組み合わせることができます。

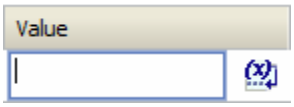
```
${release=>APP_DIR}/log
```

```
/var/${application.name}/my_data
```

```
${global=>TestBaseDir}/${application.name}
```

パラメーターの値を指定するには、次の手順を実行します。

1. パラメーターテーブルで、指定するパラメーターの [Value] 列のどこかをクリックします。[Value] 列にテキストボックスとボタンが表示されます。



2. 次の手順を実行して、特殊変数、リリースパラメーター、またはグローバルパラメーターの参照をリストから選択します。
   a. ボタンをクリックします。[Select Special Variable] ダイアログが開きます。
   b. 参照する特殊変数、リリースパラメーター、またはグローバルパラメーターを選択します。
   c. [**OK**] をクリックします。変数参照が [Value] 列に表示されます。
3. オプション: パラメーター値に含める追加のテキストを指定します。
4. ツールバーの [**Save Release**] をクリックして、変更内容を保存します。

アプリケーションのデプロイアクセス権と、適切な環境への書き込みアクセス権を持つユーザーは、デプロイメント時に、パラメーターエディターで同様の手順を実行することにより、デフォルトパラメーターの値をオーバーライドすることができます (「デプロイメントパラメーターのカスタマイズ」(551ページ)を参照)。

リリースパラメーターの定義

アプリケーションの編集アクセス権を持つユーザーは、リリースパラメーターを定義または変更することができます。これらのパラメーターは、リリースに含まれるすべてのコンポーネントで使用できます。リリースの外部では使用できません。

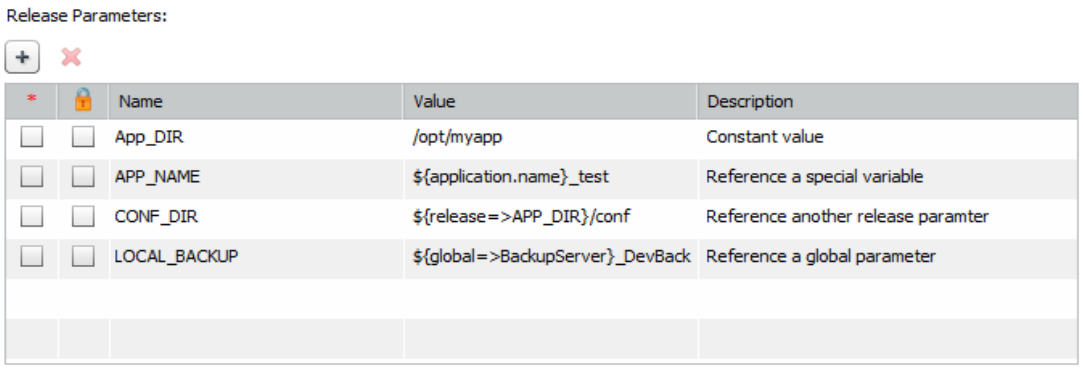
リリースパラメーターを定義または変更するには、次の手順を実行します。

1. [Application] 画面で、操作するリリースを選択します。
2. ツールバーで、[**Edit release properties**] をクリックします。[Release Attributes] ダイアログが開きます。

   このダイアログは、Manage Applicationsツールから開くこともできます (「アプリケーションの管理」(460ページ)を参照)。
3. 新しいリリースパラメーターを追加するには、次の手順を実行します。
   a. [**Add Release Parameter**] ボタンをクリックします。
   b. 新しいパラメーターの名前を入力します。名前はリリースパラメーターの中で一意である必要があります。名前に次の文字を使用することはできません: @、\、{、}
   c. [**OK**] をクリックします。
4. このパラメーターの [**Value**] 列のどこかをクリックします。

5. パラメーターの値を指定します(詳細な手順については「パラメーターの値の指定」(515ページ)を参照)。

   値の指定では、特殊変数、他のリリースパラメーター、グローバルパラメーターを参照できます。例:



6. オプション: パラメーター値を暗号化するには、(暗号化)列のチェックボックスをオンにします。このコンテキストでは、* (必須)列は無関係です。
7. ツールバーの [**Save Release**] をクリックして、変更内容を保存します。

リリースパラメーターの削除

アプリケーションの編集アクセス権を持つユーザーは、コンポーネントパラメーターまたは他のリリースパラメーターから参照されていないリリースパラメーターを削除することができます。使用されているリリースパラメーターは削除できません。

リリースパラメーターを削除するには、次の手順を実行します。

1. [Application] 画面で、操作するリリースを選択します。
2. ツールバーで、[**Edit release properties**] をクリックします。[Release Attributes] ダイアログが開きます。
3. [**Release Parameters**] テーブルで、削除するパラメーターを選択します。
4. [**Delete Parameter**] をクリックします。確認ダイアログが表示されます。
5. [**Yes**] をクリックして削除を確認します。
6. ツールバーの [**Save Release**] をクリックして、変更内容を保存します。

[Select Special Variable] ダイアログへのアクセス

[Select Special Variable] ダイアログは、アプリケーションデプロイメントユーザーインタフェースのさまざまな場所から開くことができます。

[Select Special Variable] ダイアログを開く方法

場所	パラメータータイプ	手順
[Application] 画面	コンポーネント	コンポーネントエディターで、コンポーネントパラメーターの [Value] 列のどこかをクリックします。
リリース	1. [**Edit release properties**] をクリックします。 2. リリースパラメーターの [Value] 列のどこかをクリックします。	1. [Deployment Options] 領域の [Parameters] セクションを展開します。 2. [Parameters] セクションの左上にある [**Edit**] ボタンをクリックします。[Parameter Editor] ダイアログが開きます。 3. 右端の列で、リスト内のパラメーターの値をダブルクリックします。[Parameter Values] ダイアログが開きます。 4. [Target Value] または [Environment Value] ボックスの右側の  ボタンをクリックします。
[Deployment] 画面	すべて	
[Administration] 画面	グローバル	1. [Application Settings] ページに移動します。 2. リリースパラメーターの [Value] 列のどこかをクリックします。

特別な考慮事項

パラメーターの定義を指定する際には、次の制約と考慮事項に注意してください。

サイクル

パラメーター定義に循環参照 (サイクル) を使用することはできません。たとえば、次の指定を使用すると、リリースの保存時にエラーが発生します。

*		Name	Value	Description
☐	☐	FirstOne	${global=>SecondOne}	
☐	☐	SecondOne	${global=>ThirdOne}	
☐	☐	ThirdOne	${global=>FirstOne}	

アプリケーションを保存した後でサイクルが発生した場合 (たとえば、グローバルパラメーターを後で変更した場合など)、このリリースの新しいバージョンに対するデプロイメントジョブを作成することはできなくなります。[Deployment] 画面に「無限ループ」のエラーが報告されます。

自己参照

パラメーターは自分自身を参照することはできません。

削除

既存のバージョンのコンポーネントパラメーターまたはリリースパラメーターでグローバルパラメーターが参照されている場合、そのグローバルパラメーターを削除することはできません。

現在のリリースのコンポーネントパラメーターまたはリリースパラメーターでリリースパラメーターが参照されている場合、そのリリースパラメーターを削除することはできません。

グローバルパラメーターAでグローバルパラメーターBが参照されている場合、グローバルパラメーターBを削除することはできません。

削除されたグローバルパラメーターへの参照がバージョンに含まれている場合、デプロイメントジョブは作成されません。

削除されたリリースパラメーターを参照するコンポーネントが含まれるリリースを保存した場合、アプリケーションツールバーに警告アイコン⚠が表示されます。警告アイコンをクリックすると、どのパラメーターが削除されたかを知ることができます。

名前

コンポーネントパラメーター、リリースパラメーター、グローバルパラメーターの名前には、次の文字は使用できません: @、\、{、}

また、スクリプトまたは構成ファイルコンポーネントのコンポーネントパラメーター名にスペースは使用できません。OOフローコンポーネントコンポーネントパラメーター名、リリースパラメーター名、グローバルパラメーター名にはスペースが使用できます。

リリースパラメーター名は、特定のリリースのリリースパラメーターリスト内で一意である必要があります。

グローバルパラメーター名は、グローバルパラメーターリスト内で一意である必要があります。

リリースパラメーターの名前は、コンポーネントパラメーターまたはグローバルパラメーターと一致していてもかまいません。

グローバルパラメーターの名前は、コンポーネントパラメーターまたはリリースパラメーターと一致していてもかまいません。

インポートとエクスポート

グローバルパラメーターとリリースパラメーターは、インポートとエクスポートが可能です (「コンテンツのインポートとエクスポート」(623ページ)を参照)。

部分インポートでインポートされるグローバルパラメーターは、少なくとも1つのリリースパラメーターまたはコンポーネントパラメーターから参照されているものだけです。

オーバーライド

リリースのパラメーターに対して作成された環境またはターゲットの値のオーバーライドは、元のリリースから複製されたすべてのリリースで使用されます。

グローバルパラメーターとリリースパラメーターのターゲットおよび環境オーバーライドの優先順位は、コンポーネントパラメーターの場合と同じです。ターゲットオーバーライドは環境オーバーライドよりも優先され、環境オーバーライドはデフォルト値よりも優先されます。

ターゲット

単純な例については、「クイックスタート」(440ページ)を参照してください。

概要

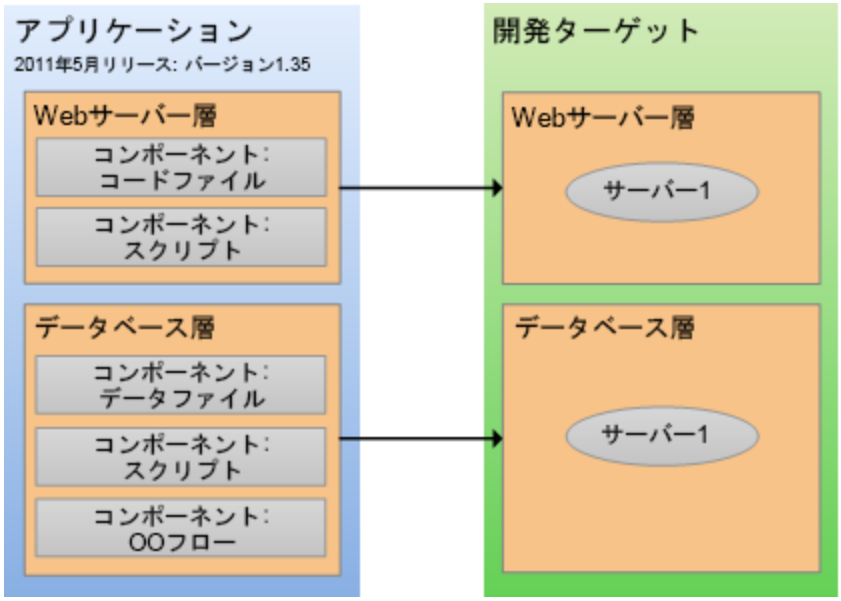
アプリケーションは、管理対象サーバーのセットにデプロイされます。このサーバーのセットを**ターゲット**と呼びます。各ターゲットは、その構造に応じて、通常は1つ以上のアプリケーションをホストするのに適しています。

ターゲットは、QAや運用などの特定の**環境**に属します。アプリケーションデプロイメントのユーザーインタフェースで使用される環境とターゲットは、メインのHPE Server Automationユーザーインタフェースのデバイスグループに反映されます。

ターゲットは**階層**によって構造化されています。階層は、アプリケーションの特定の部分に必要な機能(通常はミドルウェア)を提供します。階層の例としては、Webサーバー、アプリケーションサーバー、データベースなどが挙げられます。アプリケーションをデプロイする場合、アプリケーション階層のすべてのコンポーネントが、各サーバーの対応するターゲット階層にデプロイされます。

SAのソフトウェアポリシーを階層に関連付けることで、適切なミドルウェアがその階層にすでに存在するか、アプリケーションデプロイメントによって管理されるアプリケーションとともにデプロイされることを保証できます。

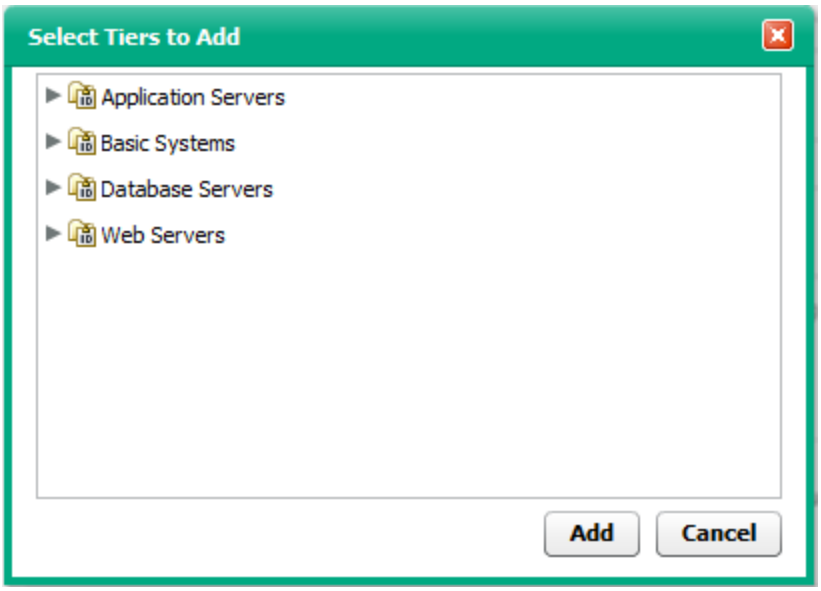
2つの階層を含むターゲット



ターゲットを作成する際に特定のアプリケーションを想定している場合は、そのアプリケーションの階層を使用して、ターゲットの構造を定義することができます。このためには、アプリケーションの特定のリリースのデプロイ可能なバージョンがすでに存在する必要があります。詳細については、「アプリケーションのデプロイ」(539ページ)を参照してください。

新しいターゲットを作成し、既存の階層を使用して構造化することもできます。次の階層は、SAに標準で付属しています。

付属の階層



これらの階層を使用してターゲットを構造化することも、独自の階層を定義することもできます(「アプリケーションデプロイメントの管理」(600ページ)を参照)。

各ターゲットは、QAや運用などの特定の**環境**に関連付けられます。環境は、ソフトウェア開発の**ライフサイクル**に対応付けられます。

自分が編集アクセス権を持っている任意の環境で、ターゲットを構築することができます(「アクセス権の設定」(586ページ)を参照)。

すべての環境を通じてターゲットに一貫した構造を与えることで、テスト環境が運用環境を正確に反映することを保証できます。

ある環境内のターゲットに含まれるサーバーの数は、同じアプリケーションが別の環境にデプロイされる際のターゲットに含まれるアイテムの数とは異なるのが普通です。

たとえば、単純なQA環境では、アプリケーションサーバーとデータベースの実行に、同じ物理(または仮想)マシンが使用される可能性があります。これに対して、運用環境では、これらの機能は複数のマシンに分配されるのが普通です。たとえば、専用のデータベースサーバーと、アプリケーションサーバーのクラスターが存在する可能性があります。

ターゲットは、1つのアプリケーションデプロイメント環境内で一意の名前を持つ必要があります。

前提条件

ターゲットを作成または変更するためには、次の条件が満たされている必要があります。

- このターゲットに必要な階層が使用可能であること。
- このターゲットの階層を満たすのに必要なサーバーがSAによって管理されていること。
- この環境でターゲットを作成するアクセス権がユーザーにあること。
- 追加するサーバーを表示するアクセス権がユーザーにあること。

SAのアクセス権の詳細については、『SA 10.50管理ガイド』を参照してください。

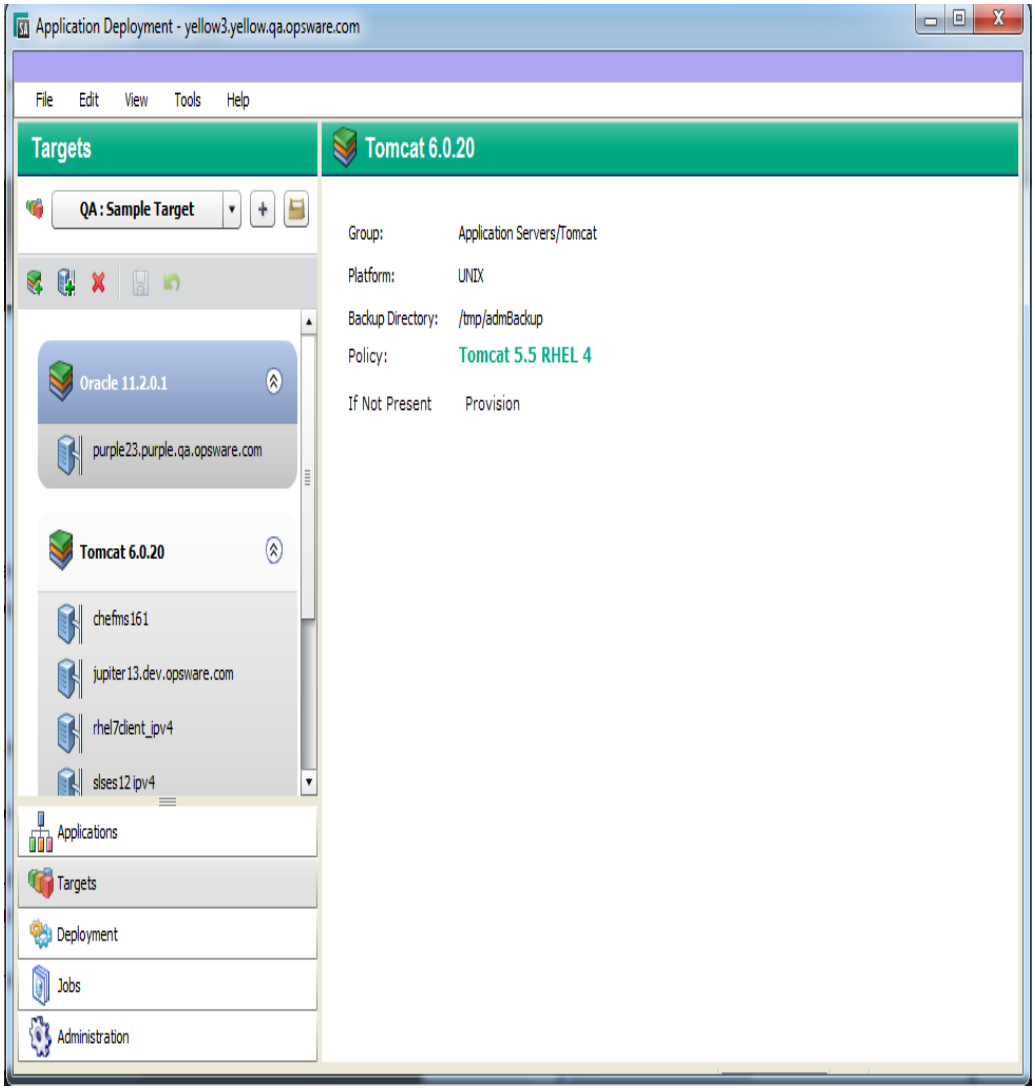
[Targets] 画面について

ターゲットを選択する前には、[Targets] 画面は「クイックスタート」(440ページ)の図のようになっているはずです。次の図はターゲット画面を示します。ターゲットを選択すると、次のようになります。

注意すべき点:

- 左上のドロップダウンリストには既存のすべてのターゲットが表示されます。この例では、Production: Web Storeターゲットが選択されています。
- 左側のペインには、ターゲットの構造が示されます。
- 右側のペインには、左側のペインで選択されているアイテムに関する情報が表示されます。1つのサーバーを選択することも、階層全体を選択することもできます。アイテムを選択すると、そのアイテムの背景色が変化します。この例では、Oracle 11.2.0.1階層が選択されています。
- 右側のパネルに表示される青いテキストは、さらに詳細な情報へのリンクになっています。この例では、この階層に関連付けられているポリシーへのリンクです。左側のパネルでサーバーを選択した場合、そのサーバーに関する詳細情報へのリンクが表示されます。

既存のターゲットを示すターゲット画面



ターゲットの操作

以下のトピックでは、ターゲットの作成と管理に関する基本的手順を説明します。

- 「新しいターゲットの作成」(525ページ)
- 「ターゲットの管理」(527ページ)

さらに詳細な情報については、各トピックに記されているリンク先を参照してください。

新しいターゲットの作成

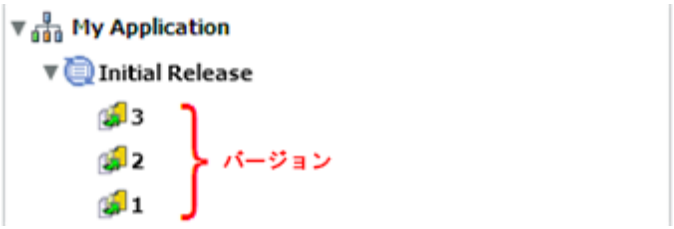
このトピックでは、新しいターゲットを作成するための基本的な手順を説明します。ここでは、アプリケーションデプロイメントターゲットの作成に必要なSAアクセス権と、ターゲットが置かれる環境を編集するためのアクセス権をユーザーが持っていることを前提としています。前提条件を参照してください。

新しいターゲットを作成して構成するには、次の手順を実行します。

1. [Targets] 画面に移動します (左下で [**Targets**] をクリックします)。
2. [**Create a New Target**] をクリックします。[Create New Target] ダイアログが開きます。
3. ターゲット名を入力します。名前は、環境内で一意である必要があります。
4. [Location] ドロップダウンフィールドで、このターゲットが置かれる環境を選択します。

   ターゲットに階層を設定するには、2つの方法があります。手動で追加する方法と、自動的に作成する方法です。

   既存のアプリケーションで使用されているのと同じ階層のセットをターゲットに自動的に作成するには、[Use Application's Tiers] を選択し、アプリケーションのバージョンを選択します。



[Select Version] ドロップダウンリストに表示されるためには、アプリケーションの特定のリリースの特定のバージョンがあらかじめ作成されている必要があります。

5. [**OK**] をクリックします。
6. このターゲットに階層を追加するには、次の手順を実行します。
   a. [**Add Tier**] をクリックします。
   b. 追加する階層のタイプを選択します。アプリケーションで使用されるのと同じ階層のセット (またはスーパーセット) を作成する必要があります。

      ソフトウェアには、標準の階層のグループが付属しています。必要な階層が利用できない場合は、アプリケーションデプロイメント管理者に追加を依頼してください。「階層の管理」(601ページ)を参照してください。
   c. [**Add**] をクリックします。
   d. アプリケーションに必要な各階層に関して、同じ手順を繰り返します。
7. ステップ6で追加した階層のそれぞれに対して、次の手順を実行して、この階層に含まれるサーバーを指定します。
   a. 対象とする階層を選択します。
   b. 階層にサーバーを追加します。
8. ステップ7を繰り返して、残りのすべての階層にサーバーを追加します。
9. [**Save Target**] をクリックして変更内容を保存します。

サーバーの追加

1. [**Add Server**] をクリックします。

   使用可能なサーバーのリストは、プラットフォームによってフィルターされます。階層に対して適切なサーバーを選択してください。例: 階層のコンポーネントがRedHat EL 5を対象としている場合、RedHat EL 5サーバーのみを選択します。
2. [Add Servers to Target] ダイアログで、追加する管理対象サーバーを見つけて選択します。CTRL キーを使用して複数のサーバーを選択できます。

   未プロビジョニングサーバーを追加する場合は、「デプロイメント時のサーバーのプロビジョニング」(534ページ)を参照してください。
3. [**OK**] をクリックします。

ターゲットの管理

適切なアクセス権があるユーザーは(「概要」(586ページ)を参照)、Manage Targetsツールを使用して、いくつかの作業を実行できます。このツールを使用すると、次の作業を実行できます。

- 「新しいターゲットグループの作成」(528ページ)
- 「ターゲットまたはターゲットグループの削除」(528ページ)
- 「ターゲットまたはターゲットグループの名前変更」(529ページ)

Manage Targetsツールを開くには、[Targets] 画面で [**Manage targets**] をクリックします。ターゲットを表示するアクセス権がない場合は、このボタンは使用できません。

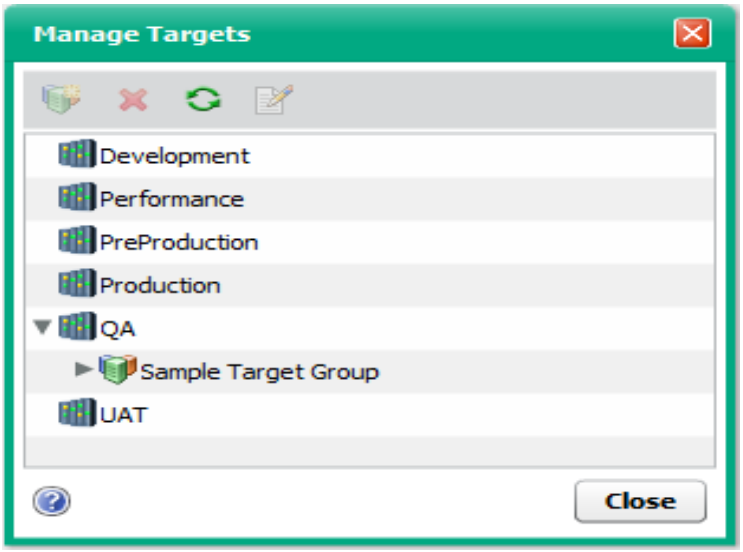
Manage Targetsツールには、次のアイテムを表示したテーブルが含まれます。

環境

ターゲットグループ(ネスト可能)

ターゲット

Manage Targetsツール



テーブルのアイテムのうち、「子」が存在するものには、前に矢印が付いています。▶矢印をクリックすると、アイテムが展開されます。▼矢印をクリックすると、アイテムが折りたたまれます。表示を更新してすべてのアイテムを折りたたむには、[Refresh] ボタンをクリックします。

特定のアイテムを編集するには、そのアイテムを含む行を選択して、ツールバーの [Edit Properties]  ボタンをクリックするか、単にアイテムをダブルクリックします。ただし、Manage Targetsツールから変更できるのは、ターゲットまたはターゲットグループの名前だけです。ターゲットの構造または内容を変更するには、[Targets] 画面に戻ります。

新しいターゲットグループの作成

ターゲットグループとは、ターゲットを階層構造で整理するためのコンテナーです。適切なアクセス権があるユーザーは (「概要」(586ページ)を参照)、ターゲットグループを作成することができます。

ターゲットグループを作成するには、次の手順を実行します。

1. [Targets] 画面に移動します (左下で [**Targets**] をクリックします)。
2. Manage Targetsツールを開きます (「ターゲットの管理」(527ページ)を参照)。
3. ターゲットグループを作成する環境を展開します。
4. トップレベルにターゲットグループを作成する場合は、環境を選択します。ネストしたターゲットグループを作成するには、新しいグループを中に作成する既存のグループを選択します。
5. [**Create Target Group**] をクリックします。
6. 新しいグループの名前を指定します。名前は、階層内のこのコンテナー内で一意である必要があります。
7. [**OK**] をクリックします。新しいグループがテーブルに表示されます。グループとターゲットは、階層のレベルごとにアルファベット順で表示されます。

ターゲットおよびターゲットグループを、同じ環境内の別の場所にドラッグアンドドロップすることができます。ただし、別の環境に移動することはできません。

ターゲットまたはターゲットグループの削除

適切なアクセス権があるユーザーは (「概要」(586ページ)を参照)、Manage Targetsツールを使用して、ターゲットまたは空のターゲットグループを削除することができます。

既存のターゲットまたはターゲットグループを削除するには、次の手順を実行します。

1. Manage Targetsツールを開きます (「ターゲットの管理」(527ページ)を参照)。
2. テーブルで、削除するアイテムを含む行を選択します。次の点に注意してください。
   - 複数のアイテムを選択するには、CtrlまたはShiftキーを使用します。ただし、複数のアイテムを削除する場合は、同じタイプでなければなりません。たとえば、複数のターゲットまたは複数のターゲットグループです。
   - ターゲットを削除する場合、スケジュール済みで実行されていないジョブも削除されます。
   - 削除できるターゲットグループは空のものだけです。
3. [**Delete**] をクリックします。

アイテムを削除してよいかどうかの確認が表示されます。[**Yes**] をクリックして削除します。

ターゲットまたはターゲットグループの名前変更

適切なアクセス権があるユーザーは (「概要」(586ページ)を参照)、Manage Targetsツールを使用して、ターゲットまたはターゲットグループの名前を変更することができます。

ターゲットまたはグループの名前を変更するには、次の手順を実行します。

1. Manage Targetsツールを開きます (「ターゲットの管理」(527ページ)を参照)。
2. テーブルで、名前を変更するターゲットまたはグループを含む行を選択します。
3. [**Edit Properties**] をクリックします。
4. ターゲットまたはグループの新しい名前を入力します。ターゲット名は、環境内で一意である必要があります。「命名規則」(438ページ)も参照してください。
5. [**OK**] をクリックします。

既存のターゲットへの階層の追加

このトピックでは、既存のターゲットに階層を追加するための基本的な手順を説明します。ここでは、アプリケーションデプロイメントターゲットの作成に必要なSAアクセス権と、ターゲットが存在する環境を編集するためのアクセス権をユーザーが持っていることを前提としています。詳細については、前提条件を参照してください。

既存のターゲットに階層を追加するには、次の手順を実行します。

1. [Targets] 画面に移動します (左下で [**Targets**] をクリックします)。
2. [Select Target] ドロップダウンリストで、階層を追加するターゲットを選択します。
3. [**Add Tier**] をクリックします。
4. 追加する階層のタイプを選択します。アプリケーションで使用されるのと同じ階層のセット (またはスーパーセット) を作成する必要があります。

   ソフトウェアには、標準の階層のグループが付属しています。必要な階層が利用できない場合は、アプリケーションデプロイメント管理者に追加を依頼してください。「階層の管理」(601ページ)を参照してください。
5. [**Add**] ボタンをクリックします。
6. 新しい階層に1つ以上のサーバーを追加します (「既存の階層へのサーバーの追加」(530ページ)を参照)。
7. [**Save Target**] をクリックして変更内容を保存します。

既存のターゲットからの階層の削除

このトピックでは、既存のターゲットから階層を削除するための基本的な手順を説明します。ここでは、アプリケーションデプロイメントターゲットの編集に必要なSAアクセス権と、ターゲットが存在する環境を編集するためのアクセス権をユーザーが持っていることを前提としています。前提条件を参照してください。

既存のターゲットから階層を削除するには、次の手順を実行します。

1. [Targets] 画面に移動します (左下で [**Targets**] をクリックします)。
2. [Select Target] ドロップダウンリストで、階層を削除するターゲットを選択します。
3. 削除する階層を選択します。
4. [**Delete Selected Item**] をクリックします。
5. [**Yes**] をクリックします。
6. [**Save Target**] をクリックして変更内容を保存します。

既存の階層へのサーバーの追加

このトピックでは、既存の階層にサーバーを追加するための基本的な手順を説明します。ここでは、アプリケーションデプロイメントターゲットの編集に必要なSAアクセス権と、関連するターゲットが存在する環境

を編集するためのアクセス権をユーザーが持っていることを前提としています。前提条件を参照してください。

既存のターゲットから階層を削除するには、次の手順を実行します。

1. [Targets] 画面に移動します (左下で [**Targets**] をクリックします)。
2. [Select Target] ドロップダウンリストで、操作するターゲットを選択します。
3. サーバーを追加する階層を選択します。
4. [**Save Target**] をクリックして変更内容を保存します。

階層からのサーバーの削除

このトピックでは、既存の階層からサーバーを削除するための基本的な手順を説明します。ここでは、アプリケーションデプロイメントターゲットの編集に必要なSAアクセス権と、ターゲットが存在する環境を編集するためのアクセス権をユーザーが持っていることを前提としています。前提条件を参照してください。

既存の階層からサーバーを削除するには、次の手順を実行します。

1. [Targets] 画面に移動します (左下で [**Targets**] をクリックします)。
2. [Select Target] ドロップダウンリストで、操作するターゲットを選択します。
3. 削除するサーバーを選択します。
4. [**Delete Selected Item**] をクリックします。
5. [**Yes**] をクリックします。
6. [**Save Target**] をクリックして変更内容を保存します。

このプロセスでは、サーバーがターゲットから削除されるだけです。サーバー自体は変更されません。

ジャストインタイムターゲット

アプリケーションデプロイメントターゲットを作成するには、次の2つの方法があります。

- デプロイメントの前に、[Targets] 画面のツールを使用して作成できます (このセクションですでに説明)。
- アプリケーションをデプロイする際に作成できます。これを「ジャストインタイム」ターゲット作成と呼びます。

ジャストインタイムターゲットは、ターゲットリソースの割り当てや選択を委任したい場合に便利です。ジャストインタイムターゲットを使用すれば、アプリケーションのデプロイ先を気にせずに、アプリケーションの定義に集中することができます。ジャストインタムターゲットとHPE Operations Orchestration (HPE OO) フローを使用することで、リソースの決定を外部化し、サードパーティ製の割り当ておよびスケジューリングツールと連携することができます。

アプリケーションデプロイメントは、指定したHPE OOフローを使用して、ジャストインタイムターゲットを完成します。ほとんどの場合、このフローは、既存の管理対象サーバーのセットを選択して、新しいターゲットに割り当てます。それ以外の場合、フローには、サーバーを選択し、そのサーバーに対して必要なアクションを実行し、SAに引き渡して、有効な管理対象サーバーにするために必要な、すべてのロジックが含まれます。

