HP Server Automation

Ultimate Edition

ソフトウェアバージョン: 10.10

ユーザーガイド: アプリケーション構成

ドキュメントリリース日: 2014年6月30日 (英語版)
ソフトウェアリリース日: 2014年6月30日 (英語版)

ご注意

保証

HP製品、またはサービスの保証は、当該製品、およびサービスに付随する明示的な保証文によってのみ規定されるものとします。ここでの記載は、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、HPはいかなる責任も負いません。

ここに記載する情報は、予告なしに変更されることがあります。

権利の制限

機密性のあるコンピュータ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、HP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。

Intel®およびItanium®は、Intel Coporationの米国およびその他の国における登録商標です。

Microsoft®、Windows®、およびWindows® XPは，Microsoft Corporationの米国における登録商標です。

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

UNIX®は、The Open Groupの登録商標です。

サポート

次のHPソフトウェアサポートオンラインのWebサイトを参照してください。

http://support.openview.hp.com

このサイトでは、HPのお客様窓口のほか、HP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。

HPソフトウェアオンラインではセルフソルブ機能を提供しています。お客様の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。HP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
- HPサポート窓口の検索
- 利用可能なサービスに関する情報の閲覧
- 他のソフトウェアカスタマーとの意見交換
- ソフトウェアトレーニングの検索と登録

一部のサポートを除き、サポートのご利用には、HP 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。HP Passport IDを登録するには、次のWebサイトにアクセスしてください。

http://h20229.www2.hp.com/passport-registration.html

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。

http://support.openview.hp.com/access_level.jsp

サポートマトリクス

サポートおよび互換性情報については、関連する製品リリースのサポートマトリクスを参照してください。サポートマトリクスと製品マニュアルは、次のHPソフトウェアサポートオンラインのWebサイトで参照できます。

http://h20230.www2.hp.com/sc/support_matrices