ジャストインタイムターゲットを作成した後、「ターゲットの管理」(527ページ)に記されている手順で、そのターゲットを管理することができます。

前提条件

ジャストインタイムターゲットを作成する前に、次の作業が必要です。

1. SAでHPE OOとの統合を設定します。
2. 次のパラメーターを受け取るHPE OOフローを定義します。

パラメーター	説明
host	SAコアサーバーのホスト名またはIPアドレス。
username	新しいターゲットを作成するSAユーザーの名前 (これは、サーバーの割り当てをこのSAユーザーのアクセス権によって制限するために必要です)。
environment	新しいターゲットを含むアプリケーションデプロイメント環境のID。
target	フローがサーバーを追加する新しいターゲットのID。
tierCounts	階層とサーバー数のパラメーターを含む、区切り文字入りの文字列。 形式は`tier,count;tier,count...`です。 `tier`は、フローがサーバーを追加する階層のIDです。 `count`は、新しい (複製または新規作成された) ターゲットのサーバー数です。

3. このHPE OOフローを、アプリケーションデプロイメントユーザーインタフェースの [Administration] 画面で、ジャストインタイムターゲットフローとして指定します。

ジャストインタイムターゲットを作成するには、ターゲットが存在する環境に対する編集アクセス権が必要です。

ジャストインタイムターゲットの作成

ジャストインタイムターゲットを作成するには、次の手順を実行します。

1. 左下で [**Deployment**] ボタンを選択します。
2. デプロイするアプリケーションのバージョンを選択します。詳細については、「特定バージョンのデプロイ」(550ページ)を参照してください。
3. [Select Target] ドロップダウンリストの右側の [**Advanced Target Selector**] をクリックします。[Advanced Target Selection] ダイアログが開きます。
4. [**Just-In-Time Target**] を選択します。
5. [**Continue**] をクリックします。[Create New Just-In-Time Target] ダイアログが開きます。このダイアログで、次の情報を指定します。
   a. Name – ターゲットの名前を入力します。
   b. Location – ターゲットが存在する環境またはターゲットグループを選択します。
   c. Based On
      - [Version] を選択すると、デプロイするバージョンに含まれるのと同じ階層を使用してターゲットが作成されます。
      - この場合、バージョンに基づいて階層を選択することにより、真の「オンデマンド」ターゲットが新規作成されます。
      - [Target] を選択すると、既存のターゲットをコピーすることでターゲットが作成されます。

        この場合、別のターゲットの階層構造をテンプレートに使用して、複製されたターゲットが作成されます。これは、場合によってリソースを柔軟に割り当てるために便利です。たとえば、オンラインストアの場合、休暇のショッピングシーズンには追加の容量を割り当てることができます。
   d. 前のステップで [Target] を選択した場合、既存のターゲットをドロップダウンリストから選択します。
   e. テーブルで、各階層に含められるサーバーの数を指定します ([New Size] 列の値をダブルクリックします)。
6. [**Create**] をクリックします。アプリケーションデプロイメントは、新しいターゲットを作成し、ジャストインタイムターゲットフローを呼び出して、ターゲットにサーバーを設定します。このフローは、

[Administration] ページで指定されます (「アプリケーション設定の管理」(616ページ)を参照)。

アプリケーションデプロイメントがターゲットを作成できない場合、エラーメッセージが表示されます。

7. [**Close**] をクリックして、[Deployment] 画面に戻ります。

   新しいターゲットが自動的に選択されています。[**Start Job**] をクリックして、デプロイメントを開始します。

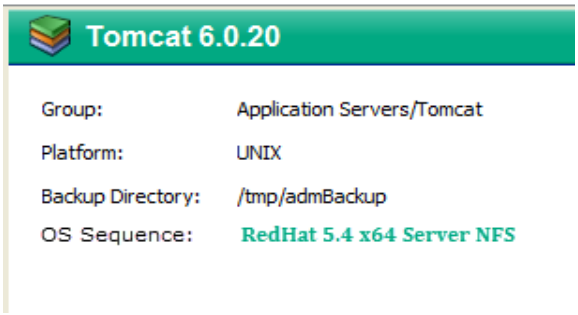
デプロイメント時のサーバーのプロビジョニング

アプリケーションデプロイメントでは、OSシーケンスを使用することで、プロビジョニングされていないサーバーにデプロイメント時にオペレーティングシステム (OS) をインストールすることができます。

OSシーケンスは、未プロビジョニングサーバーにインストールする内容 (インストールプロファイルのOSビルド情報、選択されているソフトウェアポリシーなど) を定義します。

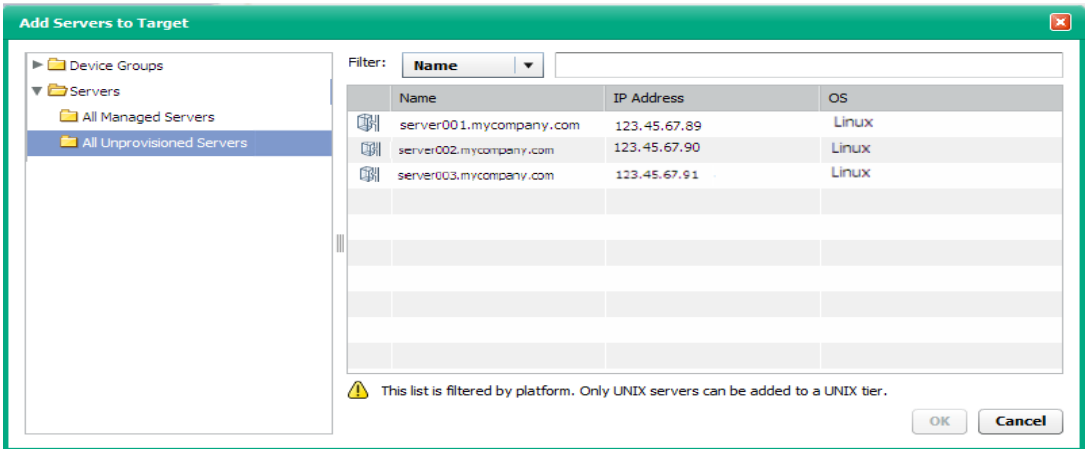
デプロイメント時にサーバーをプロビジョニングするには、3つの基本ステップがあります。

1. アプリケーションデプロイメント管理者は、OSシーケンス (および必要なミドルウェア) を特定の階層に関連付けます。

   詳細については、「階層の管理」(601ページ)を参照してください。

2. ターゲットを定義する際に、未プロビジョニングサーバープールから、1つ以上のサーバーを、ターゲットのこの階層に追加します。

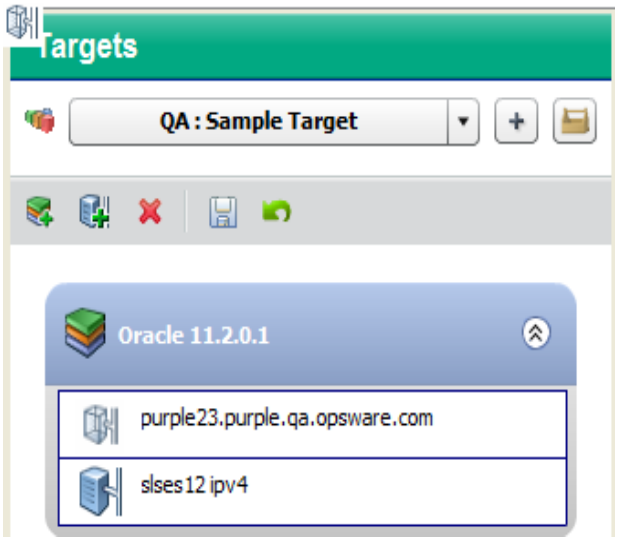
3. 

すべてのAll Unprovisioned Serversフォルダーを表示するには、SAコアに対するサーバープールアクセス権が必要です。このアクセス権は、SA管理者によってユーザーに割り当てられている必要があります(『SA 10.50管理ガイド』の「アクセス権のリファレンス」のセクションを参照)。

未プロビジョニングサーバーを見つけるには、Device Groupsフォルダーを参照する方法もあります。

- SAミニエージェントとして起動する未プロビジョニングサーバーには、白抜き アイコンが表示されます。
- 管理対象サーバーには、塗りつぶし アイコンが表示されます。

これらのアイコンは、[Add Servers to Target] ダイアログ(上の図)と、[Targets] 画面と [Deployment] 画面の両方の左側のペインに表示されます。



4. アプリケーションのバージョンをターゲットにデプロイする際に、アプリケーションデプロイメントはターゲット内の1つ以上のサーバーが未プロビジョニングであることを認識します。ステージング時に、指定したOSが、OSシーケンスに指定された情報に従って、これらのサーバーにデプロイされます。

前提条件

アプリケーションデプロイメントが未プロビジョニングサーバーにオペレーティングシステムをデプロイするためには、次の条件が満たされる必要があります。

- サーバーレコードが、SAの利用可能ライフサイクルと到達不能SA状態にある必要があります。これは、非OGFS PXEまたは非OGFS CDブートによって行うことができます。

  詳細については、『SA 10.50ユーザーガイド』の「エージェント管理」を参照してください。
- アプリケーションデプロイメントユーザーインタフェース内のAll Unprovisioned Serversフォルダーを表示するには、SAコアに対するサーバープールアクセス権が必要です。
- SAコア上のライブラリには、サーバーの適切なプロビジョニングに使用できるOSシーケンスが存在する必要があります。

  OSシーケンスを階層に関連付ける責任を負っているユーザー(通常はアプリケーションデプロイメント管理者)は、次のSAアクセス権を持つ必要があります。
  - OSシーケンス機能の管理のアクセス権
  - OSシーケンスが保存されているフォルダーに対するフォルダー読み取りアクセス権

  これらのアクセス権は、1つのSAユーザーグループから得られることが必要です。

  ただし、デプロイメント時にサーバーをプロビジョニングするには、これらのアクセス権は不要です。
- このサーバーが属する階層には、OSシーケンスが関連付けられている必要があります。

OGFS PXEまたはOGFS CDブートで起動するサーバーは、このプロセスではプロビジョニングできません(現在のSAでの制限)。詳細については、SAオンラインヘルプの「OSプロビジョニング」を参照してください。

プロビジョニングの仕組み

アプリケーションのバージョンをデプロイすると、アプリケーションデプロイメントはステージング時に次の手順を実行します。

1. 環境に関連付けられたデプロイメント前フロー(存在する場合)を実行します。
2. 指定されたターゲットで階層に関連付けられたOSシーケンス(存在する場合)を実行します。これにより、ターゲット内の未プロビジョニングサーバーに、指定されたOSがインストールされ、設定されます。
3. 指定されたターゲット内の階層に関連付けられたポリシーをアタッチし、修復します。

その後、アプリケーションデプロイメントはバージョンをデプロイします。

未プロビジョニングサーバーは、通常の方法で定義されたターゲットと、ジャストインタイムターゲットの両方に含めることができます。

次の点に注意してください。

プラットフォームの一致

階層に追加できるのは、正しいビルドエージェントタイプの未プロビジョニングサーバーだけです。たとえば、Linux OSシーケンスの階層に追加できるのは、Linuxビルドエージェントを持つサーバーだけです。同様に、Solarisをインストールできるのは Solarisビルドエージェントだけです。

OSシーケンスの存在

OSシーケンスを持つ階層に追加できるのは、未プロビジョニングサーバーだけです。未プロビジョニングサーバーを階層に追加した後で、その階層からOSシーケンスを削除した場合、アプリケーションデプロイメントはデプロイメントジョブを作成できず、エラーメッセージが表示されます。

競合

場合によっては、OSシーケンスの競合が発生することがあります。1つのサーバーが2つの異なる階層に含まれており、それらの階層に異なるOSシーケンスが存在する場合、アプリケーションデプロイメントはどのOSをサーバーにインストールすればよいかわかりません。この場合、アプリケーションデプロイメントはデプロイメントジョブを作成できず、エラーメッセージが表示されます。

ロールバックとアンデプロイ

ロールバックとアンデプロイは、OSシーケンスには適用されません。ジョブをロールバックまたはアンデプロイした場合、そのジョブをデプロイした時点で未プロビジョニングで、その後にOSシーケンスを使用してプロビジョニングされたサーバーは、そのまま管理対象サーバーとなります。サーバーを未プロビジョニングプールに戻すには、明示的に非アクティブ化する必要があります。

再プロビジョニングなし

デプロイ先のターゲットの階層にプロビジョニング済みと未プロビジョニングの両方のサーバーがあり、その階層にOSシーケンスが関連付けられている場合、OSシーケンスは未プロビジョニングサーバーだけに適用されます。すなわち、このプロセスでサーバーの再プロビジョニングを行うことはできません。

サーバーの名前変更

サーバーの設定によっては、OSシーケンスを使用してプロビジョニングを行ったときに、名前が変更されることがあります。たとえば、通常の場合、ベアメタルマシンには一般的な名前が付けられており、OSシーケンスによってもっと意味のある名前が割り当てられることがあります。同様に、管理対象だったサーバーを非アクティブ化して、未プロビジョニングプールに戻した場合、OSシーケンスを使用してもう一度プロビジョニングすると、名前が変わる可能性があります。

名前の変更はすべてジョブログに記録されますが、アプリケーションデプロイメントの[Jobs]画面には、最初のうち正しいサーバー名が表示されない可能性があります。[Jobs]画面を閉じてから開き直すと、新しい名前が表示されます。

プロビジョニングの失敗

OSシーケンスのインストールが何らかの理由で失敗した場合、デプロイメントジョブは失敗し、サーバーの状態はプロビジョニング失敗に設定されます。この場合、まずサーバーを非アクティブ化 (または削除)してから、そのサーバーにSAエージェントを再インストールする必要があります。手順については、『SA 10.50 ユーザーガイド』の「エージェント管理」を参照してください。

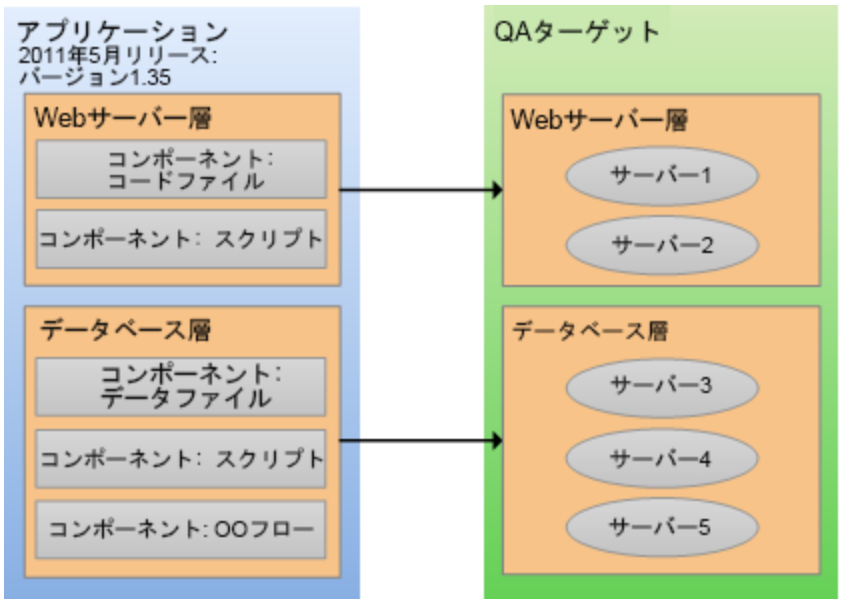
アプリケーションのデプロイ

単純な例については、「クイックスタート」(440ページ)を参照してください。

概要

アプリケーションデプロイメントでは、ユーザーは、アプリケーションの特定の**バージョン**を、**ターゲット**にデプロイします。アプリケーションのコンポーネントは、指定されたターゲットの関連する階層内のすべてのサーバーにデプロイされます。コンポーネントは、[Deployment Order] ビューに表示されている順序でデプロイされます (「コンポーネント」(469ページ)を参照)。ローリングデプロイメントが設定されている場合を除き、すべてのサーバーが完了した後で、次のコンポーネントがデプロイされます。

ターゲットへのアプリケーションのデプロイ



コンポーネントのデプロイメントには次の4つのフェーズがあります。

1. 必要なファイルはすべて、SAコアからターゲットサーバー上の一時的な場所にコピーされます。これを**ステージング**と呼びます。ソフトウェアポリシーコンポーネントと、ポリシーデプロイ方法を使用するコードおよびパッケージコンポーネントの場合、ステージングの際にポリシーがアタッチされます。
2. アプリケーションデプロイメントは、指定された**カットオーバー**時間まで待ちます。

3. コンポーネントでロールバックが有効にされている場合、ターゲットサーバー上の関連するファイルとディレクトリは、階層の定義時に指定された場所にバックアップされます(「アプリケーションデプロイメントの管理」(600ページ)を参照)。
4. コンポーネントのデプロイに必要なアクションが実行されます。このアクションの性質は、コンポーネントのタイプによって異なります。

- コードコンポーネントの場合:
  - デプロイ方法がパッケージの場合、ステージングされたZIPファイルからファイルが抽出され、ターゲットサーバー上の指定された場所に置かれます。
  - デプロイ方法がポリシーの場合、ステージング中にアタッチされたポリシーが修復されます。
- スクリプトコンポーネントの場合、指定されたスクリプトがターゲットサーバー上で実行されます。
- 構成ファイルコンポーネントの場合、ターゲットサーバー上にファイルが作成されます。
- ソフトウェアポリシーコンポーネントの場合、ステージング中にアタッチされたポリシーが修復されます。
- パッケージコンポーネントの場合、次のいずれかが行われます。
  - デプロイ方法がパッケージの場合、ターゲットサーバーにパッケージがインストールされます。
  - デプロイ方法がポリシーの場合、ステージング中にアタッチされたポリシーが修復されます。
- アプリケーション構成コンポーネントの場合、アプリケーション構成がターゲットサーバーにデプロイされます。
- OOフローコンポーネントの場合、HPE Operations Orchestrationサーバー上でフローが完了まで実行されます。
- Windowsレジストリコンポーネントの場合、関連するレジストリキーまたは値が追加または削除されます。

アプリケーションデプロイメントは、指定されたターゲット内の各サーバーに対して、選択されたバージョンに含まれる各コンポーネントを対象にこれらのステップを実行するデプロイメントジョブを作成します。ジョブの進行状況とステータスは、[Jobs] 画面で見ることができます。

どれかのコンポーネントのデプロイが失敗するか、デプロイメントジョブがキャンセルされた場合、デプロイメントは失敗の時点から自動的に**ロールバック**されます。ロールバックが有効になっている各コンポーネントに対して、コンポーネントに指定されているロールバックアクションが実行され、一時ディレクトリに保存されているステージングファイルが削除されます。

また、[Jobs] 画面からデプロイメントを手動でロールバックすることもできます。ロールバックによって、サーバーはそのデプロイメントが行われる前の状態に戻ります。

ロールバックに代わる方法として、**アンデプロイ**があります。アンデプロイプロセスは、バージョンをアンインストールし、サーバー上からその痕跡を削除します。これはロールバックよりも強力なプロセスです。サーバーを以前の状態に戻すことを目的にしてはいません。

バージョンをデプロイする際に、前のバージョンを自動的にアンデプロイするようにアプリケーションデプロイメントに指示することができます。この場合、2つのジョブが作成されます。1つは前のバージョンのアンデプロイジョブで、もう1つは現在のバージョンのデプロイジョブです。

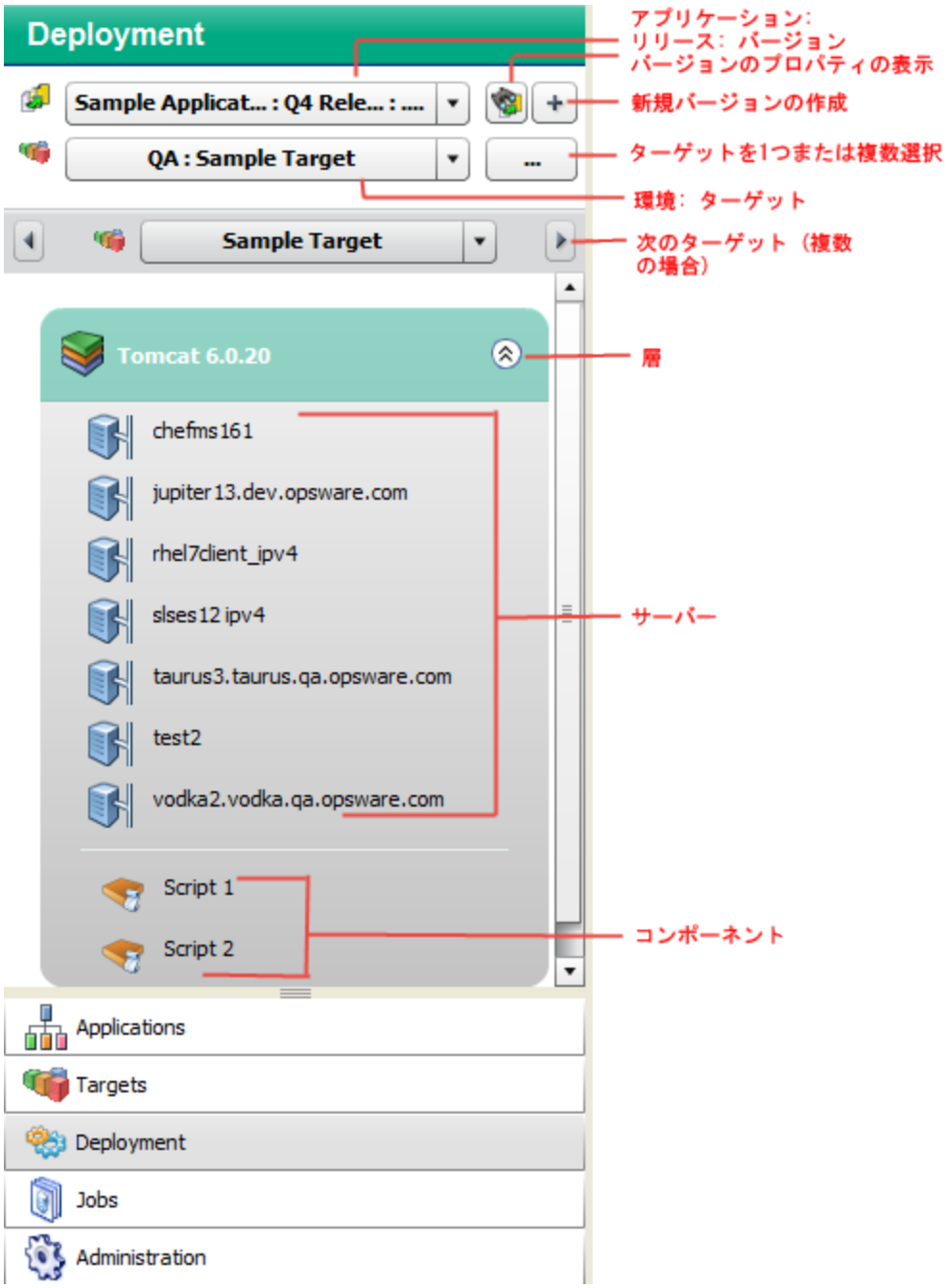
アプリケーションデプロイメントは、**ローリングデプロイメント**をサポートしています。これは、特定の階層内のコンポーネントを、その階層内のターゲットサーバーの一部だけにデプロイし、階層内の残りのサーバー上では、アプリケーションをそのまま動作させておく方法です。

- デプロイメント時にコンポーネントに使用されるパラメーターはカスタマイズできます。これは、特定のターゲットまたは環境に対してデフォルト値をオーバーライドする場合や、単に特定のデプロイメントで別の値を使用する場合に便利です。
- SAバージョン9.01以降、SAは「ジャストインタイム」ターゲットをサポートします。これは、ユーザーが指定したHPE OOフローを使用して、デプロイメント時に作成されるターゲットです。詳細については、「ターゲット」(521ページ)を参照してください。

[Deployment] 画面

[Deployment] 画面では、リリースの特定のバージョンを、1つ以上のターゲットにデプロイできます。[Deployment] 画面の左側のパネルは、構造ビューを表します。

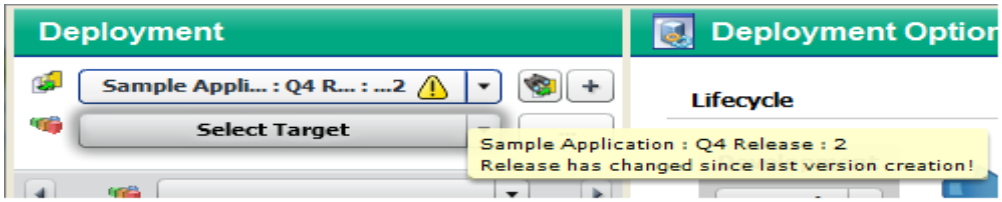
[Deployment] 画面の左側のパネル



この例では、1つのターゲットが選択されており(Advantage123)、各階層に1つのサーバーがあります。デプロイメントの対象として複数のサーバーを選択できますが、すべてのターゲットは同じ環境に存在する必要があります。

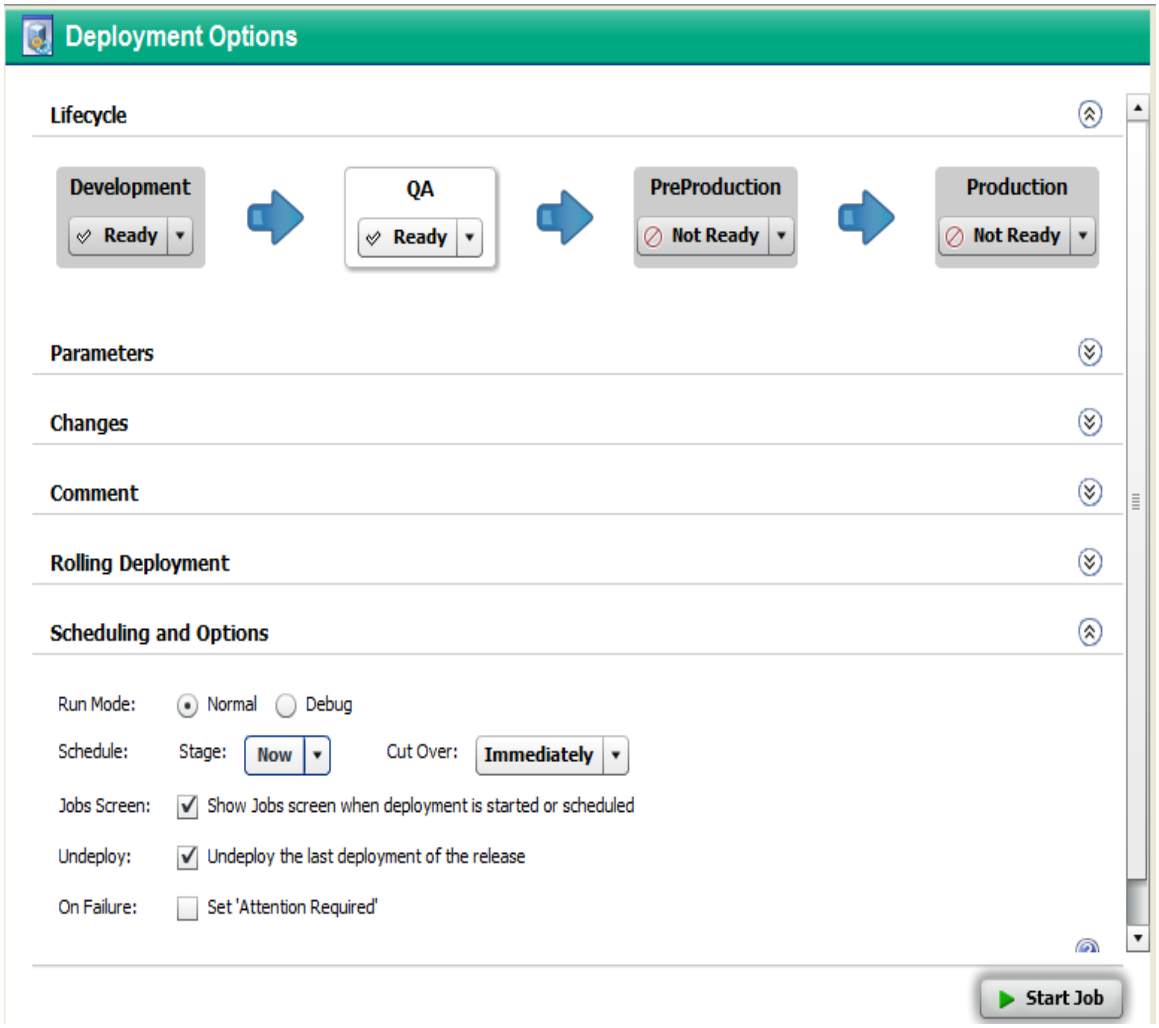
この例で「警告」記号⚠が表示されているのは、RC 3バージョンの作成後にGreen Checking 1.0リリースが変更されているからです。このアイコンが表示された場合、その上にマウスポインターを置くと、詳細情報が表示されます。

警告記号に関連するツールヒント



[Deployment] 画面の右側のパネルには、内容ビューが表示されます。これはこのデプロイメントジョブに固有の設定です。

[Deployment] 画面の右側のパネル



右側のパネルの次のセクションには、[**Start Job**] ボタンをクリックしたときに作成されるデプロイメントジョブに関する情報が表示されます。

セクションを展開するには、  アイコンをクリックします (または、青いバーをダブルクリックします)。

Lifecycle

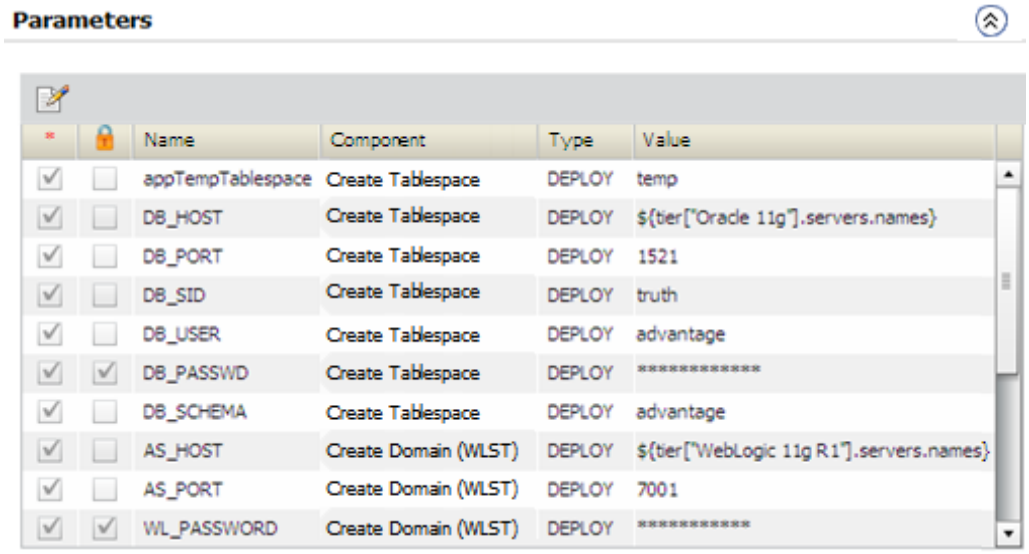
パネルの一番上の [Lifecycle] セクションには、このライフサイクルに含まれるすべての環境のステータスが表示されます。デプロイ先の環境が強調表示されます。

デプロイ先の環境は、「Not Ready」とマークされていないものに限ります。使用可能な環境は、アプリケーションデプロイメント管理者または環境の所有者が決定します。この環境に対するデプロイアクセス権を持つユーザーは、デプロイメント時に環境のステータスを更新できます。

Parameters

[Parameters] セクションでは、アプリケーションのコンポーネントで使用される編集可能なパラメーターの値を表示してカスタマイズすることができます。デプロイメント時に、アプリケーションデプロイメントは、必要なすべてのパラメーターに値があることを確認します。値がないものがある場合は、デプロイメントジョブは開始されません。

[Deployment Job] 画面のパラメーター情報



この例の場合、パラメーターの値は、アプリケーションのコンポーネントで指定されたものです。これらは「アプリケーションデフォルト」値と呼ばれます。DB_HOSTおよびAS_HOSTパラメーターには「特殊変数」が使用されており、その値は対応する階層のサーバーの名前から得られます。特殊変数は、デプロイメント時 (デプロイ先サーバーが選択されるとき) までわからないデータに対応するプレースホルダーです。

アプリケーションの値は、デプロイメント時に、特定のターゲットまたは環境全体に対してオーバーライドすることができます (「デプロイメントパラメーターのカスタマイズ」(551ページ)を参照)。

Changes

[Changes] セクションには、このリリースに関連するコードコンポーネントの変更状況が示されます。フルリリースの場合、図に示すように、すべてのファイルがリストに表示されます。デルタリリースの場合、リリースの「ベースライン」バージョンがデプロイされた後で追加または変更されたファイルがリストに表示されます。

[Deployment Job] 画面に表示されたコードコンポーネントの変更

Comment

[Comment] セクションでは、ユーザーがテキストコメントを入力することができます。これは、レポートとジョブログに表示されます。

Rolling Deployment

[Rolling Deployment] セクションでは、デプロイメントのために一度にオフラインにする各階層内のサーバーの数を指定します。これは [Number of Servers per Group] 設定で決まります。デフォルトでは、この設定は階層内のサーバーの数です。すなわち、デフォルトでは、すべてのサーバーが同時にデプロイされます。

たとえば、更新中はサーバーを負荷分散プールから外し、その後に再び追加するように、アプリケーションを作成する必要があります。これにより、ローリングバッチの更新中に、大半のサーバーを稼働させておくことができます。

たとえば、階層内に9台のサーバーがある場合、対応するアプリケーション階層の各コンポーネントを一度に3台のサーバーにデプロイし、その後に次の3台のサーバーのグループに進むことができます。この場合、[Number of Servers per Group] を3に設定します。

ローリングデプロイメントを使用する場合、同じ階層内のコンポーネントのグループが、階層内の指定した数のサーバーにデプロイされます。ローリングデプロイメントを使用しない場合、コンポーネントがデプロイされるためには、その前のコンポーネントが階層内のすべてのサーバーに正常にデプロイされている必要があります。

Scheduling and Options

[Scheduling and Options] セクションでは、デプロイメントの実行方法とタイミング、およびデバッグのオプションを指定できます (「アプリケーションデプロイメントジョブの管理」(562ページ)を参照)。

デプロイメントジョブのオプション

オプション	目的
Run Mode	デプロイメントジョブが通常モードとデバッグモードのどちらで実行されるかを決定します。ここで [Debug] を選択した場合、[**Start Job**] ボタンをクリックすると [Debugger] ウィンドウが開きます。
Stage	アプリケーションデプロイメントが、デプロイメントに必要なファイルを、SAからターゲットサーバー上の一時ディレクトリにコピーするタイミングを決定します。 [Run Mode] が [Debug] の場合、このオプションは自動的に [Now] に設定されます。
Cut Over	コンポーネントがターゲットに実際にデプロイされるタイミングを決定します。指定したカットオーバー時間に、関連ファイルおよびディレクトリのバックアップが実行され (ロールバックが有効になっている場合)、その後にコンポーネントがデプロイされます。 カットオーバー時間の指定には注意が必要です。たとえば、ステージングの完了までに20分かかる場合、カットオーバーは少なくとも30分後に設定する必要があります。アプリケーションデプロイメントでは、カットオーバー時間をステージング時間より前に設定することはできません。 [Immediately] を指定した場合、バックアップおよびデプロイメントステップは、ファイルのステージング完了後ただちに開始されます。 [Run Mode] が [Debug] の場合、このオプションは自動的に [Immediately] に設定されます。
[Jobs] 画面	[**Start Job**] ボタンをクリックしたときに、[Jobs] 画面が表示されるかどうかを決定します。 [Show job status when deployment starts] チェックボックスをオンにした場合、[Jobs] 画面が開きます。自分のデプロイメントジョブだけが表示されるように、フィルターが適用されます。デプロイメント実行の各ステップで、ジョブログを観察することができます。 このオプションを選択しない場合、[Deployment] 画面が表示されたままになります。 [Run Mode] が [Debug] の場合、このオプションは自動的にオンになります。
Undeploy	現在のデプロイメントを実行する前に、このバージョンの前のデプロイメントをアンデプロイするかどうかを決定します。

デプロイメントジョブのオプション (続き)

オプション	目的
	[Undeploy the last deployment of this release] チェックボックスがオフの場合、アプリケーションデプロイメントは前のデプロイメントをアンデプロイしません。 [Run Mode] が [Debug] の場合、このオプションは使用できません。
On Failure	ジョブが失敗した場合に、ロールバック前にジョブをAttention Requiredステータスにするかどうかを決定します。 デプロイメントジョブが失敗した場合、次のいずれかが行われます。 このオプションが選択されていない場合、アプリケーションデプロイメントはジョブステータスをFailedに設定し、自動ロールバックを実行します。 このオプションが選択されている場合、ジョブはサスペンド状態 (Attention Required) になり、ユーザーは管理対象サーバーのステータスを確認して、失敗の原因を調べることができます。 失敗に関して十分な情報が集まったら、ジョブを再開することができます。その時点で、アプリケーションデプロイメントはジョブステータスをFailedに設定し、自動ロールバックを開始します。 [Run Mode] が [Debug] の場合、このオプションは自動的にオンになります。

ジョブボタンの有効化

[**Start Job**] ボタンは、バージョンとターゲットの両方が選択された場合に有効になります。このボタンをクリックすると、指定したオプションに基づいてデプロイメントジョブをスケジュールまたは開始できます。

必要なパラメーターに値がないか、バージョンとターゲットが一致しない (たとえば、コンポーネントがRed Hat Linux用で、ターゲットサーバーでSUSE Linuxが動作している) 場合には、デプロイメントジョブは開始されません。

バージョンのデプロイ

アプリケーションをデプロイするには、そのアプリケーションの特定のバージョンをデプロイします。バージョンは、特定のリリースに関連付けられています。バージョンをデプロイするには、次の2つのステップが必要です。

- 「新しいバージョンの作成」(549ページ) (または既存のバージョンの使用)
- 「特定バージョンのデプロイ」(550ページ)

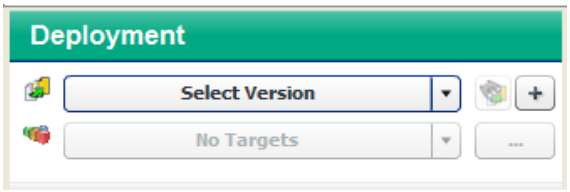
環境に対するデプロイアクセス権があれば、自分がデプロイするアクセス権がある任意のアプリケーションをデプロイすることができます(「概要」(586ページ)を参照)。

新しいバージョンの作成

アプリケーションのデプロイアクセス権がある場合(「概要」(586ページ)を参照)、そのアプリケーションの特定のリリースの新しいバージョンを作成できます。

アプリケーションのバージョンを作成するには、次の手順を実行します。

1. 左下で[**Deployment**]タブを選択します。
2. [Select Version]ドロップダウンフィールドで、対象のアプリケーションに移動します。



3. アプリケーションの適切なリリースの下で、[**Create New Version**]リンクをクリックします。
4. [Create New Version]ウィンドウで、バージョン名(または番号)を入力します。

   ここに指定した文字列がバージョン名になります。ただし、このリリースの前のバージョンに番号が含まれている場合は、アプリケーションデプロイメントが自動的にその番号を進めます。たとえば、前のバージョンが"1"だった場合は、[Version]フィールドに"2"が表示されます。前のバージョンが"V 1.2.1"だった場合は、[Version]フィールドに"V 1.2.2"という文字列が表示されます。

   アプリケーションデプロイメントが[Version]フィールドに入力した内容に関わらず、ユーザーは任意のバージョン名を指定できます。ただし、バージョン名はリリース内で一意である必要があります。
5. オプション: [Description]フィールドに、このバージョンに対して指定する追加情報を入力します。
6. オプション: 前のバージョンの作成後に更新されているコードコンポーネントの一覧を表示するには、[**Show Code Component Changes**]を選択します。
7. [**Create**]をクリックします。これにより、このアプリケーションのコンポーネントであるすべてのファイルが収集され、アプリケーションの新しいバージョンが作成されます。

   そして、バージョンを作成するジョブが開始されます。左下の進行状況バーに、ジョブのステータスと進行状況が表示されます。ジョブが失敗した場合、[Jobs]画面に移動して理由を調べます。
8. [**Close**]を選択します。アプリケーションの新しいバージョンがデプロイ可能になります。

特定バージョンのデプロイ

ユーザーは、アプリケーションの**バージョン**を、1つ以上の**ターゲット**にデプロイします。アプリケーションデプロイメントは、コンポーネントの定義 (アプリケーションの定義より) を、選択したターゲットに指定されたサーバー上で実行される**デプロイメントステップ**に変換します。デプロイメントステップは、次の作業を実行します。

アプリケーションファイルをターゲットサーバーにコピーします。

アプリケーションとともに指定されているスクリプトを実行します。

指定されたSAライブラリからアイテムを適用します。

アプリケーションで参照されているOOフローを実行します。

バージョンをデプロイするには、次の手順を実行します。

1. 左下で [**Deployment**] ボタンを選択します。
2. [Select Version] ドロップダウンリストで、デプロイするアプリケーションのバージョンに移動して選択します。既存のバージョンを選択するか、新しいバージョンを作成します (「新しいバージョンの作成」(549ページ)を参照)。

   リスト内の既存のバージョンの上にマウスポインターを置くと、その階層がリストの右側に表示されます。リリースにアスタリスク (*) がついている場合、前回のバージョンの作成以降に、アプリケーションのこのリリースが変更されていることを示します。
3. [Select Target] ドロップダウンリストで、アプリケーションのこのバージョンをデプロイするターゲットを選択します。

   ターゲットがリストに表示されない場合は、次の条件が満たされていることを確認します。
   - ターゲットの階層がアプリケーションの階層と一致する (またはそのスーパーセットである) こと。
   - この環境に対するデプロイアクセス権があること。
   - 各階層に少なくとも1つのサーバーが存在すること。
   - 環境がライフサイクルで「Not Ready」とマークされていないこと。
   - バージョンを含むリリースのライフサイクルには、環境とターゲットが含まれる必要があります。

   その他のターゲットオプションについては、[Select Target] ドロップダウンリストの右側の [**Advanced Target Selector**] をクリックしてください。

- このバージョンを複数のターゲットにデプロイする場合は、**複数のターゲット**を指定することができます。選択できるターゲットは、上記のすべての要件を満たすものだけです。
- SAコアとアプリケーションデプロイメントの設定でサポートされている場合、このデプロイメントの新しいターゲットを作成する際に、**ジャストインタイム**ターゲットを指定することができます。このオプションは、次の条件が満たされている場合のみ使用できます。
  - SAコアがHPE Operations Orchestration (HPE OO) フローを実行するように設定されていること。
  - [Administration] 画面の [Application Settings] 領域で、ジャストインタイムターゲットフローが指定されていること。

詳細については、「ジャストインタイムターゲット」(531ページ)を参照してください。

4. オプション: 右側のパネルのデプロイメントジョブの設定を確認します。これらの設定の一部は変更できます (「[Deployment] 画面」(541ページ)を参照)。セクションを展開するには、アイコンをクリックします。
5. [**Start Job**] をクリックします。これにより、デプロイメントジョブが起動されて、アプリケーションコンポーネントがターゲットサーバーにコピーされ、必要なコード、スクリプト、フローが実行されます。

デプロイメントのステータスは、[Jobs] 画面で見ることができます。

デプロイメントパラメーターのカスタマイズ

パラメーターとは、アプリケーションがデプロイされる環境またはターゲットごとに異なる情報を伝達するために用いられるプレースホルダーです。アプリケーションのパラメーターのデフォルト値は、バージョンごとに異なる場合があります。たとえば、バージョンごとに異なるデータベースユーザーを使用することができます。

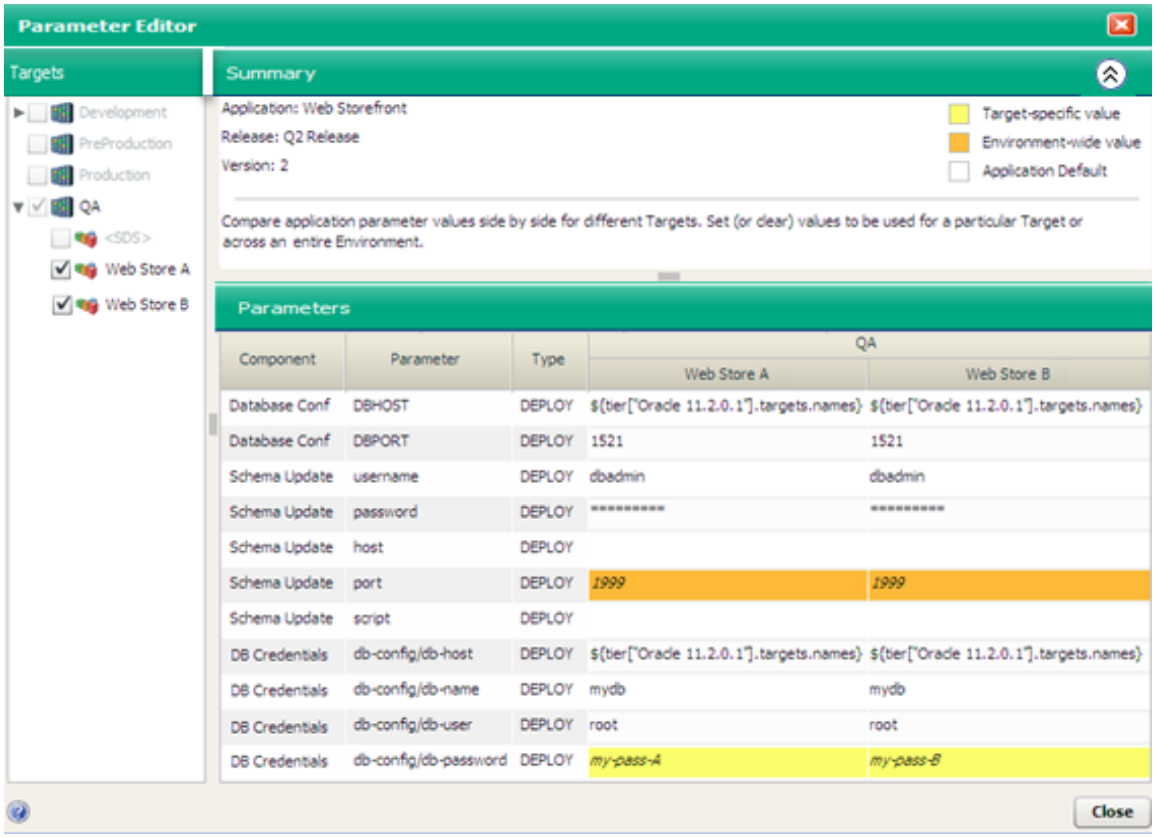
パラメーターは、アプリケーションの機能を指定するコンポーネントで最初に定義されます (「コンポーネント」(469ページ)を参照)。値を設定するには、次の3つの方法があります。

- アプリケーション自体で、コンポーネントが指定されるときに設定します。
- アプリケーションがデプロイされる環境で (アプリケーションをデプロイするユーザーが) 設定します。
- デプロイメント時に特定のターゲットに対して設定します。アプリケーションデプロイメントはターゲット固有の値を記憶するので、デプロイメントのたびに値を設定する必要はありません。

パラメーターは、アプリケーション内のコンポーネントに固有です。パラメーターの名前が変更されず、コンポーネントがデプロイメント時に元のコンポーネントからコピーされる場合は、値は新しいバージョンで再使用されます。

パラメーターエディターを使用することで、デプロイメント時のアプリケーションのパラメーターの値を指定することができます。複数のターゲットにデプロイする場合は、各ターゲットに対して値をカスタマイズできます。次の例では、QA環境内の2つのターゲットが使用されています。Web Store AとWeb Store Bです。

パラメーターエディターの例



ここでは、左側の[Targets]リストで両方のターゲットが選択されているので、両方のターゲットのパラメーター値が右側の[Parameters]テーブルに表示されています。[Targets]リストで使用できるのは、このデプロイメントに対して選択されたターゲットだけです。環境内のその他のターゲットは薄いグレー表示になっています。

[Type]列は、各コンポーネントパラメーターの値のソースを示します(「パラメーターと特殊変数」(506ページ)を参照)。

コンポーネントパラメーターのタイプ

タイプ	パラメーター値のソース
DEPLOY	定数または特殊変数
VERSION	リリースパラメーター
GLOBAL	グローバルパラメーター

コンポーネントパラメーターのタイプ (続き)

タイプ	パラメーター値のソース
	コンポーネントパラメーターが暗号化されている場合、そのコンポーネントパラメーターから参照されるグローバルパラメーターは、パラメーターエディターに表示されません。

このデプロイメントのほとんどのパラメーターには、Web Storefrontアプリケーションのこのリリースが指定されたときに設定されたデフォルト値があります。

2つのパラメーター (Database Confコンポーネントの`DBHOST`とDB Credentialsコンポーネントの`db-config/db-host`) では、「特殊変数」が使用されており、その値はこのバージョンがデプロイされるときに取得されます。この例では、その値はそれぞれのターゲットのOracle 11.2.0.1階層内のサーバーの名前です。

Schema Updateコンポーネントの`port`パラメーターの値は、アプリケーションをデプロイしたユーザーによってこのコンポーネントが構成されたQA環境から取得されます。

DB Credentialsコンポーネントの`db-config/db-password`パラメーターの値は、ターゲットごとに異なります。この例では値は定数ですが、特殊変数にすることもできます。

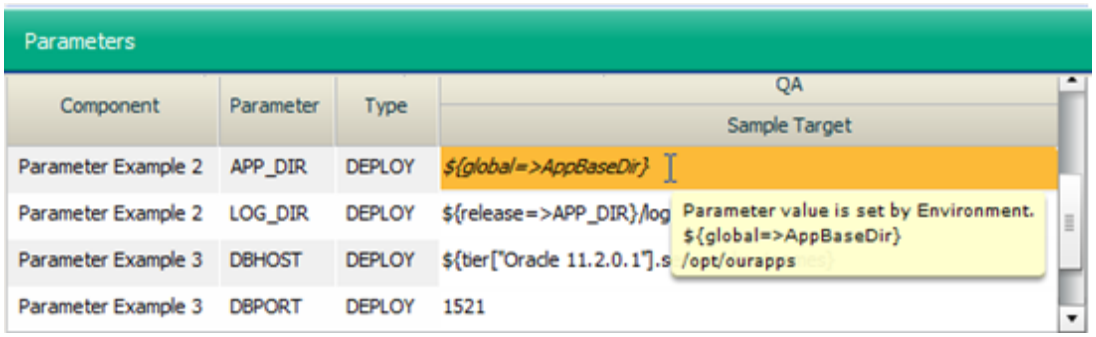
パラメーターの優先順位は次のとおりです。

1. ターゲット固有の値 (テーブルで黄色の背景)
2. 環境内で共通の値 (オレンジの背景)
3. アプリケーションのデフォルト (白の背景)

すなわち、環境内で共通の値は、アプリケーションのデフォルトよりも優先されます。ターゲット固有の値は、環境内で共通の値よりも、アプリケーションのデフォルトよりも優先されます。

パラメーター値の上にマウスポインターを置くと、そのパラメーターの解決済みの値がツールヒントに表示されます。これには、オーバーライドが起きているかどうかも示されます。

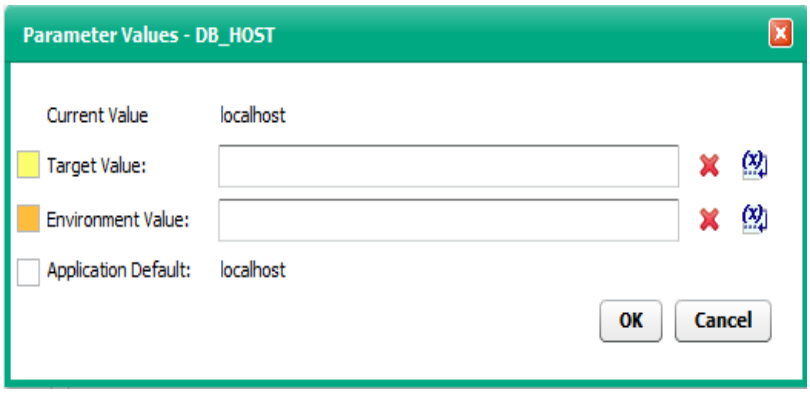
パラメーター値のツールヒントの例



次の図では、APP_DIRパラメーターの値の上にカーソルが置かれています。このパラメーターは、AppBaseDirグローバルパラメーターを参照しています。これは/opt/ourappsに解決されます。これは環境によるオーバーライドです。このことは、ツールヒントからも、テーブルのセルのオレンジの背景からもわかります。

特定のターゲットのパラメーター値を変更するには、次の手順を実行します。

1. [Deployment] 画面で、バージョンとターゲットを指定します。
2. 右側のパネルで、アイコンをクリックして、[Parameters] セクションを展開します。
3. [**Edit**] をクリックするか、テーブルのどれかの行をダブルクリックします。パラメーターエディターが開きます。
4. [Parameters] テーブルで、パラメーターの値をダブルクリックします。[Parameter Values] ダイアログが図に示すように開きます。



暗号化された値は、アスタリスク (*) の列で示されます。

定数値を指定するには、[Target Value] ボックスに値を入力します。

特殊変数、リリースパラメーター、またはグローバルパラメーター (「パラメーターと特殊変数」(506ページ)を参照) を参照するには、次の手順を実行します。

a. アイコンをクリックします。
b. 使用するアイテムをリストから選択します。
c. [**OK**] をクリックします。

[Target Value] ボックスの文字を消去して、次の優先順位の値にリセットするには、[Delete] ボタンをクリックします。. たとえば、ターゲット値を削除した場合、環境値があれば、環境値が使用されます。環境値がない場合、アプリケーションのデフォルト値が使用されます。

5. [**OK**] をクリックして、[Parameter Values] ダイアログを閉じます。

このターゲットだけでパラメーター値を変更した場合は、[Parameters] テーブルの背景が黄色になります。環境内のすべてのターゲットに値を適用した場合は、背景がオレンジになります。

リリースのパラメーターに対して作成された環境またはターゲットの値のオーバーライドは、元のリリースから複製されたすべてのリリースで使用されます。

バージョンのプロパティの表示

バージョンを作成した後、バージョンビューアーを使用して、そのプロパティを調べることができます。バージョンビューアーは、アプリケーションデプロイメントユーザーインタフェースのさまざまな場所から開くことができます。

バージョンビューアーを起動できる場所

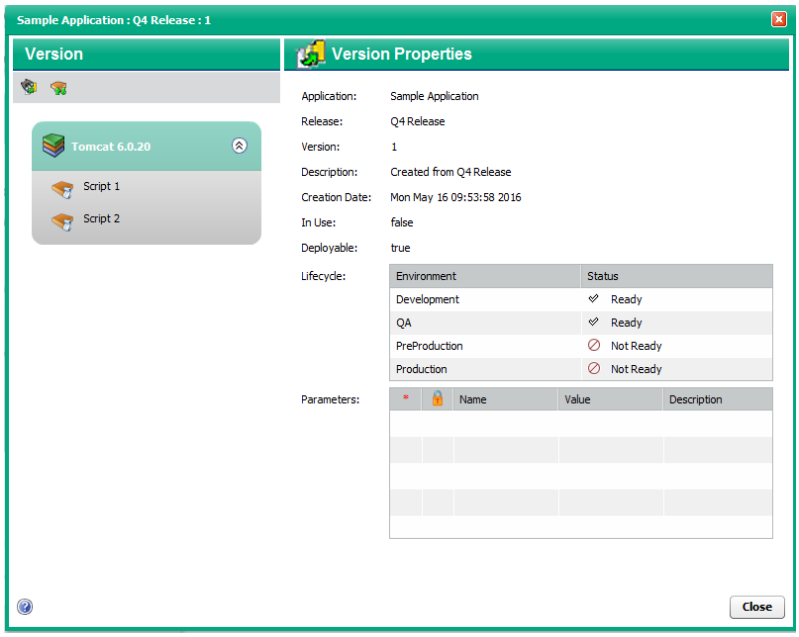
場所	開く方法
[Deployment] 画面	バージョンセレクターのすぐ右にある [**View version**] をクリックします。
[Application] 画面	Manage Applicationsツールを開き、テーブル内のバージョンを選択し、ツールバーの [**View version**] をクリックします。
[Jobs] 画面	[Job Logs] テーブルでデプロイメントジョブを選択し、ツールバーの [**View version**] をクリックします。 バージョンビューアーを開くには、ジョブ詳細ウィンドウで [Application] リンクをクリックする方法もあります。

バージョンビューアーには、現在のコンテキストで選択されているバージョンが表示されます。左側のパネルには階層とコンポーネントの情報が、右側のパネルにはバージョン、階層、またはコンポーネントのプロパティが表示されます。バージョンビューアーに表示される情報は読み取り専用です。

バージョンが作成されるときに、リリース情報は変更できませんが、階層情報は変更可能です。階層構成は、その階層で使用するミドルウェアを指定します。ミドルウェアは更新される場合があるので、これにより古いアプリケーションで新しいミドルウェアを使用することができます。バージョンビューアーに表示される階層情報は、現在の階層構成を表します。

バージョンのプロパティを表示するには、関連するアプリケーションに対する表示アクセス権が必要です (「アプリケーションのアクセス権」(597ページ)を参照)。

バージョンビューアー - 構造ビュー

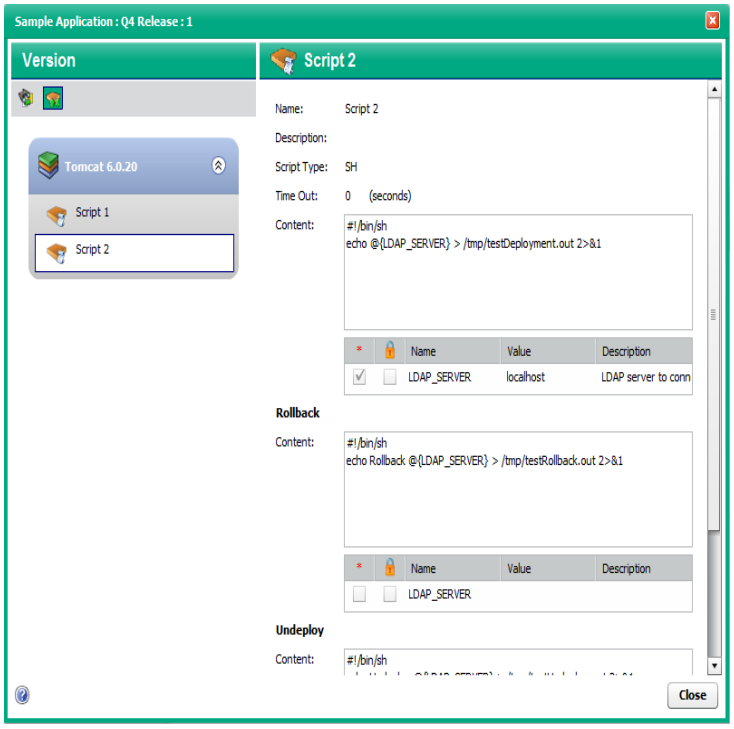


左側のパネルの情報を表示するには、次の2つの方法があります。

- **構造**ビューには、アプリケーションに含まれる階層と、各階層に含まれるコンポーネントが表示されます。
- **シーケンス**ビューには、コンポーネントがデプロイされる順序が表示されます。

どちらのビューでも、左側のパネルで階層またはコンポーネントを選択すると、右側のパネルにそのプロパティが表示されます。

バージョンビューアー - シーケンスビュー



表示される情報をさらに細かく制御するには、ツールバーのボタンを使用します。

バージョンビューアーのツールバーのボタン

ボタン	目的
	左側のパネルで、デプロイされた (またはデプロイされる) 順序でコンポーネントを表示します。
	左側のパネルで、コンポーネント順序 (シーケンス) ビューから階層 (構造) ビューに切り替えます。
	右側のパネルにバージョンのプロパティを表示します。

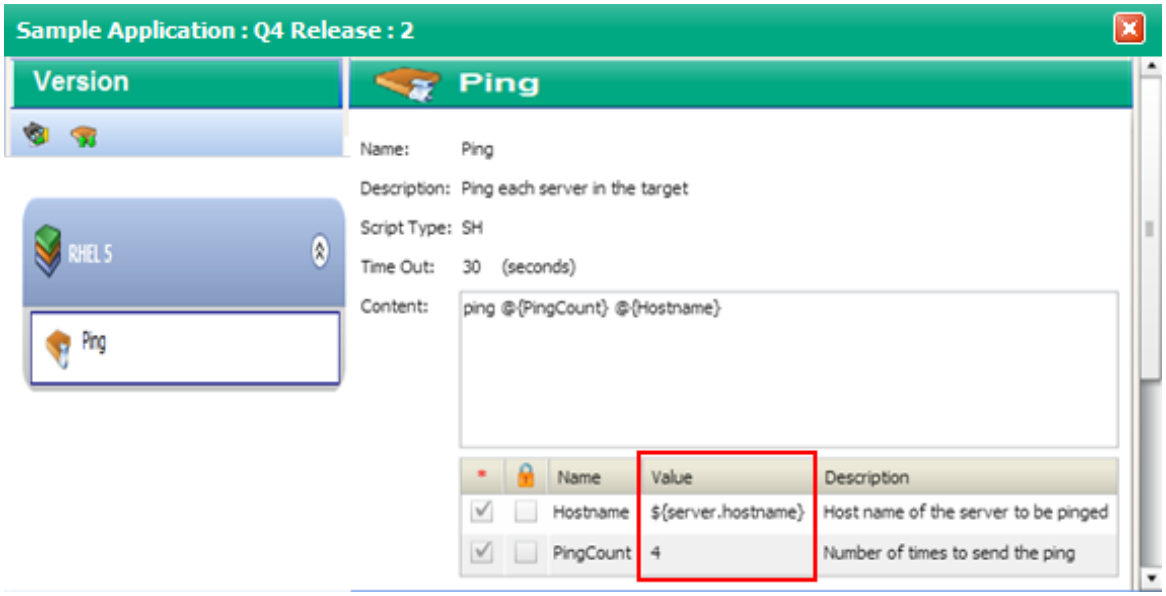
次の点に注意してください。

- [Application] 画面でアプリケーションを表示する場合、バックアップ、ロールバック、アンデプロイスクリプトが有効になっていれば、それらの名前が表示されます。これに対して、バージョンビューアーでは、バージョンが作成された時点で有効になっていたスクリプトの内容が表示されます。スクリプトの内容がバージョンにコピーされるため、その後にスクリプトを変更しても、既存のバージョンには影響しません。
- バージョンビューアーに表示されるパラメーター値は、そのコンポーネントのデフォルト値です。このバージョンがすでにデプロイされている場合、バージョンビューアーに表示される値は、そのデプロイジョブに

対応するジョブまたはステップの詳細ウィンドウに表示されるデプロイメント固有の値とは異なる場合があります(「[Jobs] 画面」(562ページ)を参照)。

次の単純な例では、アプリケーションにPingという名前のスクリプトコンポーネントがあります。これは単にターゲット内の各サーバーをpingするだけです。バージョンビューアーでは、パラメーターのデフォルト値が表示されます。

バージョンビューアーのパラメーターリスト

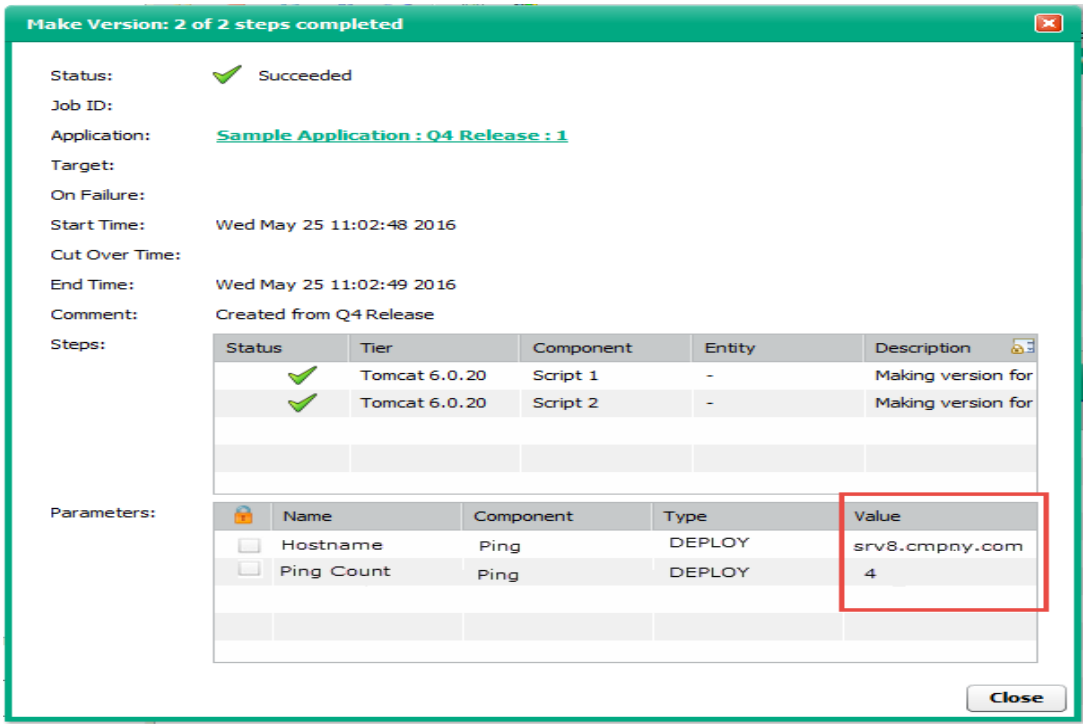


これに対して、このアプリケーションのバージョンをサーバーにデプロイする際には、Hostnameパラメーターの値はそのサーバーの実際のホスト名になります。

この状況は、このデプロイメントジョブのジョブの詳細ウィンドウで確認できます。この例では、特殊変数`${server.hostname}`に、デプロイメント時に`srv8.compny.com`という値が割り当てられています。

PingCountパラメーターは定数(特殊変数ではなく)であることに注意してください。この場合、ターゲットも環境も、デプロイメント時にデフォルト値をオーバーライドしません。このため、パラメーターの値はどちらのビューでも同じ(4)です。

ジョブ詳細ウィンドウのパラメーターリスト



デプロイメントのロールバック

バージョンのデプロイメント中に失敗が発生した場合、そのデプロイメントは失敗の時点から自動的にロールバックされます。[Jobs] 画面では、デプロイした任意のバージョンを手動でロールバックできます。

ロールバック操作が自動と手動のどちらで開始された場合でも、プロセスと結果は同じです。

ロールバック操作は、デプロイ時と逆順にコンポーネントを削除し、関連するファイル、ディレクトリ、またはポリシーを以前の状態に復元します。ロールバック手順は、コンポーネントの定義時に指定されます。コンポーネントに対してロールバックが無効にされている場合、そのコンポーネントはロールバックされません。

コンポーネントタイプごとのロールバック動作

コンポーネントタイプ	ロールバック動作
コード	アプリケーションデプロイメント管理者が管理するOS固有のロールバックおよび復元スクリプトを実行します (「アプリケーションデプロイメントの管理」(600ページ)を参照)。
スクリプト	コンポーネントに指定されている場合は、ロールバックスクリプトを実行します。
構成ファイル	アプリケーションデプロイメント管理者が管理するOS固有の復元スクリプトを実行します。

コンポーネントタイプごとのロールバック動作 (続き)

コンポーネントタイプ	ロールバック動作
アプリケーション構成	サーバーからアプリケーション構成を削除し、元のファイル(保存されている場合)を復元します。
ソフトウェアポリシー	ポリシーをサーバーからデタッチし、関連する既存のポリシー(存在する場合)を復元して、修復します。
パッケージ	デプロイ方法がポリシーの場合、パッケージを含むポリシーをデタッチして修復し、関連するポリシー(存在する場合)をアタッチして修復します。 デプロイ方法がパッケージの場合、パッケージを削除し、関連するパッケージ(存在する場合)をインストールします。
OOフロー	指定されたOOロールバックフロー(指定されている場合)を開始します。
DMAフロー	指定されたDMAフロー(指定されている場合)を実行します。
Windowsレジストリ	アプリケーションデプロイメント管理者が管理するWindows固有のロールバックスクリプトを実行します。

ロールバック操作によって、サーバーはデプロイメントが行われる前の状態に戻ります。

標準のロールバックスクリプトは、各リリースにつき1つのバックアップバージョンだけを保持します。ターゲットサーバー上のディスク領域を節約するため、既存のバックアップファイルは削除されます。このため、最も新しくデプロイされたバージョンより古いバージョンをロールバックした場合、アプリケーションデプロイメントはコードファイルを復元できません。以前のバージョンをロールバックしようとした場合、アプリケーションデプロイメントは警告メッセージを表示します。この場合、ロールバックを確認するかキャンセルすることができます。

デプロイメントを手動でロールバックするには、次の手順を実行します。

1. [Jobs] 画面に移動します(左下の[Jobs]をクリックします)。
2. [Jobs Log] テーブルで、ロールバックするデプロイメントに対応する行を選択します。
3. [**Rollback**] をクリックします。
   [Rollback Job] ダイアログが開きます。
4. [**Yes**] をクリックします。ロールバックジョブが作成され、開始されます。

実行されたコンポーネントが、ロールバックが有効でないものだけだった場合は、ロールバックする対象が存在しないこともあります。この場合、警告メッセージが表示されます。

デプロイメントのアンデプロイ

バージョンをデプロイする際に、前のバージョンがデプロイされていれば、前のバージョンをアンデプロイするように指定することができます。[Jobs] 画面では、デプロイした最新のバージョンを手動でアンデプロイできます。アプリケーションデプロイメント管理者は任意のバージョンをアンデプロイできます。

アンデプロイ操作が自動と手動のどちらで開始された場合でも、プロセスと結果は同じです。

アンデプロイ操作は、デプロイされたのと逆順でコンポーネントを削除します。アンデプロイ手順は、コンポーネントの定義時に指定されます。コンポーネントに対してアンデプロイが無効にされている場合、そのコンポーネントは削除されません。

コンポーネントタイプごとのアンデプロイ動作

コンポーネントタイプ	アンデプロイ動作
コード	デフォルトのインストール場所からファイルを削除します。
スクリプト	コンポーネントに指定されている場合は、アンデプロイスクリプトを実行します。
構成ファイル	アプリケーションデプロイメント管理者が管理するOS固有のアンデプロイスクリプトを実行します。
アプリケーション構成	アプリケーション構成を削除します。
ソフトウェアポリシー	ポリシーをサーバーからデタッチして、修復します。
パッケージ	デプロイ方法がポリシーの場合、パッケージを含むポリシーをデタッチして修復します。 デプロイ方法がパッケージの場合、パッケージを削除します。
OOフロー	指定されたOOアンデプロイフロー (指定されている場合) を開始します。
DMAフロー	指定されたDMAフロー (指定されている場合) を実行します。
Windowsレジストリ	アプリケーションデプロイメント管理者が管理するWindows固有のアンデプロイスクリプトを実行します。

アンデプロイ操作は、バージョンのすべての痕跡を削除します。サーバーを特定の状態に戻すことを目的にしてはいません。

バージョンを手動でアンデプロイするには、次の手順を実行します。

1. [Jobs] 画面に移動します (左下の [Jobs] をクリックします)。
2. [Jobs Log] テーブルで、アンデプロイするデプロイメントに対応する行を選択します。

3. [**Undeploy**] をクリックします。[Undeploy Job] ダイアログが開きます。
4. [**Yes**] をクリックします。アンデプロイが作成され、開始されます。

アプリケーションデプロイメントジョブの管理

単純な例については、「クイックスタート」(440ページ)を参照してください。

概要

アプリケーションデプロイメントは、次のタイプのジョブを作成して実行します。

- 新しいバージョンを作成するたびに、バージョンの作成ジョブが作成されます。
- バージョンをデプロイするたびに、デプロイメントジョブが作成されます (ターゲット1つあたり1つのジョブ)。
- デプロイメントがロールバックされるたびに、ロールバックジョブが作成されます (ターゲット1つあたり1つのジョブ)。これには、デプロイメントの失敗による自動ロールバックと、手動によるロールバックがあります。
- バージョンがアンデプロイされるたびに、アンデプロイメントジョブが作成されます (ターゲット1つあたり1つのジョブ)。これには、新しいバージョンがデプロイされる前の自動アンデプロイと、手動によるアンデプロイがあります。

ジョブには、一連のステップが含まれます。各コンポーネントのデプロイ、ロールバック、またはアンデプロイには、複数のステップが必要です。ステップの数は、使用するコンポーネントのタイプ、使用するコンポーネントの数、サーバーの数によって異なります。

[Jobs] 画面

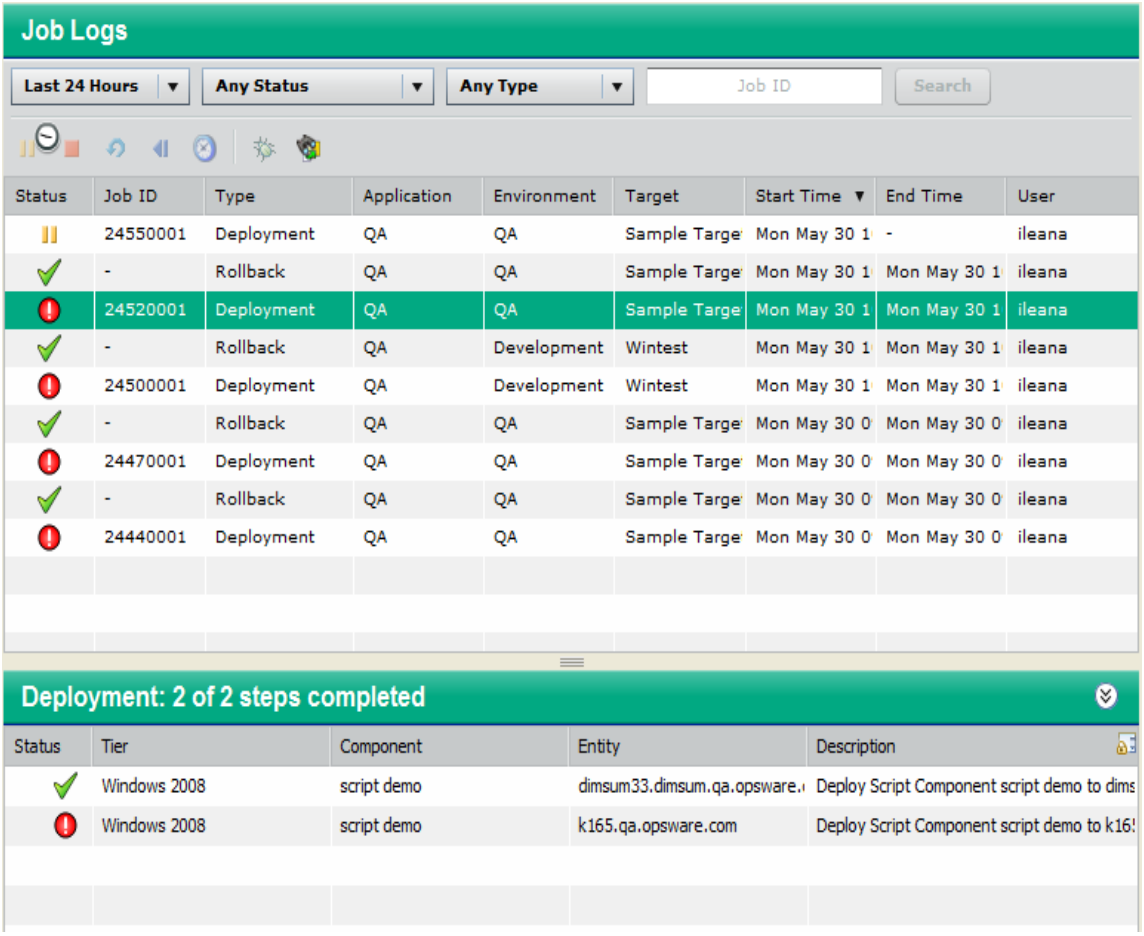
ここでは、次の内容について説明します。

- 「[Jobs] 画面」(563ページ)
- 「ブロックされたジョブ」(568ページ)

[Jobs] 画面

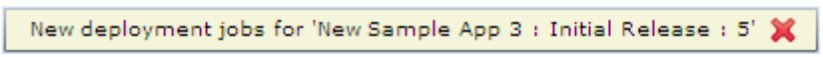
[Jobs] 画面では、アプリケーションデプロイメントジョブの過去からのログを見ることができます。次の図に示すのは、[Jobs] 画面の右側のパネルです。

[Jobs] 画面



[Job Logs] セクションには、指定したフィルター条件に一致するすべてのアプリケーションデプロイメントジョブのリストが含まれます。特定の時間ウィンドウ、ジョブステータス、ジョブタイプ、または特定のジョブIDでフィルターすることができます。

デプロイメントジョブの開始時に [Show job status when deployment starts] をオンにした場合、ジョブログは自動的にフィルターされて、現在のジョブだけが表示されます。[Job Logs] 領域の上部に、次のボックスが表示されます。



このフィルターをクリアするには、上部にある [Clear Search] ボタンを選択します。

[Job Logs] セクションの行を選択すると、そのジョブのステップが下のセクションに表示されます。この例では、次の点に注意してください。

- 正常に完了したことを示すチェックマークがついているステップはありません。
- ステップ1と2は失敗しました:
- ステップ3と8～13はまだスケジュール中です:
- ステップ4～7はスキップされます:

デプロイメントジョブのステップが1つでも失敗すると、ジョブ全体が自動的にロールバックされ、ターゲットサーバーはデプロイメント直前の状態に戻ります。

自動ロールバックジョブは常に作成されます。ロールバックの対象がない場合は、ジョブにはステップがなく、その説明に理由が示されます。

ジョブステータスインジケーター

アイコン	ステータス	説明
	成功	ステップまたはジョブは正常に完了しました。
	失敗	ステップまたはジョブは失敗しました。
	スケジュール済み	ステップまたはジョブは実行するようにスケジュールされています。
	スキップ	このステップに関連するコンポーネントまたはサーバーがデバッガーで選択されていないため、このステップはスキップされます。「デプロイメントジョブのデバッグ」(573ページ)を参照してください。
	スキップ済み	このステップに関連するコンポーネントまたはサーバーがデバッガーで選択されていないため、このステップはすでにスキップされました。「デプロイメントジョブのデバッグ」(573ページ)を参照してください。
	進行中	ステップまたはジョブは現在実行中です。多数のステップを持つジョブの場合、ステップの実行中に、[Status] 列に緑の進行状況バーが表示されます。
	注意が必要	ジョブは失敗し、デバッグを容易にするた

ジョブステータスインジケーター (続き)

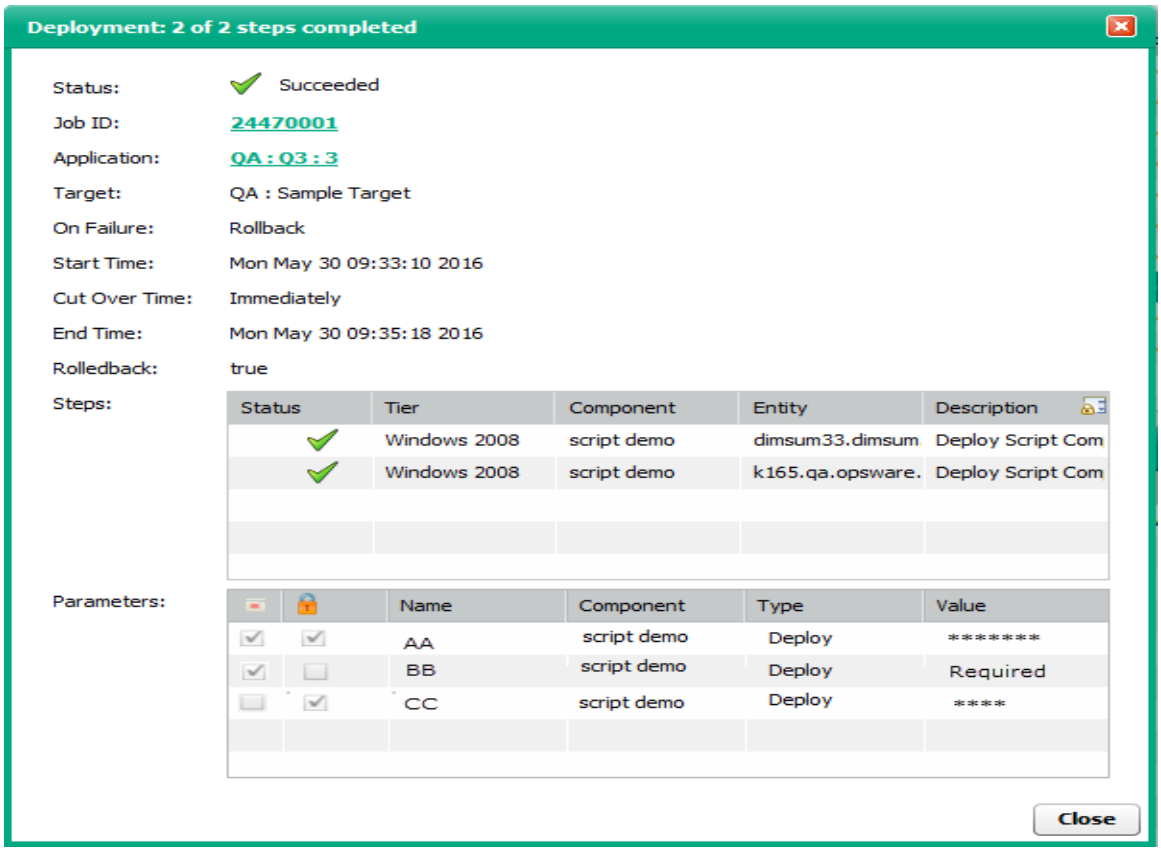
アイコン	ステータス	説明
		めサスペンド状態にあります。「デプロイメントジョブのデバッグ」(573ページ)を参照してください。
	一時停止	ジョブは一時停止されています。
一時停止 (保留中)	[Pause] ボタンが押されましたが、ジョブはまだ一時停止されていません。現在実行中のステップが完了してから、ジョブは一時停止されます。	
	キャンセル	ジョブはキャンセルされました。
キャンセル (保留中)	[Cancel] ボタンが押されましたが、ジョブはまだキャンセルされていません。現在実行中のステップが完了してから、ジョブはキャンセルされます。	

ジョブが進行中である間に、アプリケーションデプロイメント (SAの “da” コンポーネント) が停止され、その後に再開された場合、各ジョブステップのステータスが注意深く調べられ、ジョブのステータスが判定されます。すべてのジョブステップが正常に完了している場合、ジョブステータスは成功に設定されます。どれかのジョブステップが進行中である場合、そのステップは失敗とマークされ、そのステップの出力にエラーが記録され、ジョブも失敗とマークされます。

デフォルトでは、[Job Logs] テーブルで選択されているジョブの最初の進行中のステップが強調表示されます。そのステップが完了して新しいステップが開始されると、強調表示は新しいステップに移ります。強調表示が移動しないようにするには、[Scroll Lock] ボタンを使用します。

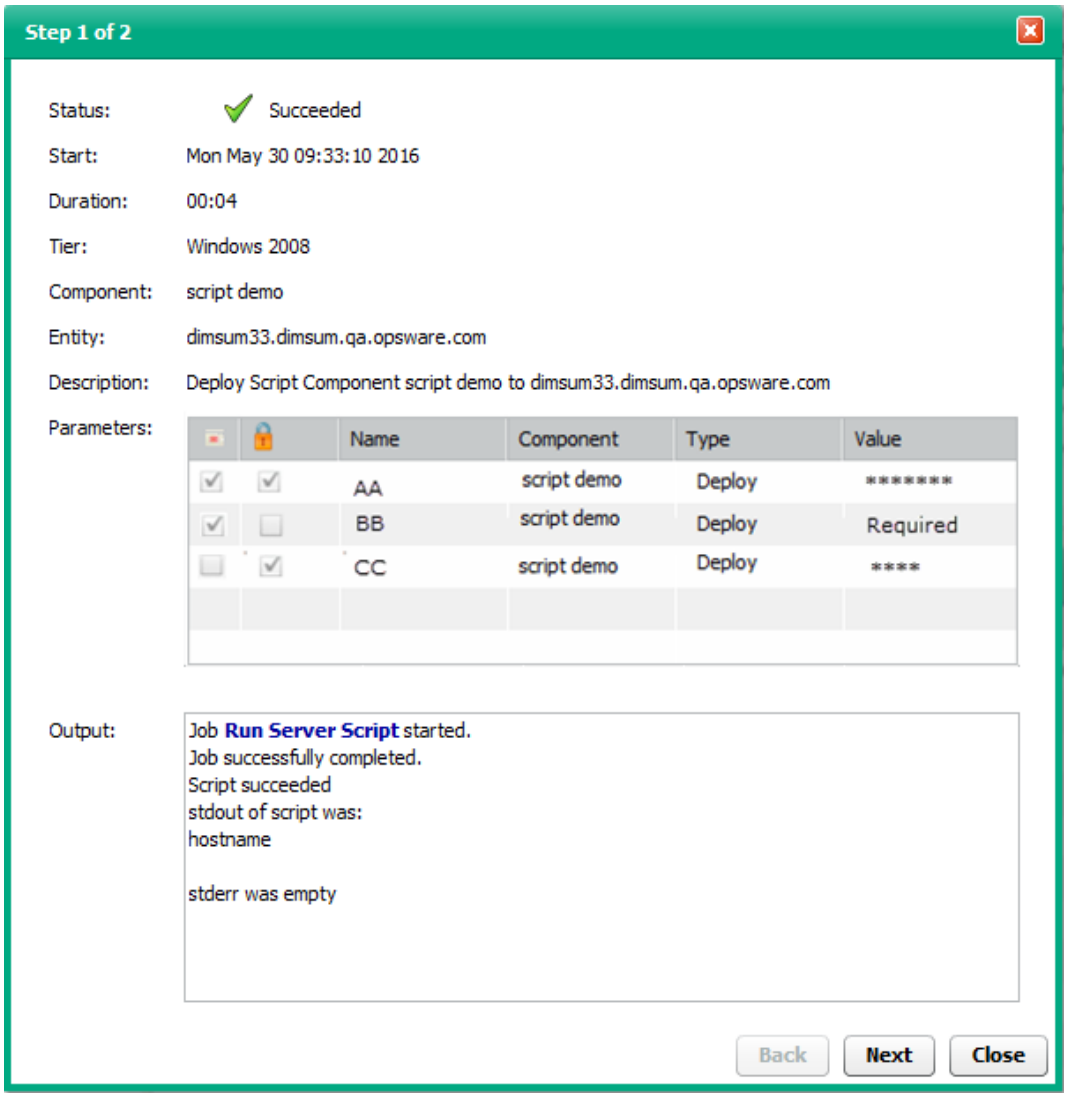
ジョブに関するさらに詳しい情報を表示するには、[Job Logs] テーブルでそのジョブに対応する行をダブルクリックします。新しいウィンドウが図のように開きます。

[Job Detail] ウィンドウ



[Steps] テーブルの行をダブルクリックすると、そのステップの実行に関する詳細が表示されます。次の例では、ステップ4の詳細が表示されています。

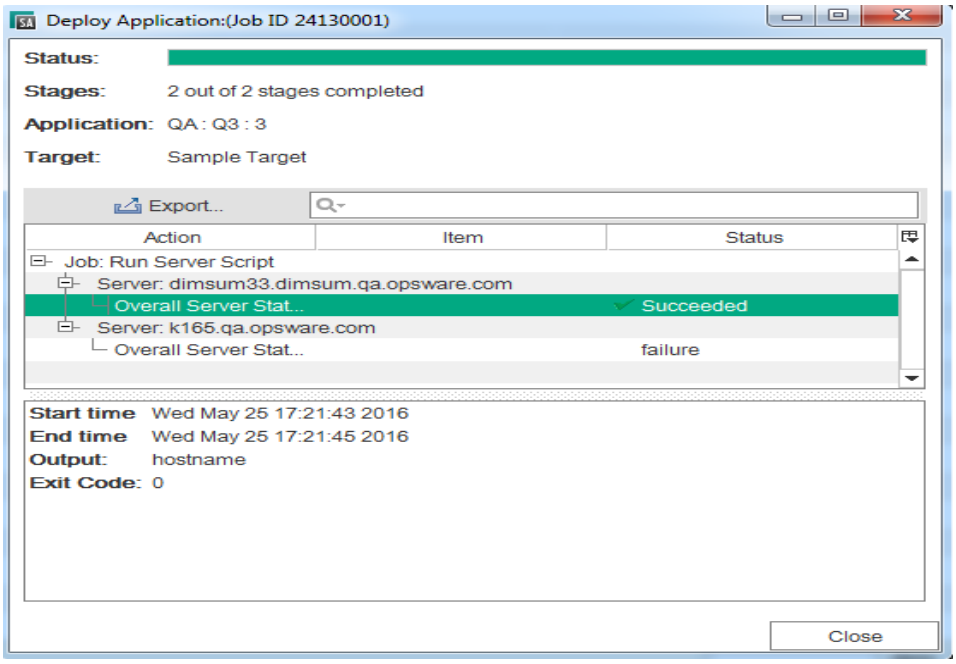
[Step Detail] ウィンドウ



この単純な例では、“script demo” という名前のスクリプトコンポーネントが、3つのパラメーターの値をstdoutに出力します。[Output] ボックスに示されているように、4つのパラメーターの値がstdoutに出力され、スクリプトは正常に完了しました。

[Step Detail] ウィンドウで、[Output] ボックスに表示されるテキストのうち、太字の青のフォントのものは、SAの [Deploy Application] ジョブウィンドウに表示される詳細情報へのリンクになっています。次の例は、[Run Server Script] のリンクをクリックした結果です。

[Deploy Application Job] ウィンドウ



このウィンドウの情報は、サーバー単位で構成されており、ターゲット内の各サーバーが示されています。[Overall Server Status] 行は、元のアプリケーションデプロイメントジョブで特定のサーバー上で実行された1つのステップに対応します。これらの行の1つをクリックすると、特定のサーバー上でのそのステップの実行に関する詳細情報が表示されます。

このウィンドウを開くには、アプリケーションデプロイメントの [Job Detail] ウィンドウでジョブIDをクリックする方法もあります。

ブロックされたジョブ

特定のタイプのジョブに対して [承認が必要] が選択されている場合 (SAクライアントの [管理] タブの [ジョブのブロック] ページ)、アプリケーションデプロイメントは、ジョブが承認待ちでブロックされているというメッセージを表示します。

承認待ちでブロックされているジョブは、SAクライアントの [ジョブとセッション] タブで承認またはキャンセルする必要があります。アプリケーションデプロイメントの [Jobs] 画面からキャンセルすることはできません。

SAクライアントでジョブがキャンセルされると、アプリケーションデプロイメントの [Jobs] 画面にはそのジョブが失敗したと示されます。

詳細については、「開発者ガイドの概要」(23ページ)の「ジョブ承認の統合」のセクションを参照してください。

ジョブの操作

[Jobs] ウィンドウでは、次の操作を実行できます。

- 「特定のジョブの検索」(569ページ)
- 「ジョブの一時停止」(570ページ)
- 「一時停止中のジョブの再開」(570ページ)
- 「ジョブのキャンセル」(571ページ)
- 「ジョブの再スケジュール」(571ページ)
- 「デプロイメントのロールバック」(572ページ)
- 「デプロイメントのアンデプロイ」(572ページ)

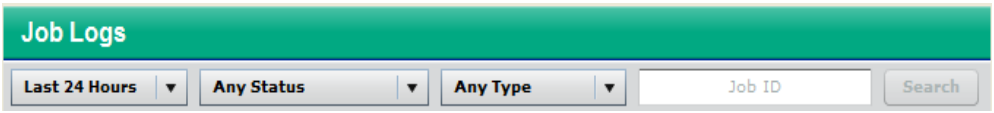
また、アプリケーションデプロイメントデバッガーを使用して、ジョブのトラブルシューティングを実行することもできます。詳細については、「デプロイメントジョブのデバッグ」(573ページ)を参照してください。

[Jobs] 画面から詳細情報を表示する方法については、「[Jobs] 画面」(562ページ)を参照してください。

特定のジョブの検索

検索ツールを使用して、[Job Logs] パネルに表示されるジョブをフィルターしたり、特定のジョブを見つけたりすることができます (ジョブIDがわかっている場合)。

[Jobs] 画面の検索ツール



この例では、先週の間に失敗したデプロイメントジョブだけが表示されます。

ジョブログをフィルターするには、次の手順を実行します。

1. [Jobs] 画面に移動します (左下の [Jobs] をクリックします)。
2. 次のいずれかを実行します。
   - 時間範囲を指定
   - ジョブステータスを指定

- ジョブタイプを指定
- ジョブIDを指定

3. [**Search**] をクリックします。

ジョブIDで検索する場合、ジョブID全体を指定する必要があります (IDの一部やワイルドカードは使用不可)。複数のジョブIDをカンマで区切って指定することができます。

検索の範囲を広げてさらに多くのジョブを表示するには、どれかのドロップダウンリストで [Any] を選択します。

ジョブの一時停止

デプロイした現在進行中のジョブを一時停止することができます。アプリケーションデプロイメント管理者は、現在進行中の任意のジョブを一時停止することができます。

ジョブを一時停止すると、アプリケーションデプロイメントは現在処理中のステップを完了してから一時停止します。現在のステップが完了するまで、ジョブのステータスは「一時停止 (保留中)」になります。現在のステップが完了すると、ジョブのステータスは「一時停止」になり、キャンセルまたは再開されるまでそのままになります。

ジョブを一時停止するには、次の手順を実行します。

1. [Jobs] 画面に移動します (左下の [Jobs] をクリックします)。
2. 一時停止するジョブをジョブログで見つけて選択します (「特定のジョブの検索」(569ページ)を参照)。
3. ツールバーで、[**Pause**] をクリックします。
4. [**Yes**] をクリックして確認します。

ジョブは、一時停止したユーザー (またはアプリケーションデプロイメント管理者) がジョブを再開するまで一時停止したままになります (「一時停止中のジョブの再開」(570ページ)を参照)。

一時停止中のジョブの再開

一時停止したジョブを再開することができます。アプリケーションデプロイメント管理者は、一時停止中の任意のジョブを再開することができます。

ジョブを再開すると、アプリケーションデプロイメントは次にスケジュールされているジョブのステップを開始します。

一時停止中のジョブを再開するには、次の手順を実行します。

1. [Jobs] 画面に移動します (左下の [Jobs] をクリックします)。
2. 再開する一時停止中のジョブをジョブログで見つけて選択します (「特定のジョブの検索」(569ページ)を参照)。
3. ツールバーで、**[Resume]** をクリックします。

ジョブは一時停止された場所から実行を継続します (「ジョブの一時停止」(570ページ)を参照)。

ジョブのキャンセル

デプロイした現在進行中または一時停止中のジョブをキャンセルすることができます。アプリケーションデプロイメント管理者は、進行中または一時停止中の任意のジョブを一時停止することができます。

ジョブをキャンセルすると、アプリケーションデプロイメントは現在処理中のステップを完了してから停止します。現在のステップが完了するまで、ジョブのステータスは「キャンセル (保留中)」になります。現在のステップが完了すると、ジョブのステータスは「キャンセル」になります。残りのステップは、「スケジュール済み」状態のままになります。

キャンセルするジョブがデプロイメントジョブの場合、アプリケーションデプロイメントはデプロイメントのロールバックを試みます。

ジョブをキャンセルするには、次の手順を実行します。

1. [Jobs] 画面に移動します (左下の [Jobs] をクリックします)。
2. キャンセルするジョブをジョブログで見つけて選択します (「特定のジョブの検索」(569ページ)を参照)。
3. ツールバーで、**[Cancel]** をクリックします。
4. **[Yes]** をクリックして確認します。

キャンセルしたジョブを再開することはできません。後でこのバージョンを同じターゲットにデプロイする必要がある場合は、単に同じバージョンをもう一度デプロイします。

ジョブの再スケジュール

デプロイした「スケジュール済み」のジョブを再スケジュールすることができます。アプリケーションデプロイメント管理者は、任意の「スケジュール済み」ジョブを再スケジュールすることができます。

ジョブを再スケジュールするには、次の手順を実行します。

1. [Jobs] 画面に移動します (左下の [Jobs] をクリックします)。
2. 再スケジュールするジョブをジョブログで見つけて選択します (「特定のジョブの検索」(569ページ)を参照)。ジョブは「スケジュール済み」状態である必要があります。
3. ツールバーで、[**Reschedule**] をクリックします。
4. 新しいステージ時間、新しいカットオーバー時間、またはその両方を選択します。
5. [**OK**] をクリックします。

次の点に注意してください。

ジョブがスケジュールされてまだ実行されておらず、ジョブで「自動アンデプロイ」が有効になっている場合、ジョブを再スケジュールすると、アンデプロイメントジョブとデプロイメントジョブの両方を再スケジュールする必要があるという情報が表示されます。

ステージングがすでに発生しており、カットオーバーがまだ発生していない場合は、カットオーバーセグメントだけを再スケジュールできます。

ジョブの実行中は、[**Reschedule**] ボタンは無効になります。

デプロイメントのロールバック

デプロイしたデプロイメントジョブが完了した後で、ジョブを手動でロールバックすることができます。これにより、サーバーはデプロイメントが行われる前の状態に戻ります。アプリケーションデプロイメント管理者は、任意のデプロイメントジョブをロールバックすることができます。

手順については、「デプロイメントのロールバック」(559ページ)を参照してください。

ジョブが失敗したかキャンセルされた場合、ロールバックが有効になっていると、アプリケーションデプロイメントは自動的にジョブをロールバックします。

標準のロールバックスクリプトは、各リリースにつき1つのバックアップバージョンだけを保持します。ターゲットサーバー上のディスク領域を節約するため、既存のバックアップファイルは削除されます。このため、最も新しくデプロイされたバージョンより古いバージョンをロールバックした場合、アプリケーションデプロイメントはコードファイルを復元できません。前のバージョンをロールバックしようとした場合、警告メッセージが表示され、ロールバックを確認またはキャンセルすることができます。

デプロイメントのアンデプロイ

デプロイしたデプロイメントジョブが完了した後で、ジョブを手動でアンデプロイすることができます。これによりデプロイメントの痕跡はすべて削除されますが、サーバーをデプロイメント前の状態に戻すための操作は

行われません。中間のバージョンがデプロイされていない場合は、アンデプロイとロールバックの効果は同じです。アプリケーションデプロイメント管理者は、任意のデプロイメントジョブをアンデプロイすることができます。

アンデプロイは、元に戻す操作としてロールバックよりも強力です。

デプロイメントジョブのデバッグ

アプリケーションデプロイメントデバッガーを使用すると、デプロイメントジョブ、ロールバックジョブ、またはアンデプロイメントジョブのトラブルシューティングを実行できます (バージョンの作成ジョブのデバッグには使用できません)。

次のシナリオを考えます。

1. 特定のコンポーネントの問題のために、デプロイメントジョブが失敗します。
2. このコンポーネントを修正するために変更を加えます。
3. その後、デバッグ可能ジョブを作成し、デバッガーを起動します。
4. デバッガーで、そのジョブのできるだけ多くの部分を無効にします。これにより、変更をすばやくテストして、正しく動作するかどうかを確認できます。
5. 変更が正しく動作したら、ジョブの他の部分を有効にして、ジョブ全体が動作するかどうかを確認します。

デバッガーは、トラブルシューティングの目的に応じて、[Deployment] 画面または [Jobs] 画面から起動することができます。デバッガーでは、次の操作を実行できます。

- ブレークポイントの使用
- 任意のコンポーネントのデプロイまたはスキップ
- 任意のサーバーへのデプロイまたはスキップ
- デプロイメントのシングルステップ実行
- 階層プロビジョニングおよびバックアップステップの有効化または無効化

コンポーネントまたはサーバーをスキップすると、そのコンポーネントまたはサーバーに関連するすべてのジョブステップが、ロールバックおよびアンデプロイメントステップを含めてスキップされます。

デバッガーを実行するには、対象アプリケーションに対する編集アクセス権 (「アクセス権の設定」(586ページ)を参照) またはアプリケーションデプロイメントの管理アクセス権 (「アクセス権の設定」(586ページ)を参照) が必要です。

デバッガーウィンドウ

デバッガーは独立したウィンドウで動作します。このウィンドウには、7つの機能領域があります。

- 「ツールバー」(574ページ)
- 「ステータスバー」(575ページ)
- 「ジョブオプション」(575ページ)
- 「ジョブの選択」(576ページ)
- 「詳細」(577ページ)
- 「ステップ」(577ページ)
- 「パラメーター」(578ページ)
- 「出力」(578ページ)

折りたたまれている領域を展開するには、青いバーにある [Expand] ボタンをクリックします。

デバッガーによる一般的なトラブルシューティング作業の実行方法については、「ジョブのトラブルシューティング」(579ページ)を参照してください。

ツールバー

デバッガーツールバーは、[Job Debugger] ウィンドウの左上にあり、トラブルシューティング中のジョブの実行方法を制御するために使用します。

デバッガーツールバーのボタン

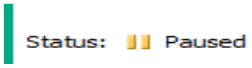
ボタン	名前	目的
	シングルステップ	デバッグジョブの次のステップを実行します。 「スキップ済み」 とマークされたステップは実行されません。
	再開	一時停止したジョブの実行を継続します。 ブレークポイントが設定されている場合、ジョブはブレークポイントに達するまで実行されます。
	一時停止	デバッグジョブを一時停止します。 ジョブの実行中には、再開ボタンの代わりに一時停止ボ

デバッガーツールバーのボタン (続き)

ボタン	名前	目的
		タンが表示されます。
	ジョブのキャンセル	デバッグジョブ全体を中止します。
	ブレークポイントの設定	[「ステップ」(577ページ)] 領域で「スケジュール済み」 ステップを選択し、このボタンをクリックすると、そのステップにブレークポイントが設定されます。ジョブの実行は、ブレークポイントがあるステップの前で停止します。 実行済みのステップや、スキップされるステップに、ブレークポイントを設定することはできません。
	ブレークポイントの削除	[「ステップ」(577ページ)] 領域でブレークポイントのあるステップを選択し、このボタンをクリックすると、ブレークポイントが削除されます。
	更新	[Job Debugger] ウィンドウ全体を強制的に更新します。

ステータスバー

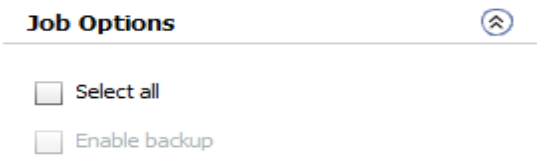
ステータスバーは、[Job Debugger] ウィンドウの左下にあります。ここには、デバッグジョブの現在のステータスが示されます。



ジョブの状態の一覧については、「ジョブステータスインジケーター」(564ページ)を参照してください。

ジョブオプション

[Job Options] 領域は、[Job Debugger] ウィンドウの左上のツールバーのすぐ下にあります。ここでは、次のオプションを設定できます。



[Select all] オプションは、このジョブに含まれるすべてのサーバーとすべてのコンポーネントを選択します。[Select all] チェックボックスをオフにした場合、ジョブのすべてのステップがスキップされます。詳細について

は、「特定のコンポーネントまたはターゲットアイテムの選択またはスキップ」(580ページ)を参照してください。

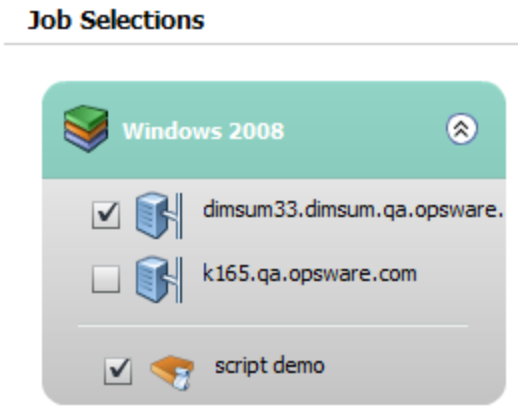
[Enable backup] オプションは、このジョブに含まれる自動バックアップステップを実行します。すべてのバックアップステップを無効にするには、このチェックボックスをオフにします。これにより、デバッグジョブの実行は高速になりますが、ロールバックが適切に行われなくなります。

[Select all] チェックボックスをオフにすると、[Enable backup] チェックボックスは自動的にオフになります。[Select all] チェックボックスをオフにした場合、バックアップステップを含めて、すべてのジョブステップがスキップされるからです。詳細については、「バックアップステップのスキップ」(584ページ)を参照してください。

これらのオプションは、デバッグジョブのステップがまだ1つも実行されていない場合のみ使用可能です。

ジョブの選択

[Job Selections] 領域は、デバッガーウィンドウの左側にあります。



ここでは、デバッグジョブに含めるサーバーやコンポーネントを個別に選択できます。特定のコンポーネントまたはサーバーに対応するチェックボックスをオフにすると、そのコンポーネントまたはサーバーに関連するステップは、ロールバックおよびアンデプロイメントのステップを含めてスキップされます。

上記の例では、サーバー`k165.opsware.com`に関連するすべてのステップがスキップされます。

また、任意の階層に関連付けられたソフトウェアポリシーまたはOSシーケンスのデプロイメントを有効または無効にすることもできます(『SA 10.50管理ガイド』を参照)。

ステップがすでに実行またはスキップされているコンポーネントまたはサーバーは、[Job Selections] エリアで無効になり、選択することはできません。

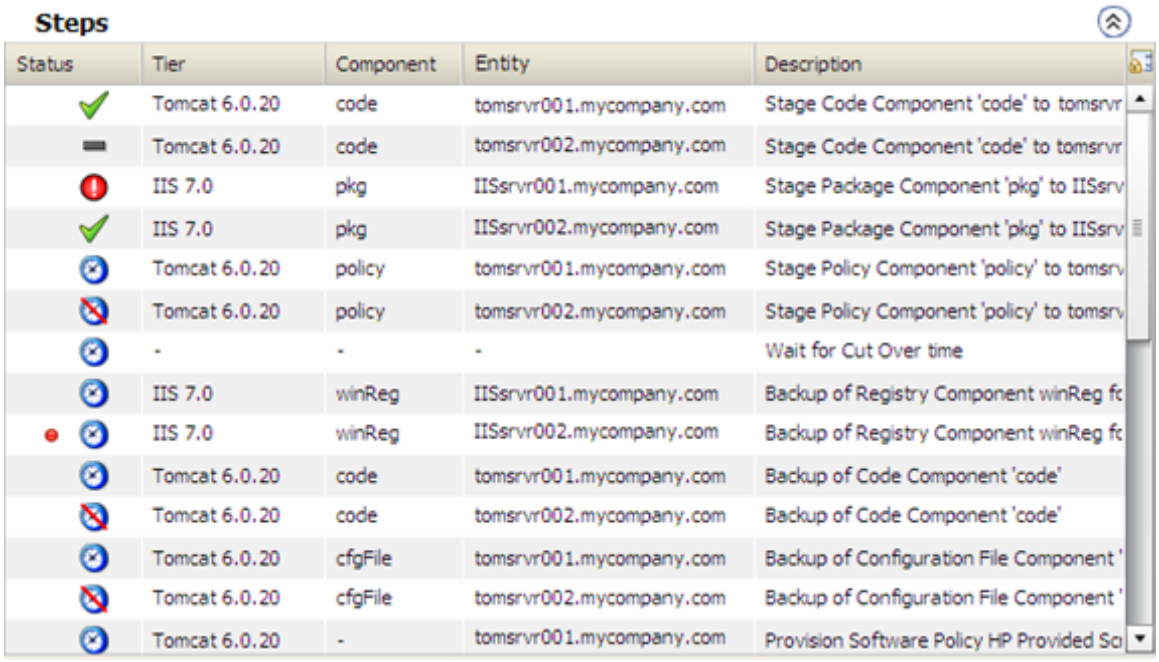
詳細

[Details] 領域は、[Job Debugger] ウィンドウの右上にあります。ここには、トラブルシューティング中のジョブに関する情報が表示されます。[Job ID]、[Target]、[Start Time] の各フィールドは、ジョブが実行されるまで入力されません。たとえば、[Deployment] ページからデバッガーを起動した場合、これらのフィールドは、ジョブの実行可能ステップが少なくとも1つ実行されるまで空白です。

[Details] 領域ではコメントの追加または変更が可能です。このコメントは、[Jobs] 画面の [Job Logs] 領域でこのジョブをダブルクリックしたときに、詳細ウィンドウに表示されます。

ステップ

[Steps] 領域は、[Job Debugger] ウィンドウの右側の [Details] 領域の下にあります。



[Steps] 領域には、アプリケーションデプロイメントがこのジョブを実行する際に実行される個々のステップが表示されます。[「ジョブの選択」(576ページ)] 領域でサーバーまたはコンポーネントのチェックボックスをオフにした場合、そのサーバーまたはコンポーネントに関連するすべてのステップがスキップされます。

上記の例では、サーバー`tomsrvr002.mycompany.com`に関連するすべてのステップがスキップされます。

ステップが実行されると、ステータスアイコンが更新されます。

デバッガーのステップステータスインジケーター

アイコン	ステータス	説明
	スケジュール済み	このステップは実行するようにスケジュールされています。
	スキップ	このステップはスキップされます (これに関連するコンポーネントまたはサーバーが[「ジョブの選択」(576ページ)] 領域で選択されていないため)。
	進行中	このステップは実行中です。
	成功	このステップは成功しました。
	失敗	このステップは失敗しました。
	スキップ済み	このステップはすでにスキップされています。

パラメーター

特定のジョブに対して初めてデバッガーを開いたときに、選択されているステップがない場合、[Parameters] テーブルにはそのジョブのすべてのパラメーターが表示されます。ジョブにパラメーターがない場合、“Job has no parameters” というメッセージが表示されます。

特定のステップを選択した場合、[Parameters] テーブルには、そのステップでデプロイされるコンポーネントに関連するパラメーターだけが表示されます。

ステップにパラメーターがない場合、“Step has no parameters” というメッセージが表示されます。

[Parameters] テーブルは読み取り専用です。デバッガーでパラメーター値をオーバーライドすることはできません。

出力

[Output] 領域には、[「ステップ」(577ページ)] 領域で選択されているステップをアプリケーションデプロイメントが実行した結果が示されます。ここには、ステップが正常に完了したかどうかが示され、`stdout`と`stderr` (該当する場合) の内容が表示されます。

[Output] 領域に太字の青いテキストで表示される情報は、さらに詳細な情報へのリンクになっています。

ジョブのトラブルシューティング

このセクションのトピックでは、デバッガーを使用して次のトラブルシューティング作業を行う方法を示します。

- 「デバッガーの起動」(579ページ)
- 「特定のコンポーネントまたはターゲットアイテムの選択またはスキップ」(580ページ)
- 「デバッグジョブの一時停止」(582ページ)
- 「デバッグジョブの再開」(582ページ)
- 「デバッグジョブのシングルステップ実行」(582ページ)
- 「ブレークポイントの操作」(583ページ)
- 「ステップに関する詳細情報の表示」(583ページ)
- 「デプロイメントへのコメントの追加」(584ページ)
- 「バックアップステップのスキップ」(584ページ)
- 「階層ポリシーまたはプロビジョニングのスキップ」(584ページ)
- 「デバッグジョブのキャンセル」(585ページ)
- 「ロールバックジョブまたはアンデプロイメントジョブのデバッグ」(585ページ)

[Job Debugger] ウィンドウの使用方法については、「デバッガーウィンドウ」(574ページ)を参照してください。

デバッガーの起動

デバッガーを起動するには、次の2つの方法があります。

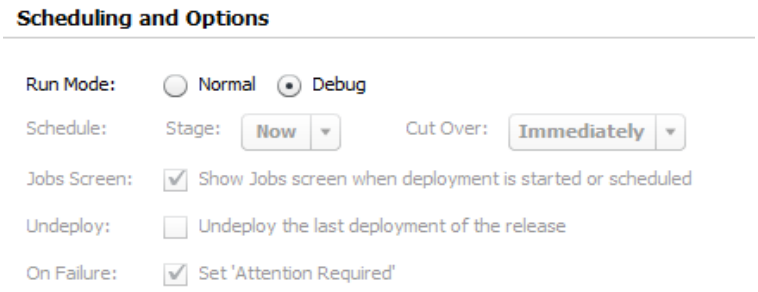
- デプロイメントジョブを最初からトラブルシューティングする場合は、[Deployment] 画面からデバッガーを起動します。
- すでに進行中または一時停止状態のジョブをトラブルシューティングするには、[Jobs] 画面から起動します。

  [Jobs] 画面では、デプロイメントジョブ、ロールバックジョブ、アンデプロイジョブに対してデバッガーを起動できます (バージョンの作成ジョブはデバッグできません)。

デバッガーを起動できるのは、一時停止、進行中、または注意が必要状態のジョブに対してだけです。スケジュール済み、完了、キャンセル、失敗状態のジョブに対しては、デバッガーを起動できません。

[Deployment] 画面からデバッガーを起動するには、次の手順を実行します。

1. [Scheduling and Options] の下で、[**Debug**] モードを選択します。



   [**Debug**] モードを選択すると、デプロイメントスケジュールとオプションが自動的に設定されます。

   これらのオプションの詳細については、「Scheduling and Options」(547ページ)を参照してください。

2. [**Start**] をクリックします。

[Jobs] 画面からデバッガーを起動するには、次の手順を実行します。

1. 右側のペインで、デバッグするジョブを選択します。
2. ツールバーの [**Debug**] をクリックします。

特定のコンポーネントまたはターゲットアイテムの選択またはスキップ

デバッグジョブが一時停止状態にある場合、[Job Selections] 設定を使用して、ジョブの再開時にどのコンポーネントがどのターゲットサーバーにデプロイされるかを指定できます。デプロイ対象は、階層内の任意のコンポーネント、すべてのコンポーネント、またはなしから選択でき、デプロイ先は、その階層の任意のターゲットアイテム、すべてのターゲットアイテム、またはなしから選択できます。

コンポーネントの設定を変更できるのは、そのコンポーネントに関連するステップがまだ1つも開始されていない場合に限ります。同様に、サーバーの設定を変更できるのは、そのターゲットアイテムに関連するステップがまだ1つも開始されていない場合に限ります。

[Deployment] ページで [Debug] モードを選択してデバッガーを起動した場合 (「デバッガーの起動」(579 ページ)を参照)、ステップが実行される前に、ジョブは自動的に一時停止状態になります。この場合、どのコンポーネントやターゲットアイテムを含めるか除外するかを完全に制御することができます。

[Jobs] 画面からデバッガーを起動した場合、[Job Selections] 設定にアクセスできるのは、次のいずれかの条件が満たされている場合だけです。

- [Jobs] 画面で [**Debug**] をクリックする前にジョブが一時停止状態にあった。
- デバッガーで [**Pause**] をクリックした。

どちらの場合でも、コンポーネントまたはターゲットアイテムの設定を変更できるのは、そのコンポーネントまたはターゲットアイテムに関連するステップが1つも開始されていない場合だけです。

いくつかの一般的なデバッグシナリオを実現する手順を以下に示します。これらの例ではすべて、[Deployment] ページで [Debug] モードを選択してデバッガーを起動したことを前提としています。

すべての階層内のすべてのターゲットアイテムにすべてのコンポーネントをデプロイするには、次の手順を実行します。

[Job Options] 領域で、[**Select all**] チェックボックスをオンにします。

特定のコンポーネントまたはターゲットアイテムに関連するすべてのステップをスキップするには、次の手順を実行します。

[Job Selections] 領域で、スキップするコンポーネントまたはターゲットサーバーに対応するチェックボックスをオフにします。

そのコンポーネントまたはターゲットアイテムに関連するすべてのステップに、[Steps] 領域で [Skip]  ステータスが表示されます。

1つの階層内のすべてのターゲットアイテムに1つのコンポーネントをデプロイするには、次の手順を実行します。

1. [Job Options] 領域で、[**Select all**] チェックボックスをオフにします。
2. [Job Selections] 領域で、デプロイするコンポーネントを選択します。その階層内のすべてのサーバーが自動的に選択されます。

1つのサーバーに関連するすべてのコンポーネントをデプロイするには、次の手順を実行します。

1. [Job Options] 領域で、[**Select all**] チェックボックスをオフにします。
2. [Job Selections] 領域で、コンポーネントをデプロイするサーバーを選択します。その階層内のすべてのコンポーネントが自動的に選択されます。

1つのサーバーに1つのコンポーネントをデプロイするには、次の手順を実行します。

1. [Job Options] 領域で、[**Select all**] チェックボックスをオフにします。
2. [Job Selections] 領域で、次の手順を実行します。
   a. デプロイするコンポーネントを選択します。その階層内のすべてのサーバーが自動的に選択されます。
   b. このコンポーネントをデプロイするサーバー以外のすべてのサーバー階層のチェックボックスをオフにします。

デバッグジョブの一時停止

デバッグジョブが進行中状態にあるときに、[「ジョブの選択」(576ページ)] 設定を変更して個々のコンポーネントまたはサーバーを含めるかスキップするには、先にジョブを一時停止する必要があります。

デバッグジョブが一時停止状態にある場合は、[「ジョブのトラブルシューティング」(579ページ)] 設定が有効になり、ステップがまだ実行またはスキップされていない任意のコンポーネントまたはサーバーの設定を変更することができます。

進行中状態にあるジョブを一時停止するには、ツールバーの [Pause] ボタンをクリックします。

ジョブはまず一時停止の保留中状態に移行し (現在進行中のステップの実行を完了するため)、その後に一時停止状態に移行します。

デバッグジョブの再開

一時停止状態にあるジョブを再開するには、ツールバーの [**Resume**] をクリックします。

注意が必要状態のデバッグジョブに対して [Resume] ボタンをクリックした場合、再開するとジョブは失敗状態に移行するという警告メッセージが表示されます。ジョブを再開するには、明示的に [OK] をクリックする必要があります。アプリケーションデプロイメントは、元のデプロイメントジョブのステータスを失敗に設定し、ロールバックジョブを開始します (該当する場合)。

デバッグジョブのシングルステップ実行

一時停止状態のデバッグジョブをシングルステップ実行するには、ツールバーの [**Single Step**] をクリックします。スケジュールされている (スキップされない) 次のステップが実行されます。

注意が必要状態のデバッグジョブに対して [**Single Step**] ボタンをクリックした場合、シングルステップ実行するとジョブは失敗状態に移行するという警告メッセージが表示されます。ジョブをシングルステップ実行するには、明示的に [OK] をクリックする必要があります。アプリケーションデプロイメントは、元のデプロイメントジョブのステータスを失敗に設定し、ロールバックジョブを開始します (該当する場合)。

ブレークポイントの操作

デバッグジョブが一時停止状態にある場合、スケジュールされている (スキップされておらず、まだ実行されていない) 任意のステップにブレークポイントを設定できます。ジョブを再開すると、デバッガーはそのブレークポイントよりも前にあるスケジュールされた (スキップされていない) ステップをすべて実行してから、一時停止します。

ブレークポイントは、1回のアプリケーションデプロイメントユーザーインタフェース (UI) セッション内で保持されます。アプリケーションデプロイメント UIを一度終了してから開き直した場合は、前のセッションで設定したブレークポイントは失われます。

ジョブが完了し、成功または失敗状態になると、デバッグはできなくなるため、すべてのブレークポイントが自動的に削除されます。

ブレークポイントを設定するには、次の手順を実行します。

1. [Steps] 領域で、ブレークポイントを設定するステップを選択します。
2. ツールバーの [**Set Breakpoint**] をクリックします。

すでに開始しているステップにブレークポイントを設定することはできません。

ブレークポイントを削除するには、次の手順を実行します。

1. [Steps] 領域で、ブレークポイントが設定されているステップを選択します。
2. ツールバーの [**Remove Breakpoint**] をクリックします。

すでに開始しているステップのブレークポイントを削除することはできません。

ステップに関する詳細情報の表示

[「ステップ」(577ページ)] 領域でステップを選択 (シングルクリック) すると、[「パラメーター」(578ページ)] 領域と[「出力」(578ページ)] 領域にそのステップに関する情報が表示されます。

[「ステップ」(577ページ)] 領域でステップをダブルクリックすると、そのステップの詳細情報がポップアップウィンドウに表示されます。表示される情報の一覧については、「[Step Detail] ウィンドウ」(566ページ)を参照してください。

デプロイメントへのコメントの追加

デバッグジョブに関連するコメントを追加または変更することができます。これにより、[Jobs] 画面で、コメントフィールドの内容によってデバッグジョブを区別できます。コメントは、デバッグジョブの状態に関わらず追加できます。

コメントを追加するには、次の手順を実行します。

1. デバッガーで、[Details] 領域を展開します。
2. [**Comment**] ボックスにコメントを入力します。
3. [**Save**] をクリックします。
4. コメントへの変更を元に戻すには、[**Reset**] をクリックします。

バックアップステップのスキップ

[Deployment] 画面で [Debug] モードを選択してデバッガーを起動した場合 (「デバッガーの起動」(579ページ)を参照)、デバッグジョブの任意のバックアップステップを有効または無効にすることができます。デフォルトでは、バックアップステップは有効になっています。これらのステップを無効にすると、デバッグ中に時間を節約できます。

コンポーネントタイプによってはバックアップステップがない場合もあります。例：スクリプト、アプリケーション構成、OOフローコンポーネントには、バックアップステップはありません。デバッグジョブにバックアップステップを持つコンポーネントがない場合、[Enable backup] は無効になります。

[Job Options] 設定を変更できるのは、ステップがまだ1つも実行されていない場合に限ります。ジョブのステップが1つでも実行を開始すると、これらの設定は変更できなくなります。
[Jobs] ページで [**Debug**] をクリックしてデバッガーを起動した場合、[「ジョブオプション」(575ページ)] 領域の設定は無効になっています。これは、デバッガーに入るときにはすでにステップが実行されているからです。

すべてのバックアップステップを無効にするには、次の手順を実行します。

[Job Options] 領域で、[Enable backup] チェックボックスをオフにします。

階層ポリシーまたはプロビジョニングのスキップ

[Deployment] 画面で [Debug] モードを選択してデバッガーを起動した場合 (「デバッガーの起動」(579ページ)を参照)、階層に関連するソフトウェアポリシーまたはOSシーケンスを有効または無効にすることが

できます。これらのステップは通常は時間がかかる上、デバッグ中のデプロイメントが失敗している場所によっては不要な場合もあります。

階層設定を変更できるのは、デバッグジョブが一時停止状態にあり、その階層に関連する階層プロビジョニングステップがまだ実行されていない場合に限ります。

階層ポリシーまたはOSプロビジョニングを無効にするには、次の手順を実行します。

1. [Job Selections] 領域で、関連する各階層の [Tier Policy]/[OS Provision] チェックボックスをオフにします。

   この設定が表示されるのは、ソフトウェアポリシー、OSシーケンス、またはその両方が存在する階層だけです (「階層の管理」(601ページ)を参照)。
2. ツールバーのボタンを使用して、デプロイメントジョブを再開またはシングルステップ実行します。

デバッグジョブのキャンセル

実行中または一時停止状態のデバッグジョブをキャンセルすることができます。

デバッグジョブをキャンセルするには、次の手順を実行します。

1. デバッガーで、ツールバーの [**Cancel**] をクリックします。
2. [Cancel Job] ダイアログボックスで、[**Yes**] をクリックします。

   この時点で、次のことが起こります。
   - ジョブはキャンセルの保留中状態に移行します (現在進行中のステップの実行を完了するため)。
   - デバッグジョブが終了されます。
   - 元のデプロイメントジョブがキャンセル状態になります。
   - 元のデプロイメントジョブがロールバックされます (該当する場合)。

ロールバックジョブまたはアンデプロイメントジョブのデバッグ

デバッガーを使用して、ロールバックジョブまたはアンデプロイメントジョブのトラブルシューティングを実行できます。プロセスはデプロイメントジョブのデバッグの場合と似ています。

ロールバックジョブまたはアンデプロイメントジョブをデバッグするには、次の手順を実行します。

1. [Jobs] 画面で、進行中のロールバックジョブまたはアンデプロイメントジョブを選択します。
2. [**Pause**] をクリックしてジョブを一時停止します。
3. ジョブが一時停止状態に移行したら、[**Debug**] をクリックしてデバッガーを起動します。

ロールバックジョブおよびアンデプロイメントジョブのデバッグには、同じ制約が当てはまります。ジョブステップが開始された後でデバッガーを起動しているため、[「ジョブオプション」(575ページ)] 設定は使用できません。ステップがすでに実行またはスキップされているコンポーネントまたはサーバーの [「ジョブの選択」(576ページ)] 設定を変更することはできません。

アクセス権の設定

概要

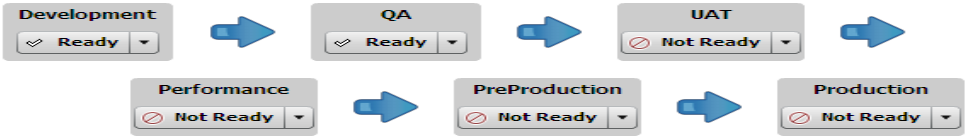
アプリケーションデプロイメントでは、次の2つの事項を細かく制御することができます。

- アプリケーションの作成、表示、編集、デプロイを誰が実行できるか
- アプリケーションコンポーネントの各種ライブラリアイテムを誰が使用できるか
- それらをどこにデプロイできるか

この制御を実現するために用いられるメカニズムが、アクセス権です。アクセス権を使用することで、ソフトウェア開発のライフサイクルを通じたアプリケーションの進展を、秩序ある方式で実現できます。アクセス権を管理する最も便利な方法は、特定のアクセス権のセットを持つユーザーグループを作成することです。

アプリケーションデプロイメントではアクセス権を非常に細かく設定できるので、特定のアプリケーションを特定の環境にデプロイするアクセス権だけをユーザーに与えることもできます。ユーザーの能力は、SAにおける一般的な能力に比べて高いように見えるかもしれませんが、実際にはかなり制限されています。

次のような単純なパッチのライフサイクルを例に取ります。



単純なアプリケーションやパッチなどの開発とリリースの目的では、次のようなユーザーグループを作成することができます。

- 「アプリケーション所有者」(587ページ)
- 「QA環境所有者」(588ページ)
- 「QAエンジニア」(588ページ)
- 「運用環境所有者」(589ページ)
- 「運用エンジニア」(589ページ)

各グループは、付与されたアクセス権の組み合わせに基づいて、特定の能力を持ちます。

アプリケーション所有者

アプリケーション所有者グループのメンバーは、次の種類の作業を実行します。

- アプリケーションの作成
- アプリケーション構成のアプリケーションへのアタッチ
- ソフトウェアポリシーのアプリケーションへのアタッチ
- パッケージのアプリケーションへのアタッチ
- 開発環境へのアプリケーションのデプロイ

上記の作業を実行するために、アプリケーション所有者には次のアクセス権が必要です。

アクセス権のタイプ	必要なアクセス権
アクションのアクセス権	システム管理: 管理対象サーバーおよびグループ アプリケーションデプロイメント: アプリケーションデプロイメントへのアクセス アプリケーションデプロイメント: アプリケーションの作成 アプリケーション構成: アプリケーション構成の管理 (読み取り) パッケージ管理: パッケージの管理 (読み取り) ポリシー管理: ソフトウェアポリシーの管理 (読み取り)
リソースのアクセス権	開発環境サーバーを含むデバイスグループへの読み取りアクセス
フォルダーのアクセス権	アプリケーション構成、パッケージ、ソフトウェアポリシーに対するリストおよび読み取りアクセス権
アプリケーションのアクセス権	アプリケーション所有者は、そのアプリケーションに対する表示、編

(続き)

アクセス権のタイプ	必要なアクセス権
	集、デプロイのアクセス権を、他のユーザーやグループに付与することができます。
環境のアクセス権	開発環境に対するデプロイアクセス権

QA環境所有者

QA環境所有者グループのメンバーは、次の種類の作業を実行します。

- ターゲットの作成
- QA環境内のターゲットへのサーバーの追加

上記の作業を実行するために、QA環境の所有者には次のアクセス権が必要です。

アクセス権のタイプ	必要なアクセス権
アクションのアクセス権	システム管理: 管理対象サーバーおよびグループ アプリケーションデプロイメント: アプリケーションデプロイメントへのアクセス
リソースのアクセス権	QA環境サーバーを含むデバイスグループへの読み取りアクセス
フォルダーのアクセス権	なし
アプリケーションのアクセス権	なし
環境のアクセス権	QA環境に対する編集アクセス権

QAエンジニア

QAエンジニアグループのメンバーは、次の種類の作業を実行します。

- QA環境へのアプリケーションのデプロイ
- デプロイメントジョブのトラブルシューティング

上記の作業を実行するために、QAエンジニアには次のアクセス権が必要です。

アクセス権のタイプ	必要なアクセス権
アクションのアクセス権	システム管理: 管理対象サーバーおよびグループ アプリケーションデプロイメント: アプリケーションデプロイメントへのアクセス
リソースのアクセス権	QA環境サーバーを含むデバイスグループへの読み取りアクセス
フォルダーのアクセス権	なし
アプリケーションのアクセス権	関連するアプリケーションに対するデプロイアクセス権
環境のアクセス権	QA環境に対するデプロイアクセス権

運用環境所有者

運用環境所有者グループのメンバーは、次の種類の作業を実行します。

- ターゲットの作成
- 運用環境内のターゲットへのサーバーの追加

上記の作業を実行するために、運用環境の所有者には次のアクセス権が必要です。

アクセス権のタイプ	必要なアクセス権
アクションのアクセス権	システム管理: 管理対象サーバーおよびグループ アプリケーションデプロイメント: アプリケーションデプロイメントへのアクセス
リソースのアクセス権	運用環境サーバーを含むデバイスグループへの読み取りアクセス
フォルダーのアクセス権	なし
アプリケーションのアクセス権	なし
環境のアクセス権	運用環境に対する編集アクセス権

運用エンジニア

運用エンジニアグループのメンバーは、次の種類の作業を実行します。

- 運用環境へのアプリケーションのデプロイ
- デプロイメントジョブのトラブルシューティング

上記の作業を実行するために、運用エンジニアには次のアクセス権が必要です。

アクセス権のタイプ	必要なアクセス権
アクションのアクセス権	システム管理: 管理対象サーバーおよびグループ アプリケーションデプロイメント: アプリケーションデプロイメントへのアクセス
リソースのアクセス権	運用環境サーバーを含むデバイスグループへの読み取りアクセス
フォルダーのアクセス権	なし
アプリケーションのアクセス権	関連するアプリケーションに対するデプロイアクセス権
環境のアクセス権	運用環境に対するデプロイアクセス権

アクセス権のタイプ

アプリケーションデプロイメントのユーザーインタフェース(UI)でユーザーが表示したりアクセスしたりできる機能を決定するアクセス権には、さまざまなタイプがあります。

「アクションのアクセス権」(591ページ)

次の3つのタイプのアクションアクセス権は、アプリケーションデプロイメントへのアクセスに影響します。

- 管理対象サーバーおよびグループアクセス権は、ユーザーが管理対象サーバーおよびデバイスグループを表示できるかどうかを決定します。この能力は、アプリケーションデプロイメントターゲットにサーバーを追加するために必要です。
- アプリケーションデプロイメントのアクセス権は、アプリケーションデプロイメントのUIにアクセスできるかどうか、アプリケーションを作成できるかどうか、アプリケーションデプロイメント管理者権限があるかどうかを決定します。
- その他の3つのアクセス権によって、アプリケーションコンポーネント内のソフトウェアポリシー、パッケージ、アプリケーション構成、またはOSシーケンスを参照できるかどうか、またはそれらを階層にアタッチできるかどうかが決まります。

すべてのアクションアクセス権は、SAシステム管理者によって設定されます。

「デバイスグループのアクセス権」(595ページ)

これらのアクセス権は、ユーザーがアクセスできるデバイスグループを決定します。デバイスグループへのアクセス権は、そのデバイスグループ内のサーバーをアプリケーションデプロイメントターゲットに追加するために必要です。

デバイスグループのアクセス権は、SAシステム管理者によって設定されます。

「ライブラリフォルダーのアクセス権」(595ページ)

さらに、フォルダーのアクセス権によって、ユーザーがアプリケーション内で参照するか、階層にアタッチできる個々のライブラリアイテム(ソフトウェアポリシー、パッケージ、アプリケーション構成、またはOSシーケンス)が決まります。

フォルダーのアクセス権は、SAシステム管理者によって設定されます。

「アプリケーションデプロイメントのアクセス権」(597ページ)

アプリケーションデプロイメントのアクセス権には、次の2つのタイプがあります。

- アプリケーションアクセス権は、特定のアプリケーションの表示、編集、またはデプロイを実行できるユーザーを決定します。

  アプリケーションのアクセス権は、アプリケーションを作成したユーザー、またはそのアプリケーションに対する編集アクセス権を付与された任意のユーザーによって設定されます。アプリケーションデプロイメント管理者も、アプリケーションのアクセス権を設定できます。
- 環境アクセス権は、特定の環境にアプリケーションをデプロイできるユーザーを決定します。また、その環境でアプリケーションデプロイメントターゲットを作成または変更できるユーザーも決定します。環境のアクセス権は、アプリケーションデプロイメント管理者によって設定されます。

このセクションでは、アプリケーションデプロイメントの一般的な作業の実行に必要な最小限のアクセス権を示します。アクセス権の詳細については、『SA 10.50管理ガイド』を参照してください。

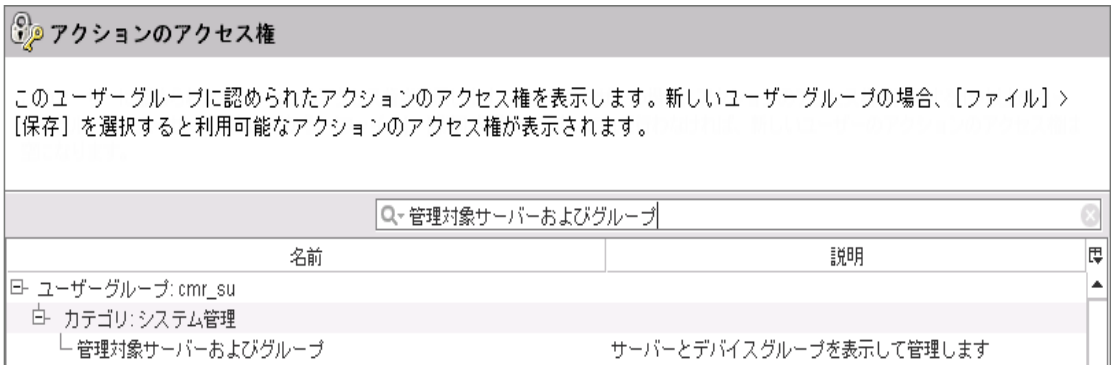
アクションのアクセス権

アプリケーションデプロイメントに関連するSAのアクションのアクセス権を次に示します。

アクションのアクセス権は、SAシステム管理者が、SAクライアントの[管理]画面を使用して、ユーザーグループに付与します(詳細については、『SA 10.50管理ガイド』を参照してください)。

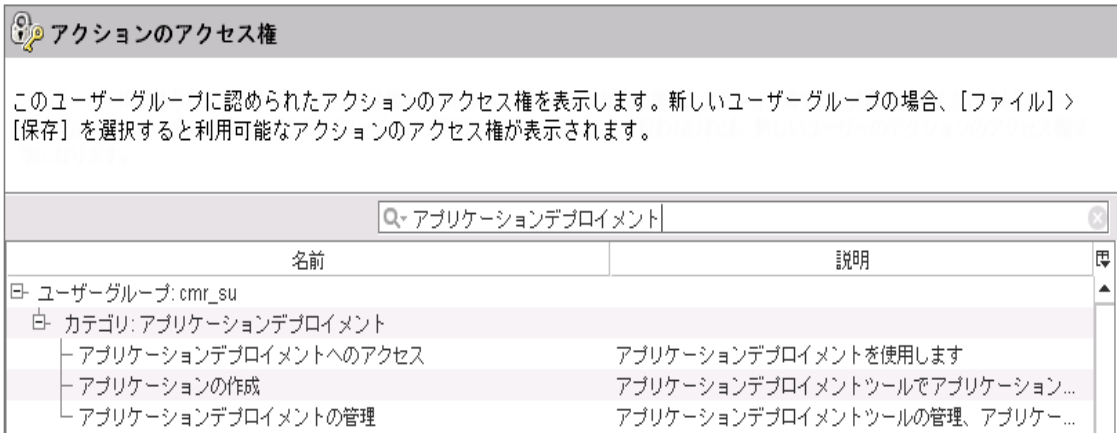
管理対象サーバーおよびグループのアクセス権

アプリケーションデプロイメントターゲットを作成または変更する場合、サーバーおよびデバイスグループの表示と管理の権限が必要です。



アプリケーションデプロイメント へのアクセスのアクセス権

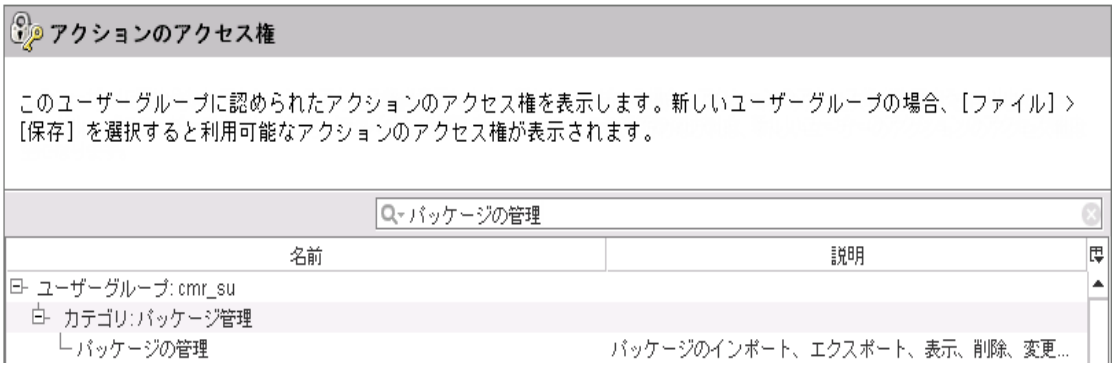
アプリケーションデプロイメント UIにアクセスできるかどうかと、そこで何 ができるかは、次 のアクセス権 によって決 まります。



- アプリケーションデプロイメント へのアクセスは、アプリケーションデプロイメントを起 動 するためのアクセス権 です。このアクセス権 がない場 合 、[ツール] メニューにアプリケーションデプロイメントメニューの項 目 が表 示 されず、Webブラウザーを使 用 してアプリケーションデプロイメントにログインすることはできません。
- アプリケーションの作 成 は、アプリケーションデプロイメントユーザーインタフェースでアプリケーションを作成 するためのアクセス権 です。さらに、各 アプリケーションには、固 有 の表 示 /編 集 /デプロイのアクセス権 があります (「アプリケーションのアクセス権 」(597ページ)を参 照)。
- アプリケーションを特 定 の環 境 にデプロイするには、追 加 のアクセス権 が必 要 です (「環 境 のアクセス権 」(599ページ)と「デバイスグループのアクセス権 」(595ページ)を参 照)。
- アプリケーションデプロイメントの管 理 は、アプリケーションデプロイメント UIの [管 理] 画 面 にアクセスするためのアクセス権 です。また、アプリケーション、ターゲット 、または環 境 の作 成 と変 更 のためのアクセス権 でもあります。

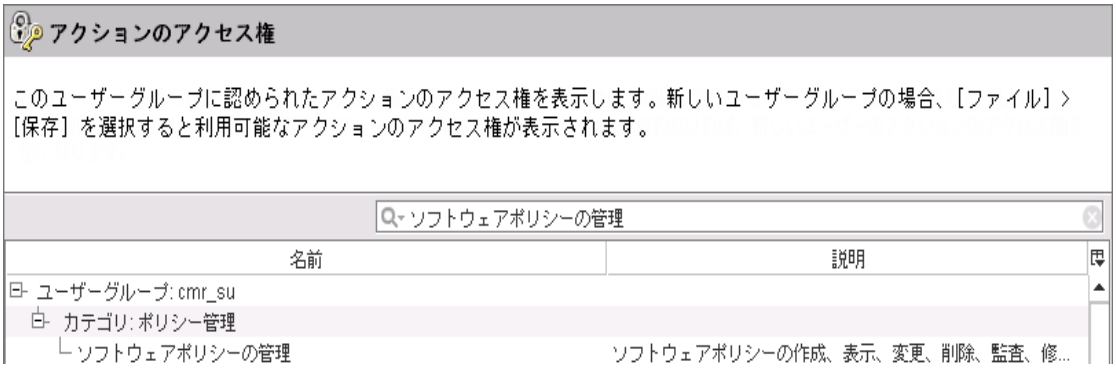
パッケージのアクセス権

アプリケーションコンポーネント内のソフトウェアパッケージへの参照を追加したり、階層にアタッチしたりするには、パッケージの管理 (読み取り) アクセス権が必要です。



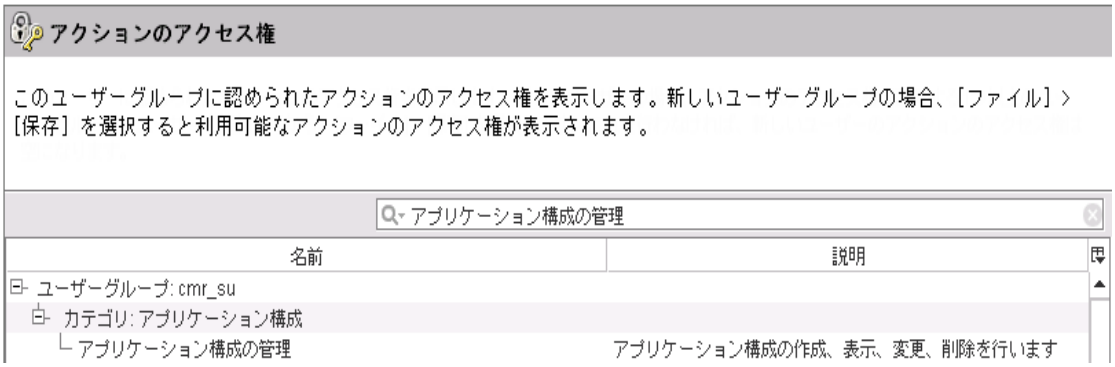
ポリシーのアクセス権

アプリケーションコンポーネント内のソフトウェアパッケージを参照したり、階層にアタッチしたりするには、ソフトウェアポリシーの管理 (読み取り) アクセス権が必要です。



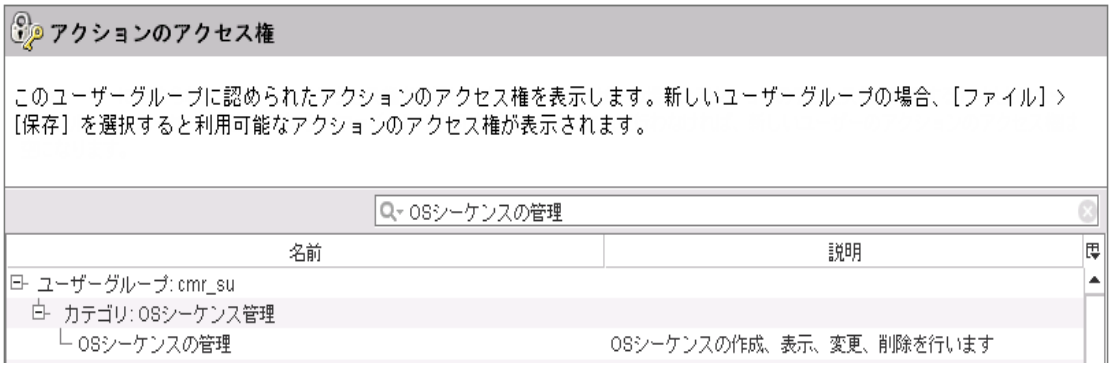
アプリケーション構成の管理のアクセス権

コンポーネント内のアプリケーション構成を参照するには、アプリケーション構成の管理 (読み取り) アクセス権が必要です。



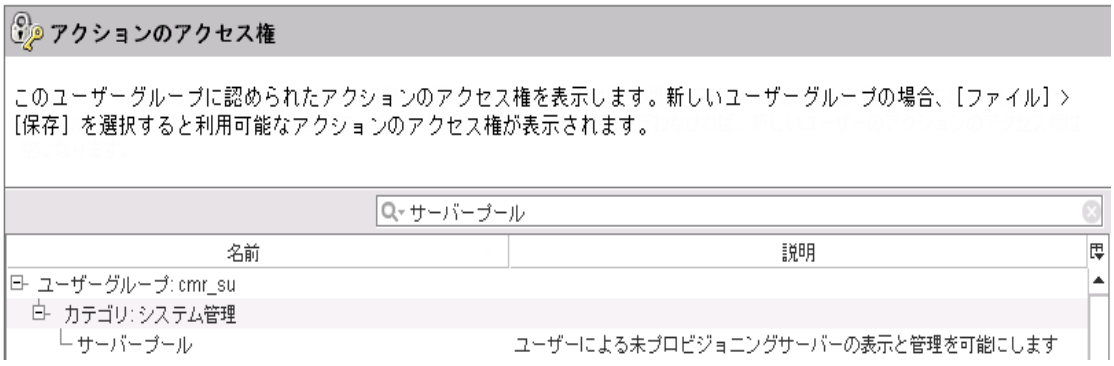
OSシーケンスの管理のアクセス権

コンポーネント内のアプリケーション構成を参照するには、OSシーケンスの管理 (読み取り) アクセス権が必要です。



サーバープールのアクセス権

デプロイメント時にOSシーケンスを使用してサーバーをプロビジョニングするには、サーバープールのアクセス権が必要です。

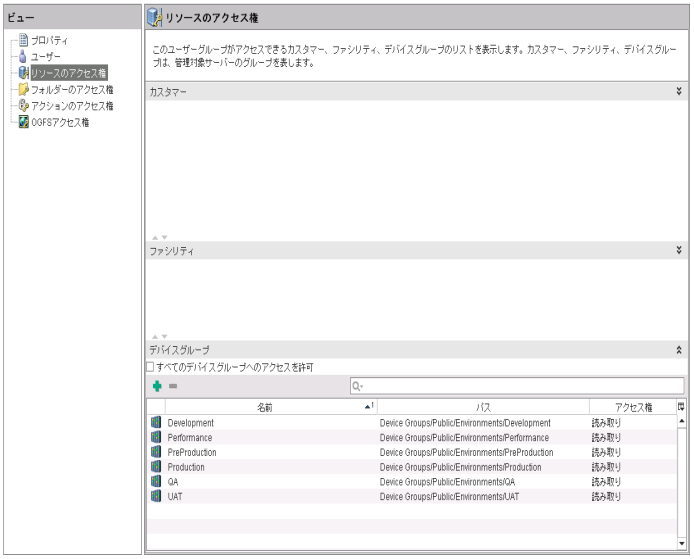


詳細については、「デプロイメント時のサーバーのプロビジョニング」(534ページ)を参照してください。

デバイスグループのアクセス権

アプリケーションデプロイメントターゲットを作成する必要がある場合、これらのターゲットで使用されるサーバーを含むデバイスグループに対する読み取りアクセス権が必要です。たとえば、QA所有者には、QA Serversデバイスグループに対する読み取りアクセス権が必要です。

デバイスグループのアクセス権は、SAシステム管理者が、SAクライアントの[管理]画面を使用して、ユーザーグループに付与します(詳細については、『SA 10.50管理ガイド』を参照してください)。



この例では、このリストに表示されている環境内のすべてのサーバーが、アプリケーションデプロイメントUIの[Add Servers to Target]ダイアログでこのユーザーグループから使用可能になります。

ライブラリフォルダーのアクセス権

このトピックで説明するアクセス権は、SAシステム管理者によって、[Library]画面でフォルダーレベルで設定されます。

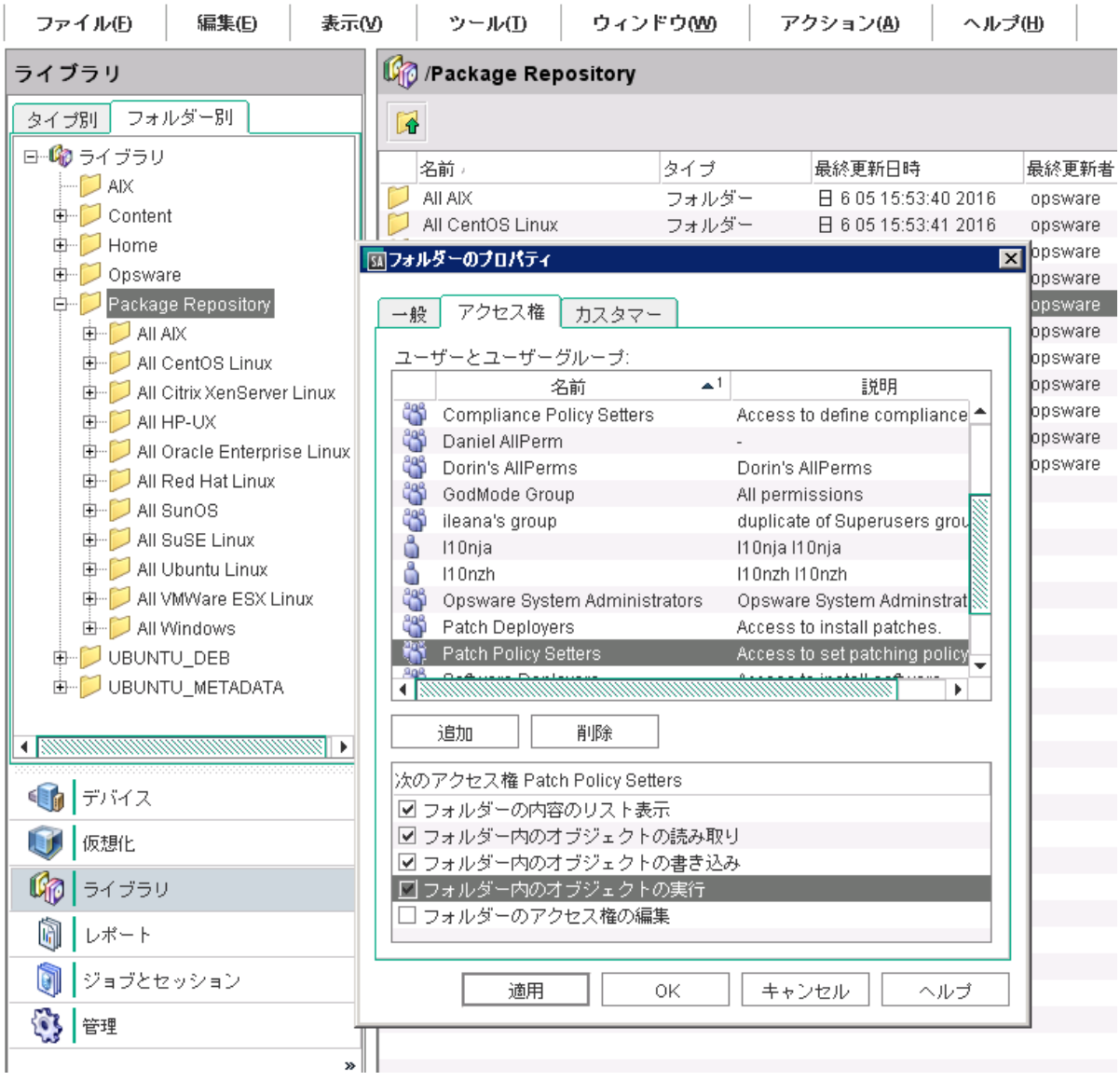
次のアイテムを必要とするアプリケーションを作成または変更するには、ライブラリでそれらの読み取りとリストを行うためのアクセス権が必要です。

- ソフトウェアポリシー
- パッケージ

- アプリケーション構成
- OSシーケンス

ライブラリのアイテムへのアクセスは、フォルダーレベルで制御されます。「パッケージに対してライブラリフォルダーのアクセス権を設定する場所」(596ページ)に、パッケージに対してこれらのアクセス権を設定する方法の例が示されています。

パッケージに対してライブラリフォルダーのアクセス権を設定する場所



また、ライブラリアイテムの各タイプに対する管理設定 (パッケージの管理など) に対する読み取り権限も必要です (「アクションのアクセス権」(591ページ)を参照)。

アプリケーションデプロイメントのアクセス権

アプリケーションデプロイメント UIでユーザーが実行できる機能を決定するアクセス権には、2つのタイプがあります。

- 「アプリケーションのアクセス権」(597ページ)
- 「環境のアクセス権」(599ページ)

アプリケーションのアクセス権は、アプリケーション所有者、またはそのアプリケーションに対する書き込みアクセス権を付与された任意のユーザーによって設定されます。アプリケーションデプロイメント管理者も、アプリケーションのアクセス権を設定できます。

環境のアクセス権は、アプリケーションデプロイメント管理者によって設定されます。

アプリケーションのアクセス権

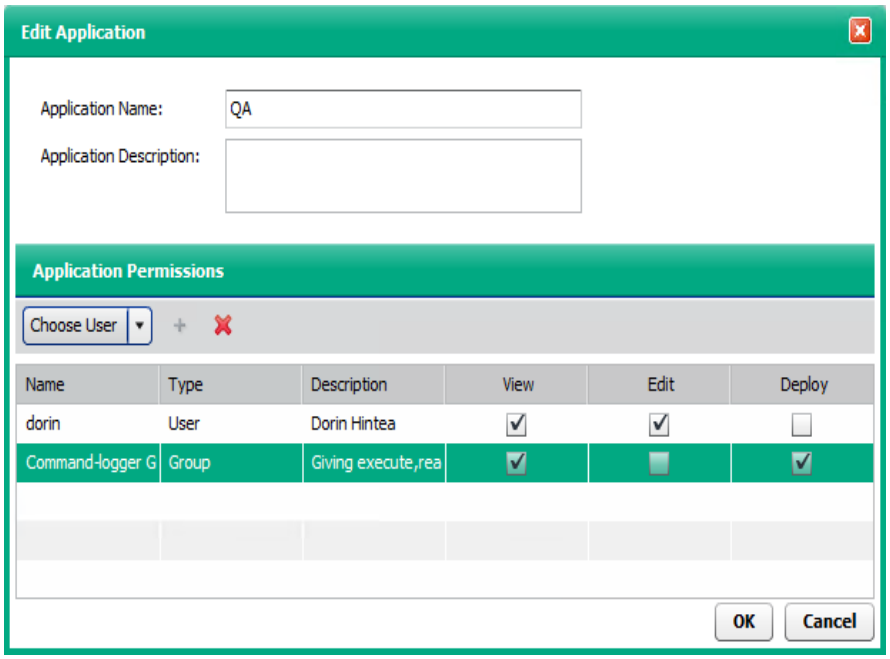
アプリケーションのアクセス権には、次の3つのタイプがあります。

- 表示 – アプリケーションがManage Applicationsツールと[Select Release]ドロップダウンリストに表示されるかどうかを決定します。
- 編集 - アプリケーションのプロパティを編集できるユーザーを決定します。
- デプロイ - アプリケーションをデプロイできるユーザーを決定します。関連する環境にデプロイできるアクセス権も必要です(「環境のアクセス権」(599ページ)を参照)。

デフォルトでは、アプリケーションの表示または編集のアクセス権を持つのは、アプリケーションを作成したユーザー(所有者)とアプリケーションデプロイメント管理者だけです。

その他のユーザーがアプリケーションの表示、編集、またはデプロイを実行できるようにするには、対応するアクセス権をそのユーザーに明示的に付与する必要があります。

アプリケーションのアクセス権を設定する場所



[Choose User] ドロップダウンリストで使用可能なユーザーとグループは、SAクライアントで管理されるユーザーおよびグループと同じです。

アプリケーションのアクセス権を変更できるのは、アプリケーションの所有者、そのアプリケーションの編集アクセス権を明示的に付与されたユーザー、およびアプリケーションデプロイメント管理者だけです。

アプリケーションへのアクセス権を付与するには、次の手順を実行します。

1. [Application] 画面に移動します (左下の [Applications] をクリックします)。
2.  アイコンをクリックして、Manage Applicationsツールを開きます。
3. Manage Applicationsツールで、対象となるアプリケーションを見つけて選択します。
4. [**Edit Properties**] をクリックします (または、編集するアプリケーションをダブルクリックします)。[Edit Application] ダイアログが開きます。
5. [Application Permissions] セクションで、[Choose User] リストから個々のユーザーまたはユーザーグループを選択します。
6. [**Add User**] をクリックします。
7. 付与するアクセス権を、[View]、[Edit]、[Deploy] の中から選択します。
8. アクセス権を付与する各ユーザーまたはグループに対して、ステップ5からステップ7までを繰り返します。
9. オプション: ユーザーまたはグループをリストから削除するには、そのユーザーまたはグループをリストで

選択し、✖をクリックします。

10. [**OK**] をクリックして [Edit Application] ダイアログを閉じ、変更内容を保存します。

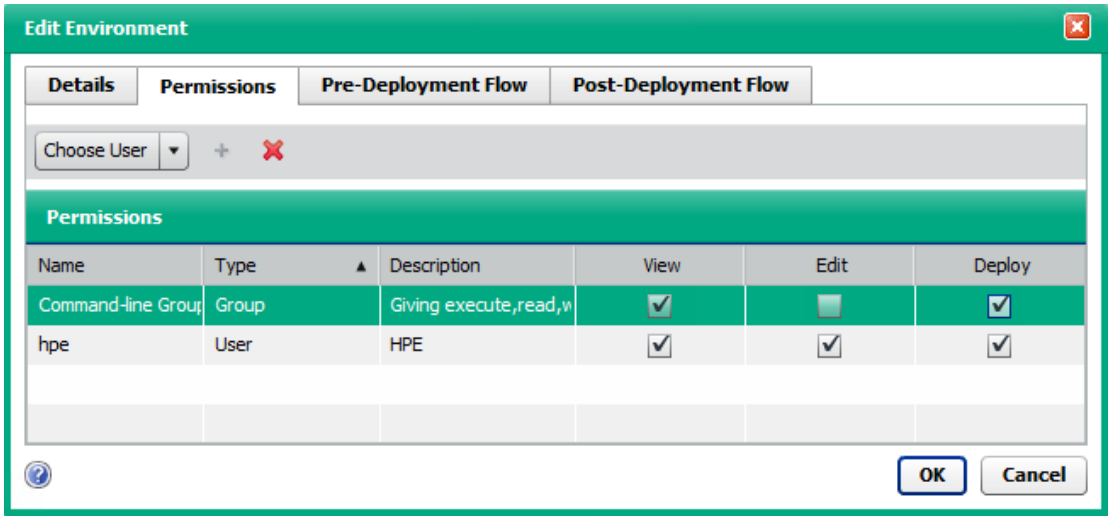
環境のアクセス権

環境のアクセス権には、次の3つのタイプがあります。

- 表示 – ターゲットの構造を表示できるユーザーを決定します。Manage Targetsツールと[Select Target]ドロップダウンリストにターゲットが表示されるのは、関連する環境に対する表示アクセス権がある場合だけです。
- 編集 – 環境内でターゲットを変更できるユーザーを決定します。これには次の作業が含まれます。
  - ターゲットの作成、名前変更、削除
  - ターゲットへのサーバーの追加
  - ターゲットからのサーバーの削除
- デプロイ – この環境内のターゲットにアプリケーションをデプロイできるユーザーを決定します。関連するアプリケーションをデプロイできるアクセス権も必要です (「アプリケーションのアクセス権」(597ページ)を参照)。

デフォルトでは、環境に対する表示、編集、またはデプロイのアクセス権を持つのは、アプリケーションデプロイメント管理者だけです。他のユーザーが特定の環境の表示、編集、またはデプロイを実行する必要がある場合には、アプリケーションデプロイメント管理者が対応するアクセス権を明示的に付与する必要があります。

環境のアクセス権を設定する場所



[Choose User] ドロップダウンリストで使用可能なユーザーとグループは、SAクライアントで管理されるユーザーおよびグループと同じです。

環境のアクセス権を変更できるのは、アプリケーションデプロイメント管理者だけです。

環境へのアクセス権を付与するには、次の手順を実行します。

1. [Administration] 画面に移動します (左下の [Administration] をクリックします)。
2. 左側のパネルで、[Environments] をクリックします。
3. 右側のパネルで、操作する環境を選択します。
4. [**Edit Properties**] をクリックします (または、編集する環境をダブルクリックします)。[Edit Environment] ダイアログが開きます。
5. [Environment Permissions] の下で、[Choose User] リストから個々のユーザーまたはユーザーグループを選択します。
6. [**Add User**] をクリックします。
7. 付与するアクセス権を、[View]、[Edit]、[Deploy] の中から選択します。
8. アクセス権を付与する各ユーザーまたはグループに対して、ステップ5からステップ7までを繰り返します。
9. オプション: ユーザーまたはグループをリストから削除するには、そのユーザーまたはグループをリストで選択し、✖をクリックします。
10. [**OK**] をクリックして [Edit Environment] ダイアログを閉じ、変更内容を保存します。

アプリケーションデプロイメントの管理

この項では、[Administration] 画面で使用できる機能の実行方法を説明します。次のトピックがあります。

- 「階層の管理」(601ページ)
- 「環境の管理」(604ページ)
- 「ライフサイクルの管理」(607ページ)
- 「スクリプトの管理」(609ページ)
- 「コードコンポーネントソースタイプの管理」(611ページ)
- 「アプリケーション設定の管理」(616ページ)

アプリケーションデプロイメントユーザーインタフェースの [Administration] 画面を表示して、この項に示されている設定を変更するには、アプリケーションデプロイメントの管理アクセス権が必要です (「概要」(586ページ)を参照)。

階層の管理

アプリケーションデプロイメント管理者は、アプリケーションの作成とデプロイに使用できる階層を管理できます。次の作業を実行できます。

- 「新しい階層の作成」(601ページ)
- 「新しい階層グループの作成」(602ページ)
- 「階層構造の変更」(602ページ)
- 「階層または階層グループの削除」(603ページ)

これらのプロセスについては、この後で簡単に説明します。すべてのトピックは、アプリケーションデプロイメント管理者権限の存在を前提としています。

新しい階層の作成

階層を管理するには、アプリケーションデプロイメントの管理のためのアクセス権が必要です。

新しい階層を作成して、既存の階層構造内の任意の場所に配置できます。作成した階層は、アプリケーションと環境に対する編集アクセス権を持つすべてのユーザーから使用できます。

新しい階層を作成するには、次の手順を実行します。

1. [Administration] 画面に移動します (左下の [Administration] をクリックします)。
2. 左側のパネルで、[**Tiers**] をクリックします。
3. 右側のパネルで、[**Create Tier**] をクリックします。[New Tier] ダイアログが開きます。
4. 次のプロパティを指定します。
   - Name – 固有の名前を入力します (命名規則を参照)。
   - Group – 新しい階層を作成する階層構造内のグループを見つけて選択します。
   - Platform – WindowsまたはUNIXを選択します。
   - Backup Directory – オプションで、ロールバックが必要になった場合に、この階層内のコンポーネントのバックアップファイルを保存する、ターゲットサーバー上の場所を指定します。

- オプション: Associated Policy – [**Select Policy**] をクリックし、ソフトウェアポリシーをライブラリから選択します。これにより、この階層の機能をサポートするために必要なミドルウェアが、ターゲットサーバー上に存在することが保証されます。
- Deploy Behavior – デプロイメント時に、関連するポリシーがサーバーにまだアタッチされていなかった場合に、アプリケーションデプロイメントがどうするかを指定します。
- オプション: OS Provisioning – [**Select OS Sequence**] をクリックし、OSシーケンスをライブラリから選択します。これは、ターゲットに含まれる未プロビジョニングサーバーに、適切なオペレーティングシステムをインストールして構成するために用いられます。詳細については、「デプロイメント時のサーバーのプロビジョニング」(534ページ)を参照してください。

既存の階層に対して、これらのプロパティを変更することもできます。

5. [**OK**] をクリックします。

新しい階層グループの作成

階層グループを管理するには、アプリケーションデプロイメントの管理のためのアクセス権が必要です (「アクセス権の設定」(586ページ)を参照)。

階層構造内に新しい階層グループを作成できます。その後、既存の階層を新しい階層グループに移動するか、新しい階層をその中に作成することができます。

新しい階層グループを作成するには、次の手順を実行します。

1. [Administration] 画面に移動します (左下の [Administration] をクリックします)。
2. 左側のパネルで、[**Tiers**] をクリックします。
3. 右側のパネルで、[**Create Tier Group**] をクリックします。[New Tier Group] ダイアログが開きます。
4. 次の項目を指定します。
   - Name – 固有の名前を入力します (命名規則を参照)。
   - Group – 新しい階層グループを作成する階層構造内の「親」グループを見つけて選択します。
5. [**OK**] をクリックします。

階層構造の変更

階層を管理するには、アプリケーションデプロイメントの管理のためのアクセス権が必要です (「アクセス権の設定」(586ページ)を参照)。

既存の階層または階層グループを、既存の他の任意の階層グループ内に移動することができます。

階層を別の階層グループに移動するには、次の手順を実行します。

1. [Administration] 画面に移動します (左下の [Administration] をクリックします)。
2. 左側のパネルで、[**Tiers**] をクリックします。
3. 右側のパネルで、階層または階層グループをダブルクリックします。
4. 選択したアイテムの移動先となる階層構造内の「親」グループを見つけて選択します。
5. [**OK**] をクリックします。

階層または階層グループの削除

階層を管理するには、アプリケーションデプロイメントの管理のためのアクセス権が必要です (「アクセス権の設定」(586ページ)を参照)。

使用中の (現在ターゲットサーバーにデプロイされているバージョンで使用されているか、リリースまたはターゲットの一部である) ものを除いて、既存の任意の階層または階層グループを削除できます。階層グループ内のどれかの階層が使用中の場合、その階層グループを削除することはできません。

階層または階層グループを削除するには、次の手順を実行します。

1. [Administration] 画面に移動します (左下の [Administration] をクリックします)。
2. 左側のパネルで、[**Tiers**] をクリックします。
3. 右側のパネルで、削除する階層または階層グループを選択します。
4. [**Delete**] をクリックします。
5. [**Yes**] をクリックして確認します。

使用中の階層を削除するには、先に次の手順を実行する必要があります。

- その階層を使用しているすべてのバージョンをアンデプロイして削除します。
- その階層を使用しているリリースをすべて削除します。
- その階層を使用しているターゲットをすべて削除します。
- 削除できるのは、空の階層グループだけです。

一度使用された階層を削除するのは非常に困難です。

環境の管理

アプリケーションデプロイメント管理者は、アプリケーションデプロイメントに使用可能な環境を管理できます。次の作業を実行できます。

- 「新しい環境の作成」(604ページ)
- 「環境に対するアクセス権の変更」(606ページ)
- 「環境に対するOOフローの指定」(606ページ)
- 「環境の削除」(606ページ)

これらのプロセスについては、この後で簡単に説明します。すべてのトピックは、アプリケーションデプロイメント管理者権限の存在を前提としています。

新しい環境の作成

環境を管理するには、アプリケーションデプロイメントの管理のためのアクセス権が必要です(「アクセス権の設定」(586ページ)を参照)。

新しい環境を作成して、特定のユーザーまたはユーザーグループから使用可能にすることができます。各デプロイメントの前と後に実行するOOフローを1つずつ指定できます。

アプリケーションデプロイメントコンテキスト内の環境は、SAデバイスグループに反映されます。新しい環境を作成すると、次回の同期の後で、対応するデバイスグループが表示されます。

- ただちに同期を行うには、環境を右クリックして、[**Export**] を選択します。
- 対応するデバイスグループを表示するには、環境を右クリックして、[**View Device Group**] を選択します。

新しい環境を作成するには、次の手順を実行します。

1. [Administration] 画面に移動します(左下の[Administration] をクリックします)。
2. 左側のパネルで、[**Environments**] をクリックします。
3. 右側のパネルで、[Create Environment] ボタンをクリックします。  . [New Environment] ダイアログが開きます。

4. [Details] タブで、次のプロパティを指定します。
   - Name – 固有の名前を入力します (命名規則を参照)。
   - Initial Status – 環境のデフォルトステータスを指定します。これは、環境がデフォルトでアプリケーションを受け入れるか、あるいはバージョンごとに明示的な許可が必要かを決定します。この環境へのデプロイのアクセス権を持つユーザーは、[Deployment] 画面でこのステータスを変更することができます。
   - Undeploy – このチェックボックスをオンにすると、デフォルトの自動アンデプロイ設定が有効になります。ユーザーはこの設定をオーバーライドできます。
5. [Permissions] タブで、この環境の表示、編集、デプロイを許可するユーザーを指定します。
   - [View] アクセス権を持つユーザーは、[Select Targets] ドロップダウンリスト ([Targets] および [Deploy] 画面) と [Manage Targets] ツール ([Targets] 画面からアクセス可能) でこの環境を表示できます。
   - [Edit] アクセス権を持つユーザーは、[Edit Environment] ダイアログを開いて、環境のプロパティを変更することができます。また、ターゲットの作成と変更も可能です。
   - [Deploy] アクセス権を持つユーザーは、この環境へのアプリケーション (自分がデプロイアクセス権を持つもの) のデプロイを実行できます。

   ユーザーアクセス権を指定するには、次の手順を実行します。

   a. [Choose User] リストで、個々のユーザーまたはユーザーグループを選択します。
   b. [Add User] ボタン: ＋をクリックします。
   c. 付与するアクセス権を、[View]、[Edit]、[Deploy] の中から選択します。
   d. アクセス権を付与する各ユーザーまたはグループに対して、ステップaからステップcまでを繰り返します。
   e. オプション: ユーザーまたはグループをリストから削除するには、そのユーザーまたはグループをリストで選択し、✖をクリックします。
6. オプション: [Pre-Deployment Flow] タブで、この環境内のターゲットサーバーへのすべてのデプロイメントの直前に開始するOOフローを選択します。たとえば、デプロイメント前に運用環境での監視を無効にすることができます。

   [Change] ボタンをクリックし、OOライブラリからフローを選択します。値を変更するパラメーターの [Value] セルをダブルクリックします (「パラメーターと特殊変数」(506ページ)を参照)。

   デプロイメント前またはデプロイメント後のOOフローに対してパラメーター値を指定する際には、グローバルパラメーターまたはリリースパラメーターを参照できます (「パラメーターと特殊変数」(506ページ) を参照)。

7. オプション: [Post-Deployment Flow] タブで、この環境内のターゲットサーバーへのすべてのデプロイメントの直後に開始するOOフローを選択します。たとえば、デプロイメント後に監視を再び有効にすることができます。
8. [**OK**] をクリックして [New Environment] ダイアログを閉じ、変更内容を保存します。

環境に対するアクセス権の変更

環境を管理するには、アプリケーションデプロイメントの管理のためのアクセス権が必要です(「アクセス権の設定」(586ページ)を参照)。

任意の環境に対する表示、編集、デプロイのアクセス権を変更することができます。手順については、「新しい環境の作成」(604ページ)のステップ5を参照してください。

環境に対するOOフローの指定

環境を管理するには、アプリケーションデプロイメントの管理のためのアクセス権が必要です(「アクセス権の設定」(586ページ)を参照)。

任意の環境に対して、デプロイメント前とデプロイメント後のフローを変更することができます。手順については、「新しい環境の作成」(604ページ)のステップ6とステップ7を参照してください。

環境の削除

環境を管理するには、アプリケーションデプロイメントの管理のためのアクセス権が必要です(「アクセス権の設定」(586ページ)を参照)。

使用中の(現在ターゲットサーバーにデプロイされているバージョンで使用されているか、ライフサイクルの一部である)環境を除いて、既存の任意の環境を削除できます。

環境を削除するには、次の手順を実行します。

1. [Administration] 画面に移動します(左下の [Administration] をクリックします)。
2. 左側のパネルで、[**Environments**] をクリックします。
3. 右側のパネルで、削除する環境を選択します。
4. [**Delete**] をクリックします。
5. [**Yes**] をクリックして確認します。

使用中の環境を削除するには、先にその環境を使用しているすべてのバージョンをアンデプロイする必要があります。また、環境がライフサイクルに属している場合は、そのライフサイクルから環境を削除する必要があります。

ライフサイクルの管理

アプリケーションデプロイメント管理者は、アプリケーションデプロイメントに使用可能なライフサイクルを管理できます。次の作業を実行できます。

- 「ライフサイクルの作成」(607ページ)
- 「ライフサイクルの変更」(608ページ)
- 「ライフサイクルの削除」(608ページ)

これらのプロセスについては、この後で簡単に説明します。すべてのトピックは、アプリケーションデプロイメント管理者権限の存在を前提としています。

ライフサイクルの作成

ライフサイクルを管理するには、アプリケーションデプロイメントの管理のためのアクセス権が必要です(「アクセス権の設定」(586ページ)を参照)。

ライフサイクルは、ソフトウェアアプリケーションが、開発環境から、さまざまなレベルのテスト環境を経て、最終的に運用環境にスムーズに移行できるようにするために用いられます。ライフサイクルを使用すれば、さまざまな環境へのデプロイメントを明示的に許可または禁止することで、1つの環境から次の環境へのアプリケーションの進行を制御することができます。

新しいライフサイクルの作成、既存のライフサイクルの変更、使用されていないライフサイクルの削除を行うことができます。

アプリケーションデプロイメントライフサイクルは、SAの動作にはまったく無関係です。アプリケーションデプロイメントのコンテキストでのみ意味を持ちます。

新しいライフサイクルを作成するには、次の手順を実行します。

1. [Administration] 画面に移動します(左下の[Administration]をクリックします)。
2. 左側のパネルで、[**Lifecycles**]をクリックします。
3. 右側のパネルで、[Create Lifecycle] ボタンをクリックします。  . [New Lifecycle] ダイアログが開きます。

4. 次のプロパティを指定します。
   - Name – 固有の名前を入力します (命名規則を参照)。
   - Lifecycle – 横向きの矢印を使用して、1つ以上の環境を [Available Environments] ボックスから [Lifecycle] ボックスに移動します。縦向きの矢印を使用して、[Lifecycle] ボックス内の環境の順序を変更します。
5. [**OK**] をクリックして [New Lifecycle] ダイアログを閉じ、変更内容を保存します。

ライフサイクルの変更

ライフサイクルを管理するには、アプリケーションデプロイメントの管理のためのアクセス権が必要です (「アクセス権の設定」(586ページ)を参照)。

ライフサイクル内の環境に対して、名前や順序の変更を行うことができます。

ライフサイクルを編集するには、次の手順を実行します。

1. [Administration] 画面に移動します (左下の [Administration] をクリックします)。
2. 左側のパネルで、[**Lifecycles**] をクリックします。
3. 右側のパネルで、編集するライフサイクルを選択します。
4. [**Edit Properties**] をクリックします。[Edit Lifecycle] ダイアログが開きます。
5. オプション: ライフサイクルの新しい名前を入力します。
6. オプション: 横向きの矢印を使用して、1つ以上の環境を [Available Environments] ボックスから [Lifecycle] ボックスに移動します。縦向きの矢印を使用して、[Lifecycle] ボックス内の環境の順序を変更します。
7. [**OK**] をクリックして [Edit Lifecycle] ダイアログを閉じ、変更内容を保存します。

ライフサイクルの削除

ライフサイクルを管理するには、アプリケーションデプロイメントの管理のためのアクセス権が必要です (「アクセス権の設定」(586ページ)を参照)。

使用中の (現在ターゲットサーバーにデプロイされているバージョンまたはリリースで使用されている) ライフサイクルを除いて、既存の任意のライフサイクルを削除できます。

ライフサイクルを削除するには、次の手順を実行します。

1. [Administration] 画面に移動します (左下の [Administration] をクリックします)。
2. 左側のパネルで、[**Lifecycles**] をクリックします。
3. 右側のパネルで、削除するライフサイクルを選択します。
4. [**Delete**] をクリックします。
5. [**Yes**] をクリックして確認します。

ライフサイクルを削除するには、ライフサイクルがリリースまたはバージョンに関連付けられていないことが必要です。

スクリプトの管理

アプリケーションデプロイメント管理者は、デプロイ、バックアップ、ロールバック、アンデプロイ操作に使用するスクリプトを管理できます。これらのスクリプトは、コード、構成ファイル、Windowsレジストリの各コンポーネントから参照されます。次の作業を実行できます。

- 「新しいスクリプトの作成」(609ページ)
- 「既存のスクリプトの変更」(610ページ)
- 「スクリプトの削除」(611ページ)

これらのプロセスについては、この後で簡単に説明します。すべてのトピックは、アプリケーションデプロイメント管理者権限の存在を前提としています。

スクリプトの内容は、バージョンの作成時にコピーされます。スクリプトを変更しても、それより前に作成されたバージョンには影響しません。

新しいスクリプトの作成

スクリプトを管理するには、アプリケーションデプロイメントの管理のためのアクセス権が必要です。

スクリプトは、アプリケーションコンポーネントで参照されます。スクリプトは、デプロイメント、バックアップ、ロールバック、アンデプロイの操作を容易にするために使用されます。基本的なスクリプトのセットが標準で付属しています。これらのスクリプトを変更することも、新しいスクリプトを作成することもできます。アプリケーションを作成するアクセス権を持つユーザーは、コンポーネントを作成する際にこれらのスクリプトを参照できます。

新しいスクリプトを作成するには、次の手順を実行します。

1. [Administration] 画面に移動します (左下の [Administration] をクリックします)。
2. 左側のパネルで、[**Scripts**] をクリックします。
3. 右側のパネルで、[**Create Script**] をクリックします。[New Script] ダイアログが開きます。
4. 次のプロパティを指定します。
   - Name – 固有の名前を入力します (「命名規則」(438ページ)を参照)。
   - Component – このスクリプトを参照するコンポーネントのタイプ。
   - Purpose – [Deploy]、[Undeploy]、[Backup]、[Rollback] のいずれかを選択します。コンポーネントによって、スクリプトは異なるデプロイメントフェーズで使用されます。たとえば、デプロイスクリプトを使用するのは、Windowsレジストリコンポーネントだけです。
   - Platform – WindowsまたはUNIXを選択します。
   - Primary – このチェックボックスをオンにすると、このプラットフォーム上のこのタイプのコンポーネントに対して同じ目的で使用されるスクリプトが複数存在する場合に、これが新しいコンポーネントに使用するデフォルトのスクリプトであることを指定できます。これはたとえば、新しいスクリプトの開発やテストの際に役立ちます。
   - Type – スクリプトのタイプを選択します。使用可能な選択肢は、指定したプラットフォームによって異なります。
   - Content – スクリプトを構成する実際の命令を入力します。
5. 既存のスクリプトに対して、これらのプロパティを変更することもできます。
6. [**OK**] をクリックして変更内容を保存し、[New Script] ダイアログを閉じます。

既存のスクリプトの変更

スクリプトを管理するには、アプリケーションデプロイメントの管理のためのアクセス権が必要です (「アクセス権の設定」(586ページ)を参照)。

[Edit Script] ダイアログを使用して、既存の任意のスクリプトを変更できます。

スクリプトを変更するには、次の手順を実行します。

1. [Administration] 画面に移動します (左下の [Administration] をクリックします)。
2. 左側のパネルで、[**Scripts**] をクリックします。
3. 右側のパネルで、変更するスクリプトを選択します。
4. [**Edit Properties**] をクリックします。[Edit Script] ダイアログが開きます。

5. スクリプトのプロパティに必要な変更を加えます(「新しいスクリプトの作成」(609ページ)を参照)。
6. [**OK**] をクリックして変更内容を保存し、[Edit Script] ダイアログを閉じます。

スクリプトの削除

スクリプトを管理するには、アプリケーションデプロイメントの管理のためのアクセス権が必要です(「アクセス権の設定」(586ページ)を参照)。

使用中の(リリースで使用されている)スクリプトを除いて、既存の任意のスクリプトを削除できます。

スクリプトを削除するには、次の手順を実行します。

1. [Administration] 画面に移動します(左下の[Administration] をクリックします)。
2. 左側のパネルで、[**Scripts**] をクリックします。
3. 右側のパネルで、削除するスクリプトを選択します。
4. [**Delete**] をクリックします。
5. [**Yes**] をクリックして確認します。

スクリプトを削除するには、リリースから参照されていないことが必要です。

コードコンポーネントソースタイプの管理

アプリケーション開発者は、コードコンポーネントを使用して、アプリケーションに必要なファイルを指定します。コードコンポーネントソースタイプは、これらのファイルへのアクセス方法をアプリケーションデプロイメントに伝える役割を果たします。

ソースタイプは有効または無効にできます。どの時点でも、少なくとも1つのソースタイプが有効にされている必要があります。参照されているソースタイプを無効にすることはできません。ソースタイプを無効にするには、そのソースタイプを参照しているバージョンをすべて削除し、そのソースタイプを参照しているリリースを変更する必要があります。

アプリケーションデプロイメントは、オープンソースのビルドシステムであるCruiseControlを使用して、ソースタイプがファイルシステムまたは特定のソースコード管理システム(CVS、Subversionなど)であるコードコンポーネントを管理します。これらのソースタイプに対しては、必要なCruiseControl構成設定を含むXMLスニペットを指定する必要があります。ここでは、「ファイルシステム」(612ページ)と「ソースコード管理システム」(614ページ)に関する例を示します。

CruiseControlの設定の詳細については、次のリソースを参照してください。

http://cruisecontrol.sourceforge.net/main/configxml.html

コードコンポーネントソースタイプの編集

コードコンポーネントソースタイプを管理するには、アプリケーションデプロイメントを管理するためのアクセス権が必要です。

[Edit Code Component Source Type] ダイアログボックスでは、すべてのタイプのコードコンポーネントソースの設定を編集できます。どのタイプのコードコンポーネントソースを編集する場合でも、最初にこのダイアログを開く必要があります。

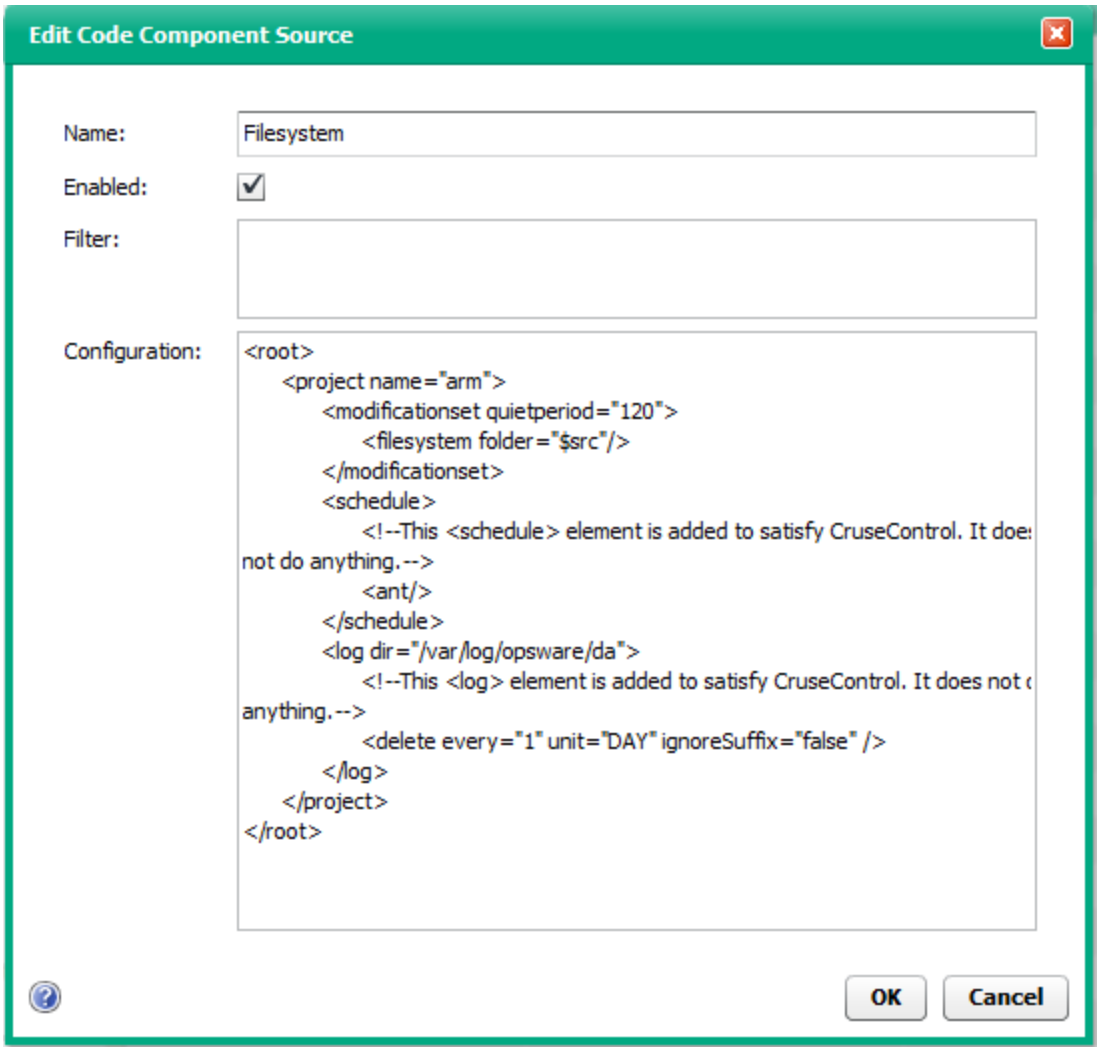
コードコンポーネントソースを編集するには、次の手順を実行します。

1. [Administration] 画面に移動します (左下の [**Administration**] をクリックします)。
2. 左のペインで、[**Code Component Source Types**] を選択します。
3. 右のペインで、次のいずれかを実行します。
   - 新規ソースタイプを作成するには、[**Create Source Type**] をクリックします。
   - 既存のソースタイプを変更するには、ソースタイプを選択し、[**Edit Properties**] をクリックします。[Edit Lifecycle] ダイアログが開きます。
4. このコードコンポーネントソースタイプをアプリケーション開発者から使用可能にするには、[**Enabled**] を選択します。
5. 次に示す個々のコンポーネントソースタイプの詳細な編集手順を実行します。
   - ファイルシステム
   - URL
   - ソースコード管理システム

ファイルシステム

コードコンポーネントに対してファイルシステムソースを設定するには、ここに示すCruiseControl設定を指定する必要があります。

ファイルシステムソースタイプの設定



`quietperiod`は、前回の変更が行われてから、新しいバージョンが作成可能になるまでの秒数を表します。

`$src`変数は、コードコンポーネントに指定されているソースディレクトリを表します。

URL

URLコードコンポーネントに対して編集できる設定は、[Name] フィールドと[Enable] フィールドだけです。URL自体はアプリケーション開発者が指定する必要があります。「URLの使用」(475ページ)を参照してください。

ソースコード管理システム

ソースコード管理システム(CVS、Subversionなど)との統合のために、アプリケーションデプロイメントにはオープンソースのビルドシステムであるCruiseControlが組み込まれています。CruiseControlの設定は、ソースタイプの一部として、XMLスニペットで記録されます。他のソースコード管理システムを追加するには、単に新しいXMLスニペットを追加するだけで済みます。

Server Automationコアの時刻は、ソースコード管理システムのサーバーと同期されている必要があります。そうしないと、どのファイルをコードコンポーネントに含めるかの判定が、時刻のずれのために誤りになる可能性があります。

アプリケーションデプロイメントとの統合の前に、必ずソースコード管理システムのクライアントソフトウェアをSAコアサーバーにインストールしてください。

ソースコード管理システムを統合する方法

ソースコード管理システムのクライアントソフトウェアをSAコアサーバーにインストールします。

SAコアサーバー上に、コードコンポーネントに含めるソースのクライアントコピーを作成します。CVSおよびSubversionでは、これはリポジトリの「チェックアウト」と呼ばれています。

[Edit Code Component Source] ダイアログを開きます(上の「コードコンポーネントソースタイプの編集」(612ページ)を参照)。

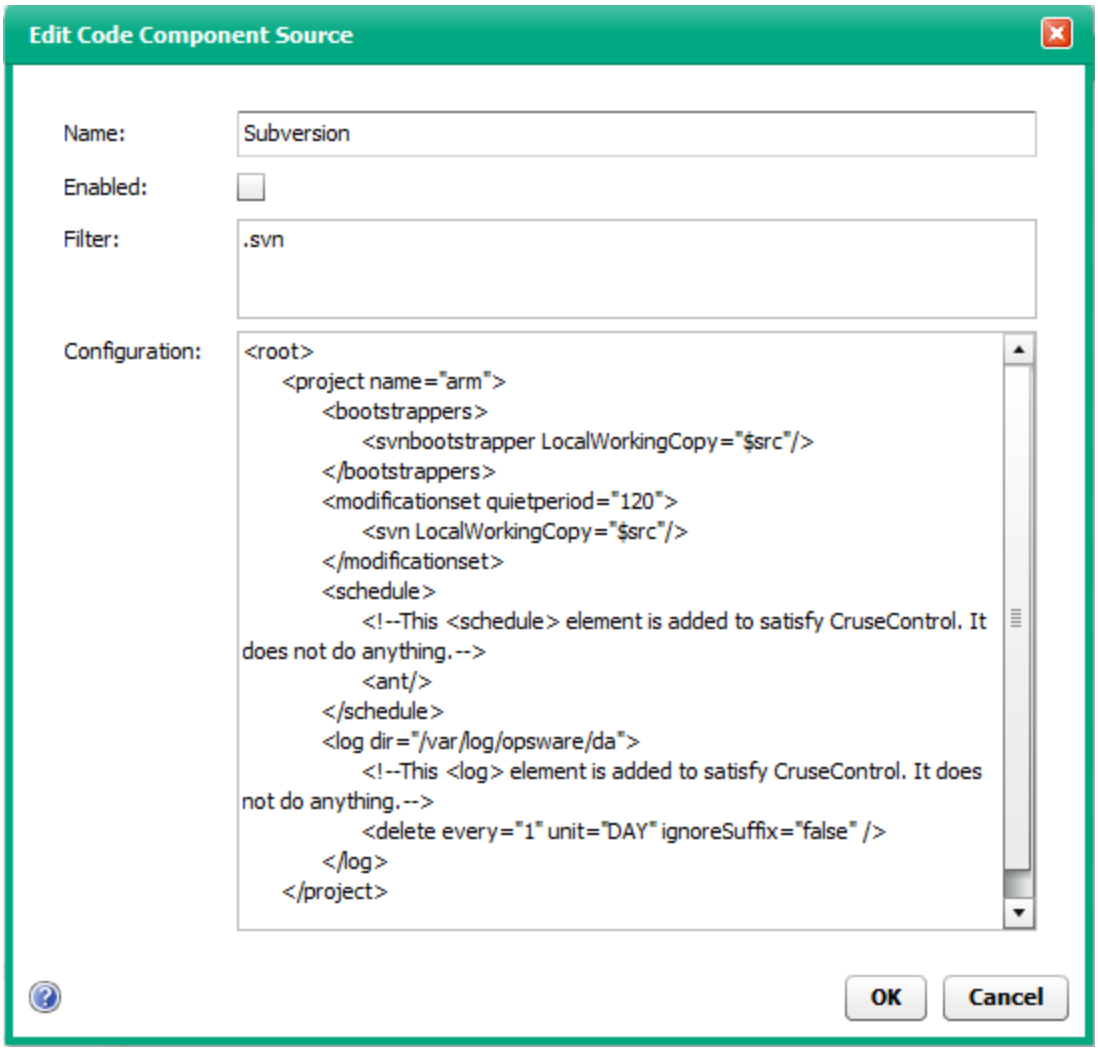
ソースコード管理システムに必要なCruiseControl設定およびフィルター条件を追加します。

それが終わったら、アプリケーション開発者は、このクライアントコピーのパスを、コードコンポーネントで参照することができます。「ソースコード管理システムの使用」(475ページ)を参照してください。

例: Subversion

次に示すのは、Subversionコードコンポーネントソースのサンプル設定です。

Subversionソースタイプの設定



SAコアサーバー上でこのSubversionモジュールをチェックアウトするには、次のコマンドを使用します。

```
svn checkout URL
```

例:

```
svn checkout https://mySVNserver.mycompany.com:444/svn/proj1/trunk
```

SAマルチコアまたはマルチスライス環境でのSVNクライアントのセットアップ

マルチコアまたはマルチスライス環境で、すべてのサーバーにSubversion (SVN) クライアントをインストールする手間を避けるには、ネットワークファイルシステム (NFS) マウントが使用できます。次のプロセスはLinuxサーバー上でテストされています。

マスターコアサーバー上で、次の手順を実行します。

1. SVNクライアント (CollabNet Subversionクライアントなど) を次のディレクトリにインストールします。
   `/opt/CollabNet_Subversion`
2. `/etc/exports`ファイルに次の行を追加します。
   `/opt/CollabNet_Subversion *(ro)`
3. 次のコマンドを使用して、NFSサービスを再起動します。
   `/etc/init.d/nfs restart`

2番目 (およびそれ以降) のサーバーで、次の手順を実行します。

1. マスターコアのエクスポートされたドライブにマウントします。たとえば、次の手順を実行します。

   ```
   cd /opt
   mkdir CollabNet_Subversion
   mount <MasterCore>:/opt/CollabNet_Subversion /opt/CollabNet_Subversion
   ```

   ここで、MasterCoreはマスターコアサーバーのDNS名またはIPアドレスです。
2. `/usr/bin/svn`からSVNへのリンクを作成します。

   ```
   ln -s /opt/CollabNet_Subversion/bin/svn /usr/bin/svn
   ```

これで、SVNが2番目のサーバーにインストールされているように使用できるようになります。

同じNFSマウント方法を使用して、CVSクライアントをセットアップすることもできます。

アプリケーション設定の管理

アプリケーションデプロイメントの動作を決定するアプリケーション設定を管理することができます。

アプリケーション設定を変更するには、次の手順を実行します。

1. [Administration] 画面に移動します (左下の [**Administration**] をクリックします)。
2. 左のペインで、[**Application Settings**] を選択します。
3. 次の設定を必要に応じて変更します。
   - 「バージョン番号」(617ページ)
   - 「ジャストインタイムターゲットフロー」(618ページ)

- 「コードおよび構成ファイルコンポーネントのベースディレクトリ」(618ページ)
- 「グローバルパラメーター」(621ページ)

4. [**Save**] をクリックして変更内容を保存します。

これらの設定を変更するには、アプリケーションデプロイメントの管理のためのアクセス権が必要です。

バージョン番号

このグループには、次の3つの設定が含まれます。

初期バージョン

リリースの初期バージョンの名前と番号の付け方を指定します。デフォルトは1です。番号を指定した場合、新しいバージョンを作成するたびに、アプリケーションデプロイメントはそれを起点としてバージョン番号を増やしていきます。

ここに指定した値は、ユーザーがリリースの最初のバージョンを作成したときに、[Create New Version] ダイアログの [Version] ボックスに表示されます。ユーザーは初期値をオーバーライドできます。

バージョンコンテキスト

バージョンラベルの作成方法を指定します。

[Use Release Name] を選択すると、リリース名と初期バージョン番号を結合してバージョンラベルが作成されます。

ソフトウェアデプロイ方法

コードおよびパッケージコンポーネントのデプロイメントのデフォルト方法を選択します。

新規バージョンを作成した場合、アプリケーションデプロイメントは、指定したファイルまたはパッケージと、必要なスクリプトを含むパッケージを作成します。さらに、次のいずれかのオプションを指定できます。

- Attach Policy as Default – 新規バージョンが作成されたときに、ポリシーも作成します。このバージョンをデプロイするときに、このポリシーを各ターゲットサーバーにアタッチして修復します。
- Install Package as Default – 各ターゲットサーバーに対して、パッケージの「アドホック」インストールを実行します。
- Always Attach Policy – 常にポリシーを作成し、アタッチし、修復します。
- Always Install Package – 常にパッケージの「アドホック」インストールを実行します。

[as Default] オプションの1つを選択した場合、ユーザーはコンポーネントの作成または編集の際にデプロイ方法をオーバーライドできます (「コンポーネント」(469ページ)を参照)。

[Always] オプションの1つを選択した場合、[Locked] インジケーターが表示され、ユーザーはデプロイ方法をオーバーライドできません。

ジャストインタイムターゲットフロー

この設定は、デプロイメント時にジャストインタイムターゲットフローを作成するためにアプリケーションデプロイメントが使用するOperations Orchestration (OO) フローを指定します (「ジャストインタイムターゲット」(531ページ)を参照)。

このフローは、OOライブラリに存在する必要があります。詳細については、「ライブラリ内のアイテムの検索と選択」(504ページ)を参照してください。

この設定は、SAとOOの統合が設定されている場合のみ使用できます。

コードおよび構成ファイルコンポーネントのベースディレクトリ

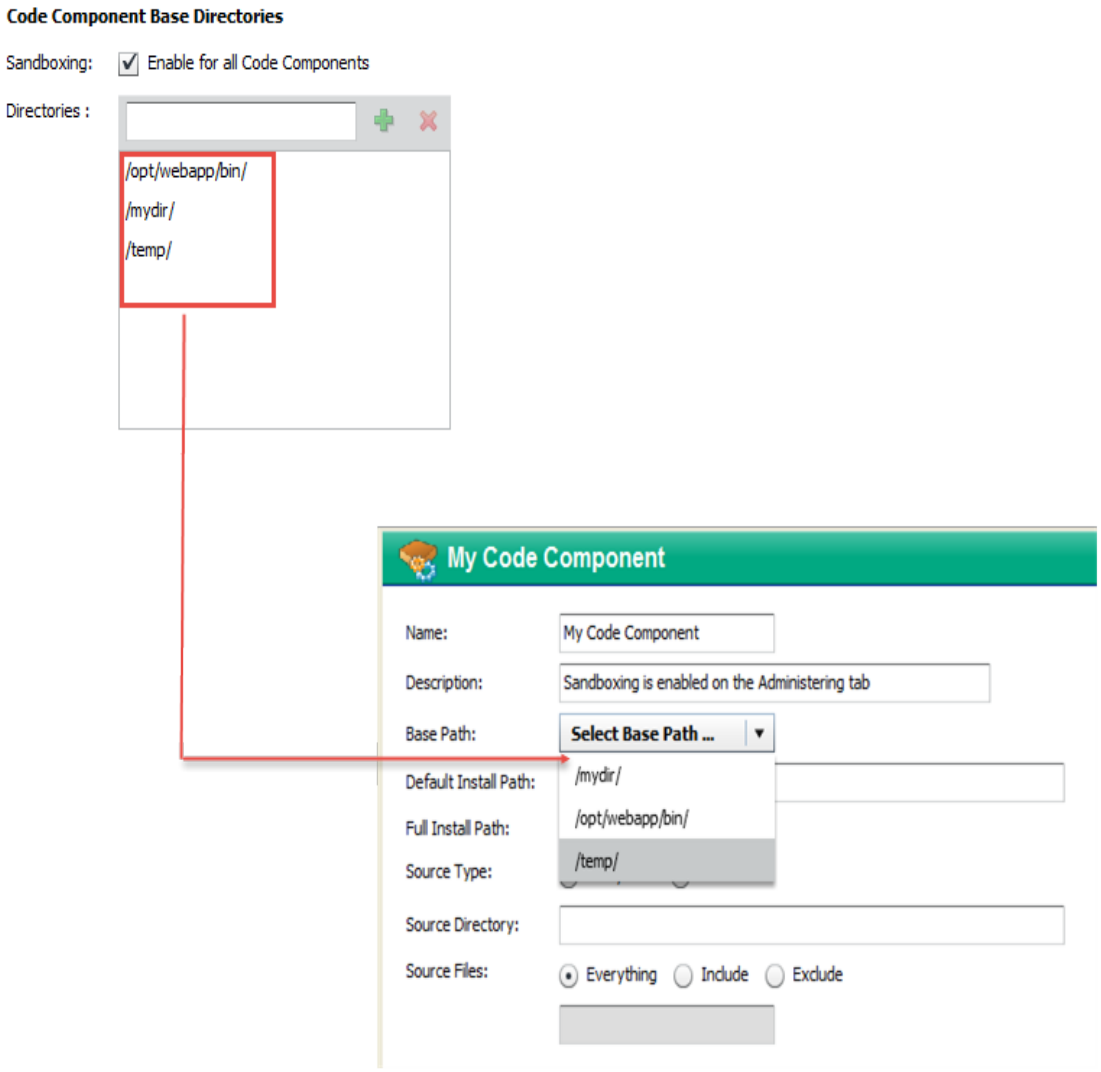
これらの設定を使用すると、コードおよび構成ファイルコンポーネントのデプロイメント用に、独立した「サンドボックス」を作成することができます。これにより、これらのコンポーネントによって作成されるファイルは、制御されたファイルシステムの場所だけにインストールされます。

次の例では、コードコンポーネントについて、サンドボックスメカニズムの動作を説明します。構成ファイルコンポーネントについても仕組みは同じです。

動作の仕組み

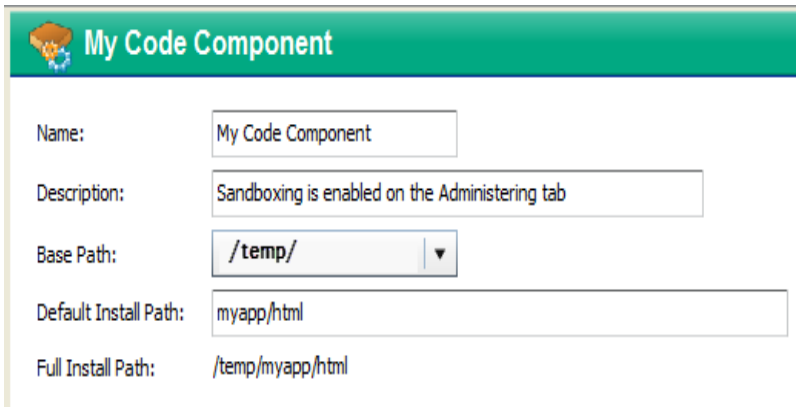
サンドボックスを有効にした場合、コードコンポーネントおよび構成ファイルコンポーネントに対して、リストベースディレクトリ (サンドボックス) を指定できます。これらのディレクトリは、[Applications] タブのコンポーネントプロパティエディターの [Base Path] ドロップダウンリストに表示されます。

ベースディレクトリの例



コードコンポーネントの場合、選択したベースパスと、指定したデフォルトインストールパスを結合することにより、フルインストールパス (ターゲット上でファイルが置かれる場所) が決定されます。

サンドボックスを使用したコードコンポーネント

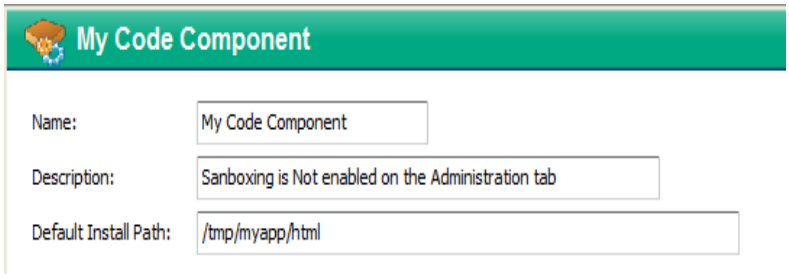


サンドボックスを使用した場合、アプリケーション設計者は、コードまたは構成ファイルコンポーネントを作成する際に、ベースパスを指定する必要があります。ベースパスが指定されていないコードまたは構成ファイルコンポーネントがリリースに含まれる場合、そのリリースのバージョンを作成することはできません (サンドボックスが有効にされている場合)。

既存のリリースのコンポーネントに使用されているベースディレクトリを削除した場合、新しいベースパスをコンポーネントに指定しない限り、そのリリースの新しいバージョンを作成することはできません。ただし、既存のバージョンをデプロイすることは引き続き可能です。

サンドボックスが有効にされていない場合、[Base Path] および [Full Install Path] フィールドは、コードコンポーネントのプロパティエディターに表示されません。

サンドボックスを使用しないコードコンポーネント



設定方法

サンドボックスは、コードコンポーネント、構成ファイルコンポーネント、またはその両方に対して設定できます。

サンドボックスを設定するには、次の手順を実行します。

1. [Administration] 画面の左のペインで、[**Application Settings**] を選択します。
2. 右のペインで、次のオプションのどちらかまたは両方を選択します。
   - **Enable for all Code Components**
   - **Enable for all Configuration File Components**
3. サンドボックス化するコンポーネントタイプのそれぞれについて、次の手順を実行します。
   a. [Directories] テキストボックスに、ベースパスディレクトリを入力します。使用できるのは絶対パスだけです。相対パス (`../webapps`など) は使用できません。
   b. [**Add directory choice**] をクリックします。
   c. 追加するベースディレクトリのそれぞれに対して、ステップ (a) と (b) を繰り返します。
4. [**Save**] をクリックして変更内容を保存します。

既存のベースパスディレクトリを削除するには、次の手順を実行します。

1. [Directories] ボックスで、既存のディレクトリを選択します。
2. [**Remove directory choice**] をクリックします。
3. [**Save**] をクリックして変更内容を保存します。

サンドボックスを有効にした場合、[Directories] リストだけが変更可能です。

グローバルパラメーター

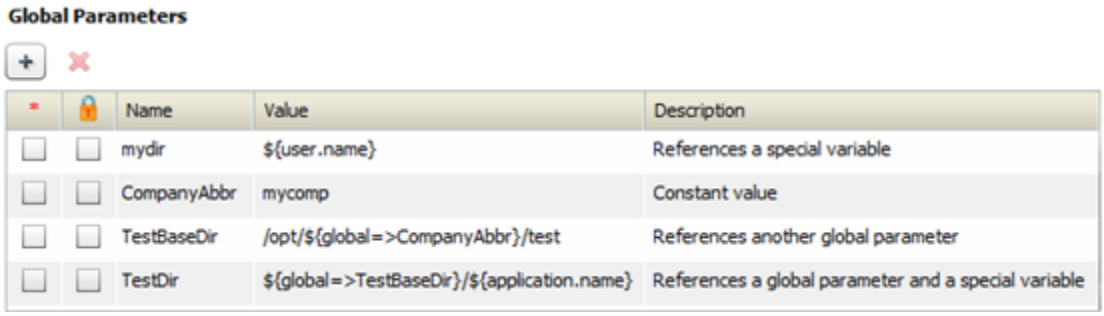
グローバルパラメーターは、すべてのアプリケーションから使用できます。アプリケーションの作成または特定のアプリケーションの編集のためのアクセス権を持つSAユーザーは、コンポーネントパラメーターのデフォルト値またはリリースパラメーターの値を指定する際に、グローバルパラメーターを参照できます (パラメーターと特殊変数については「パラメーターと特殊変数」(506ページ)を参照)。

アプリケーションのデプロイアクセス権と、適切な環境への書き込みアクセス権を持つSAユーザーは、デプロイメント時に、パラメーターエディターで同様の手順を実行することにより、グローバルパラメーターを使用して、デフォルトパラメーターの値をオーバーライドすることができます (「デプロイメントパラメーターのカスタマイズ」(551ページ)を参照)。

グローバルパラメーターを定義または変更するには、次の手順を実行します。

1. [Application Settings] ページで、[Global Parameters] テーブルまで下にスクロールします。
2. 新しいグローバルパラメーターを追加するには、次の手順を実行します。
   a. [**Add New Global Parameter**] をクリックします。
   b. 新しいグローバルパラメーターの名前を入力します。名前はグローバルパラメーターの中で一意である必要があります。名前に次の文字を使用することはできません: @、\、{、}
   c. [**OK**] をクリックします。
3. このパラメーターの [Value] 列のどこかをクリックします。
4. パラメーターの値を指定します (詳細な手順については「パラメーターの値の指定」(515ページ)を参照)。

   値の指定では、特殊変数と他のグローバルパラメーターを参照できます。例:


Global Parameters

*		Name	Value	Description
☐	☐	mydir	${user.name}	References a special variable
☐	☐	CompanyAbbr	mycomp	Constant value
☐	☐	TestBaseDir	/opt/${global=>CompanyAbbr}/test	References another global parameter
☐	☐	TestDir	${global=>TestBaseDir}/${application.name}	References a global parameter and a special variable

5. オプション: パラメーター値を暗号化するには、 (暗号化) 列のチェックボックスをオンにします。

   このコンテキストでは、* (必須) 列は無関係です。
6. ツールバーの [**Save**] をクリックして、変更内容を保存します。

現在他のパラメーターから参照されていないグローバルパラメーターは、削除することができます。

グローバルパラメーターを削除するには、次の手順を実行します。

1. [Global Parameters] テーブルで、削除するパラメーターを選択します。
2. [**Delete Global Parameter**] をクリックします。確認ダイアログが表示されます。
3. [**Yes**] をクリックして削除を確認します。
4. ツールバーの [**Save**] をクリックして、変更内容を保存します。

コンテンツのインポートとエクスポート

このトピックでは、Server Automation (SA) からアプリケーションデプロイメントデータのインポートとエクスポートを行う手順を説明します。次のトピックがあります。

- 「作業」(623ページ)
- 「特別な考慮事項」(624ページ)

作業

アプリケーションデプロイメントデータをXMLファイルにエクスポートするか、XMLファイルからデータをインポートすることができます。これは、異なるSAコアの間でアプリケーションデプロイメントデータをコピーまたは移動する場合、アーカイブ目的でデータをバックアップする場合、またはスクリプトを使用してアプリケーションデプロイメントシナリオを作成する場合に役立ちます。

次のアイテムがインポートおよびエクスポートされます。

- ライフサイクル
- 環境
- アプリケーション
- アプリケーショングループ
- 階層と階層グループ
- リリース
- コンポーネント
- スクリプト
- ソースタイプ

インポートの結果をプレビューして、作成される内容を確認し、既存のアプリケーションデプロイメントデータとの間に競合 (衝突) が発生するかどうかを判定できます。

アプリケーションデプロイメントデータベース全体をインポートまたはエクスポートすることも、特定のアプリケーションをインポートまたはエクスポートすることもできます。特定のアプリケーションをインポートまたはエクスポートする場合、そのアプリケーションに関連付けられたすべてのアイテム、すなわちリリース、ライフサイクル、環境、ソースタイプ、コンポーネント (該当する場合はロールバックおよびアンデプロイスクリプトを含む)、階層が含まれます。

特別な考慮事項

次のアイテムはエクスポートされません。

- リリースバージョン
- デプロイメントジョブ
- ターゲットまたはターゲットグループ
- 環境のデプロイメント前またはデプロイメント後フロー
- アプリケーション設定 ([Administration] 画面)
- アプリケーションのアクセス権
- 環境のアクセス権
- SAライブラリオブジェクト (ソフトウェアポリシー、アプリケーション構成、パッケージ)
- HPE OOフローの内容または定義
- これらのアイテムは、インポートすることもできません。

リリースバージョンはインポートまたはエクスポートできないため、デルタリリース情報をインポートまたはエクスポートする方法はありません。

インポート操作の際に、インポートが実行されているSAコアにSAライブラリアイテムが存在しない場合、そのアイテムはそれを含むオブジェクトには表示されません。たとえば、階層にポリシーが存在し、そのポリシーがそのSAコアに存在しない場合、ポリシーは空白になります。

インポート操作の際に競合が見つかったときに、アイテムをスキップするか上書きするかを指定できます。デフォルトの動作は、アイテムをスキップして、インポートを続行することです。ただし、`--importConflict overwrite`を指定した場合、競合するアイテムは、インポートされたXMLファイルの内容に基づいて更新されます。

コマンドラインツール

アプリケーションデプロイメントには、アプリケーションデプロイメントデータのインポートとエクスポートに使用できるコマンドラインツールが用意されています。構文は次のとおりです。

```
/opt/opsware/da/bin/admtool.sh -opType [--appListTypeappList|appListFile] --
fileTypefilePath [--importConflict overwrite|skip]
```

このツールを使用すると、すべてのアプリケーションまたはその一部をインポートまたはエクスポートできます。

admtool.shのオプション

オプション	説明
opType	操作を指定します。import、export、またはpreviewが使用できます。 短縮形として、-i、-e、または-pが使用できます。
appListType	オプション: 特定のアプリケーションをインポートまたはエクスポートすることと、アプリケーション名を指定する方法を指定します。 importAppList – コマンドラインでアプリケーション名を指定 exportAppList – コマンドラインでアプリケーション名を指定 importAppListFile - CSVファイルでアプリケーション名を指定 exportAppListFile - CSVファイルでアプリケーション名を指定 このオプションを指定しない場合、すべてのアプリケーションがインポートまたはエクスポートされます。
appList	インポートまたはエクスポートされるアプリケーションのリスト。アプリケーションは次のようにセミコロンで区切ります。 app1;app2;app3;...appN importAppListおよびexportAppListオプションと組み合わせて使用します。
appListFile	インポートまたはエクスポートされるアプリケーションのリストを記録したCSVファイル。アプリケーションは次のようにセミコロンで区切ります。 app1;app2;app3;...appN importAppListFileおよびexportAppListFileオプションと組み合わせて使用します。
fileType	インポートまたはプレビュー操作の場合はimportFileを指定します。 エクスポート操作の場合はexportFileを指定します。
filePath	インポート、エクスポート、またはプレビュー対象のXMLファイルのフルパスと名前。 エクスポート操作でファイル名を指定しない場合、デフォルトのエクスポートファイルが使用されます。デフォルトのエクスポートファイルは、/etc/opt/opsware/da/da.confファイルで次のように指定できます。 exportFile=/var/tmp/my_export_default_file da.confファイルでexportFileが指定されておらず、コマンドラインでもエクスポートファイルが指定されていない場合、エクスポートされるデータは次のファイルに書き込まれます。/opt/opsware/da/bin/ADMExport.xml
importConflict	オプション: インポート操作の際に競合が発生したときの動作を指定します。skip

admtool.shのオプション (続き)

オプション	説明
	またはoverwriteが使用できます。競合が発生するのは、インポートXMLファイル内のどれかのアイテムが、アプリケーションデプロイメントデータベースにすでに存在する場合です。 skipを指定すると、データベース内の競合しているアイテムは変更されません。これはデフォルトの動作です。 overwriteを指定すると、競合しているアイテムは、インポートファイルの内容に基づいて更新されます (「上書きによる競合解決」(628ページ)を参照)。 このオプションは、-c overwriteまたは-c skipと短縮できます。

構文の例

次の表に、Application Deployment Managerでサポートされるインポートとエクスポートのシナリオのタイプごとの正しい構文を示します。

プレビューの構文の例

プレビューのシナリオ	例
フルインポートのプレビュー (指定したXMLファイル内のすべてのアプリケーション)	`/opt/opsware/da/bin/admtool.sh -p --importFile myImportData.xml`
競合時に上書きするフルインポートのプレビュー	`/opt/opsware/da/bin/admtool.sh -p --importFile myImportData.xml --importConflict overwrite`
特定アプリケーションの部分的インポートのプレビュー (コマンドラインで指定)	`/opt/opsware/da/bin/admtool.sh -p --importAppList "app1;app2;app3" --importFile myImportData.xml`
競合時に上書きする特定アプリケーションの部分的インポートのプレビュー (コマンドラインで指定)	`/opt/opsware/da/bin/admtool.sh -p --importAppList "app1;app2;app3" --importFile myImportData.xml --importConflict overwrite`
特定アプリケーションの部分的インポートのプレビュー (CSVファイルに指定)	`/opt/opsware/da/bin/admtool.sh -p --importAppListFile app_list.csv --importFile myImportData.xml`
競合時に上書きする特定アプリケーションの部分的インポートのプレビュー (CSVファイルに指定)	`/opt/opsware/da/bin/admtool.sh -p --importAppListFile app_list.csv --importFile myImportData.xml --importConflict overwrite`

詳細については、「プレビューの例」(630ページ)を参照してください。

インポートの構文の例

インポートのシナリオ	例
フルインポート (指定したXMLファイル内のすべてのアプリケーション)	`/opt/opsware/da/bin/admtool.sh -i --importFile myImportData.xml`
競合時に上書きするフルインポート	`/opt/opsware/da/bin/admtool.sh -i --importFile myImportData.xml --importConflict overwrite`
特定アプリケーションの部分的インポート (コマンドラインで指定)	`/opt/opsware/da/bin/admtool.sh -i --importAppList "app1;app2;app3" --importFile myImportData.xml`
競合時に上書きする特定アプリケーションの部分的インポート (コマンドラインで指定)	`/opt/opsware/da/bin/admtool.sh -i --importAppList "app1;app2;app3" --importFile myImportData.xml --importConflict overwrite`
特定アプリケーションの部分的インポート (CSVファイルに指定)	`/opt/opsware/da/bin/admtool.sh -i --importAppListFile app_list.csv --importFile myImportData.xml`
競合時に上書きする特定アプリケーションの部分的インポート (CSVファイルに指定)	`/opt/opsware/da/bin/admtool.sh -i --importAppListFile app_list.csv --importFile myImportData.xml --importConflict overwrite`

詳細については、「インポートの例」(632ページ)を参照してください。

エクスポートの構文の例

エクスポートのシナリオ	例
すべてのアプリケーションデプロイメントデータのフルエクスポート	`/opt/opsware/da/bin/admtool.sh -e --exportFile myExportData.xml`
特定アプリケーションの部分的エクスポート (コマンドラインで指定)	`/opt/opsware/da/bin/admtool.sh -e --exportAppList "app1;app2;app3" --exportFile myExportData.xml`
特定アプリケーションの部分的エクスポート (CSVファイルに指定)	`/opt/opsware/da/bin/admtool.sh -e --exportAppListFile app_list.csv --exportFile myExportData.xml`

詳細については、「エクスポートの例」(635ページ)を参照してください。

上記の例で、`app_list.csv`ファイルには、セミコロン (;) で区切ったアプリケーション名のリストが記録されています。例: `app1;app2;app3;app4`

上書きによる競合解決

競合が発生するのは、インポートXMLファイル内のアイテムが、このSAコア上のアプリケーションデプロイメントデータベースにすでに存在する場合です。デフォルトでは、競合しているアイテムはインポート時にスキップされます。

これに対して、`--importConflict overwrite`オプションを指定した場合は、競合しているアイテムは変更されます。特定のリリースに関連付けられている一部のアイテムが削除される場合もあります。

次の表に、`--importConflict overwrite`を指定した場合のインポート操作の結果の一覧を示します。

競合解決のシナリオ

アイテムのタイプ	インポートファイルに存在 データベースに存在しない	インポートファイルに存在 データベースに存在	インポートファイルに存在しない データベースに存在
アプリケーション アプリケーショングループ 特定アプリケーションに関連付けられたリリース 環境 ライフサイクル 階層 階層グループ スクリプト ソースタイプ	アイテムは作成される	アイテムはインポートファイルの内容に基づいて更新される	アイテムは変更されないままデータベースに残る
特定リリースに関連付けられた階層 特定リリースに関連付けられた階層に関連付けられたコンポーネント	アイテムは作成される	アイテムはインポートファイルの内容に基づいて更新される	アイテムは削除される

インポートの競合に関連するメッセージの例については、「重要なメッセージ」(629ページ)の下の「データの競合に関するメッセージ」(629ページ)を参照してください。

ログファイル

admtool.shツールは、詳細なログファイルを生成します。ファイル名は、実行される操作によって異なります。

/var/log/opsware/da/importData.log

/var/log/opsware/da/exportData.log

/var/log/opsware/da/previewData.log

重要なメッセージ

プレビュー操作の場合、次のタイプのメッセージ (stdoutに出力) は、このインポートが実行されると何か新しいものが作成されることを示します。

```
Import: 'CodeComponentSourceType' 'QA' will be created.
```

次のタイプのメッセージは、インポートまたはエクスポートされるアプリケーションのリストに含まれていないため、アプリケーションがスキップされることを示します。

```
Skip: Application 'AppD' due to Application Filter.
```

インポート操作の場合、次のタイプのメッセージ (stdoutに出力) は、ライブラリアイテム (ソフトウェアポリシー、パッケージ、アプリケーション構成など) が、インポート先のSAコアに存在しないことを示します。

```
Not Found:Package'OPSWpytwist2-40.0.0.7.325.zip(SunOS 5.8)'does not exist
```

次のタイプのメッセージは、インポート先のSAコアにある既存のアプリケーションデプロイメントデータと、インポートXMLファイル内のデータとの間に、競合が存在するかどうかを示します。

データの競合に関するメッセージ

モード	競合解決	競合検出	サンプルメッセージ
プレビュー	上書き	あり	`Conflict: Release 'App1:Release1' will be updated.`
プレビュー	スキップ	あり	Conflict: Release 'App1:Release1' already exists. will skip.
プレビュー	上書きまたはスキップ	なし	`Import: Release 'App1:Release1' will be created.`

データの競合に関するメッセージ (続き)

モード	競合解決	競合検出	サンプルメッセージ
インポート	上書き	あり	`Updated Script 'Windows Code Rollback' based on importConflict 'overwrite' option.`
インポート	上書き	あり	`Deleted Component 'Script 1 (Sample Application:Q3 Release:Tomcat 6.0.20)' based on importConflict 'overwrite' option.`
インポート	スキップ	あり	`Lifecycle 'Standard' already exists.Will not import based on importConflict 'skip' option.`
インポート	上書きまたはスキップ	なし	`Created Lifecycle 'Complete'.`

プレビューの例

次の単純化した例は、インポートおよびエクスポート操作のフルと部分的の違いを示すためのものです。XMLファイルにはアプリケーション定義だけが含まれます。代表的なファイルには、アプリケーショングループ、リリース、ライフサイクル、環境、コンポーネント、スクリプト、階層、階層グループ、ソースタイプの情報も含まれます。

プレビュー (`-p`) 操作を行うと、XMLファイル内の情報が正常にインポートされるかどうか、競合が存在するかどうかを確認することができます。

この例では、インポートファイルに4つのアプリケーションが含まれています。MyAppA、MyAppB、MyAppC、MyAppDです。MyAppAアプリケーションは、このSAコアにすでに存在します。

このXMLインポートファイル (`myImportData.xml`) の内容を次に示します。

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<DAData xmlns="http://www.hp.com/DA">
       <ModelVersion>1.3</ModelVersion>
       <ReleaseVersion>09.003.001</ReleaseVersion>
       <SCM_VERSION>0</SCM_VERSION>
       <Applications>
              <Application>
                     <Name>MyAppA</Name>
                     <Description>This application exists</Description>
```

```
                </Application>
        <Application>
                <Name>MyAppB</Name>
                <Description>This is a new application - no conflict</Description>
        </Application>
        <Application>
                <Name>MyAppC</Name>
                <Description>This is a new application - no conflict</Description>
        </Application>
        <Application>
                <Name>MyAppD</Name>
                <Description>This is a new application - no conflict</Description>
        </Application>
        </Applications>
</DAData>
```

例 1: フルインポートのプレビュー

次のコマンドは、myImportData.xmlファイル内のすべてのアプリケーションデプロイメント情報のインポートをプレビューします。

```
/opt/opsware/da/bin/admtool.sh -p --importFile /tmp/myImportData.xml
```

プレビューを実行すると、次のメッセージがstdoutに表示されます。

```
Importer - ========== Begin Previewing with importConflict option 'skip' ...
AbstractBuilder - Conflict: Application 'MyAppA' already exists. Will skip.
AbstractBuilder - Import: Application 'MyAppB' will be created.
AbstractBuilder - Import: Application 'MyAppC' will be created.
AbstractBuilder - Import: Application 'MyAppD' will be created.
Importer - Previewing DA data from '/tmp/myImportData.xml' has been completed
ImportExport - ========== End Previewing ==========
```

この例では、MyAppAはすでにこのSAコアに存在するためインポートされず、デフォルトの競合解決方法(スキップ)が適用されます。

上書きによる競合解決を使用するには、次のコマンドを代わりに使用します。

```
/opt/opsware/da/bin/admtool.sh -p --importFile /tmp/myImportData.xml --
importConflict overwrite
```

この場合、次のメッセージがstdoutに表示されます。

```
Importer - ========== Begin Previewing with importConflict option 'overwrite'
AbstractBuilder - Conflict: Application 'MyAppA' will be updated.
AbstractBuilder - Import: Application 'MyAppB' will be created.
AbstractBuilder - Import: Application 'MyAppC' will be created.
AbstractBuilder - Import: Application 'MyAppD' will be created.
Importer - Previewing DA data from '/tmp/myImportData.xml' has been completed
ImportExport - ========== End Previewing ==========
```

例2: 部分的インポートのプレビュー

次のコマンドは、MyAppAとMyAppBのインポートをプレビューします。

```
/opt/opsware/da/bin/admtool.sh -p --importAppList "MyAppA;MyAppB" --importFile
/tmp/myImportData.xml
```

```
Importer - ========== Begin Previewing with importConflict option 'skip' ...
Importer - Applying Application Filter [MyAppB, MyAppA]
AbstractBuilder - Conflict: Application 'MyAppA' already exists. Will skip.
AbstractBuilder - Import: Application 'MyAppB' will be created.
AbstractBuilder - Skip: Application 'MyAppC' due to Application Filter.
AbstractBuilder - Skip: Application 'MyAppD' due to Application Filter.
Importer - Previewing DA data from '/tmp/myImportData.xml' has been completed
ImportExport - ========== End Previewing ==========
```

この例では、MyAppAはすでにこのSAコアに存在するためインポートされず、デフォルト (スキップ) の競合解決方法が適用されます。MyAppCとMyAppDは、`--importAppList`引数に含まれていないため、インポートされません。

上書きによる競合解決を使用してこのインポートをプレビューするには、次のコマンドを使用します。

```
/opt/opsware/da/bin/admtool.sh -p --importAppList "MyAppA;MyAppB" --importFile
/tmp/myImportData.xml --importConflict overwrite
```

この場合、次のメッセージが`stdout`に表示されます。

```
Importer - ========== Begin Previewing with importConflict option 'overwrite'
Importer - Applying Application Filter [MyAppB, MyAppA]
AbstractBuilder - Conflict: Application 'MyAppA' will be updated.
AbstractBuilder - Import: Application 'MyAppB' will be created.
AbstractBuilder - Skip: Application 'MyAppC' due to Application Filter.
AbstractBuilder - Skip: Application 'MyAppD' due to Application Filter.
Importer - Previewing DA data from '/tmp/myImportData.xml' has been completed
ImportExport - ========== End Previewing ==========
```

インポートの例

前の例に続けて、同じXMLファイル (`myImportData.xml`) の内容をインポートします。

例 1: フルインポート

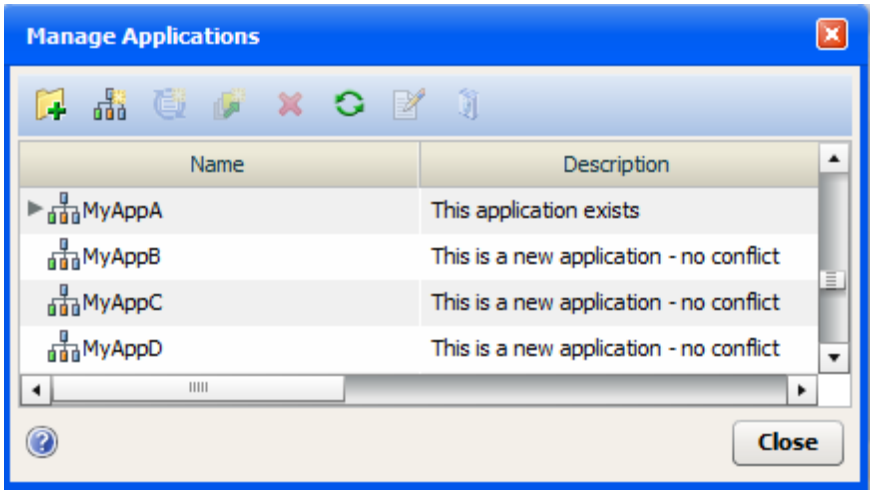
次のコマンドは、myImportData.xmlファイル内のすべてのアプリケーションデプロイメント情報を、デフォルトの競合解決方法 (スキップ) でインポートします。

```
/opt/opsware/da/bin/admtool.sh -i --importFile /tmp/myImportData.xml
```

MyAppAはすでに存在するためスキップされ、その他のアプリケーション (MyAppB、MyAppC、MyAppD) は作成されるはずです。これは次のメッセージによって確認されます。

```
Importer - ========== Begin Importing with importConflict option 'skip' ...
AbstractBuilder - Application 'MyAppA' already exists. Will not import based
on importConflict 'skip' option.
AbstractBuilder - Created Application '<top>.MyAppB'.
AbstractBuilder - Created Application '<top>.MyAppC'.
AbstractBuilder - Created Application '<top>.MyAppD'.
Importer - Importing DA data from '/tmp/myImportData.xml' has been completed
ImportExport - ========== End Importing ==========
```

インポート操作の後、4つのアプリケーションがすべて [Manage Applications] ダイアログに表示されます。



次の点に注意してください。

無効なXMLを含むファイルからデータをインポートしようとすると、インポートは実行されません。

ソフトウェアポリシー、パッケージ、またはアプリケーション構成コンポーネントをインポートする場合、関連するライブラリアイテムが存在しないと、次のようなメッセージが表示されます。

```
Not Found:Package'OPSWpytwist2-40.0.0.7.325.zip(SunOS 5.8)'does not exist
```

この場合、別のライブラリアイテムを選択するか、存在しないアイテムをこのコア上のライブラリに追加して、そのアイテムを選択してからでないと、バージョンを作成することはできません。

上書きによる競合解決を使用してこのインポートを実行するには、次のコマンドを使用します。

```
/opt/opsware/da/bin/admtool.sh -i --importFile /tmp/myImportData.xml --
importConflict overwrite
```

この場合、次のメッセージがstdoutに表示されます。

```
Importer - ========== Begin Importing with importConflict option 'overwrite'
AbstractBuilder - Updated Application '<top>.MyAppA' based on importConflict
'overwrite'
AbstractBuilder - Created Application '<top>.MyAppB'.
AbstractBuilder - Created Application '<top>.MyAppC'.
AbstractBuilder - Created Application '<top>.MyAppD'.
Importer - Importing DA data from '/tmp/myImportData.xml' has been completed
ImportExport - ========== End Importing ==========
```

例2: 部分的インポート

MyAppAだけが存在するとした場合、次のコマンドはMyAppBだけをmyImportData.xmlファイルからインポートします。

```
/opt/opsware/da/bin/admtool.sh -i --importAppList "MyAppA;MyAppB" --importFile
/tmp/myImportData.xml
```

結果は次のようになるはずです。

MyAppAは、すでにこのSAコアに存在するため、スキップされます。

MyAppBは作成されます。

MyAppCとMyAppDは、--importAppList引数に含まれていないため、作成されません。

この結果は、次のメッセージによって確認されます。

```
Importer - ========== Begin Importing with importConflict option 'skip' ... =======
Importer - Applying Application Filter [MyAppB, MyAppA]
AbstractBuilder - Application 'MyAppA' already exists. Will not import based on
importConflict 'skip' option.
AbstractBuilder - Created Application '<top>.MyAppB'.
AbstractBuilder - Skipped import of Application 'MyAppC' due to Application Filter.
AbstractBuilder - Skipped import of Application 'MyAppD' due to Application Filter.
Importer - Importing DA data from '/tmp/myImportData.xml' has been completed
ImportExport - ========== End Importing ==========
```

上書きによる競合解決を使用してこのインポートを実行するには、次のコマンドを使用します。

```
/opt/opsware/da/bin/admtool.sh -i --importAppList "MyAppA;MyAppB" --importFile
/tmp/myImportData.xml --importConflict overwrite
```

この場合、次のメッセージがstdoutに表示されます。

```
Importer - ========== Begin Importing with importConflict option 'overwrite' ...
Importer - Applying Application Filter [MyAppB, MyAppA]
AbstractBuilder - Updated Application '<top>.MyAppA' based on importConflict
'overwrite' option.
AbstractBuilder - Created Application '<top>.MyAppB'.
AbstractBuilder - Skipped import of Application 'MyAppC' due to Application Filter.
AbstractBuilder - Skipped import of Application 'MyAppD' due to Application Filter.
Importer - Importing DA data from '/tmp/myImportData.xml' has been completed
ImportExport - ========== End Importing ==========
```

エクスポートの例

エクスポート操作には2種類あります。フルエクスポートは、このSAコアに存在するすべてのアプリケーションデプロイメントデータを、指定したXMLファイルに書き込みます。部分的エクスポートは、指定したアプリケーションに関連する情報だけを書き込みます。

エクスポートできるアイテムの一覧については、「作業」(623ページ)を参照してください。

エクスポートできないアイテムの一覧については、「特別な考慮事項」(624ページ)を参照してください。

例1: フルエクスポート

次のコマンドは、このSAコアのすべてのアプリケーションデプロイメントをエクスポートします。

```
/opt/opsware/da/bin/admtool.sh -e --exportFile /tmp/myFullExportData.xml
```

例2: 部分的エクスポート

次のコマンドは、MyAppAとMyAppBだけをエクスポートします。

```
/opt/opsware/da/bin/admtool.sh -e --exportAppList "MyAppA;MyAppB" --exportFile
/tmp/myPartialExportData.xml
```

前回のインポートの後でMyAppAまたはMyAppBが変更されていないと仮定すると、エクスポート後の`myPartialExportData.xml`ファイルの`<Applications>`部分は次のようになります。

`myPartialExportData.xml`ファイルの`<Applications>`部分は次のようになります。

```
       <Applications>
              <Application>
                     <Name>MyAppB</Name>
                     <Description>This is a new application - no conflict</Description>
       </Application>
       <Application>
```

```
                    <Name>MyAppA</Name>
                    <Description>This application exists</Description>
                </Application>
</Applications>
```

ドキュメントのフィードバックを送信

本ドキュメントについてのご意見、ご感想については、電子メールでドキュメント制作チームまでご連絡ください。このシステムに電子メールクライアントが設定されている場合は、上記のリンクをクリックすると、次の情報が件名行に記載された電子メールウィンドウが開きます。

フィードバック: 開発者ガイド (Server Automation 10.50)

フィードバックを追加して [送信] をクリックしてください。

電子メールクライアントが使用できない場合は、Webメールクライアントのメッセージに上記の情報をコピーし、hpe_sa_docs@hpe.com までフィードバックをお送りください。

ご協力をお願いいたします。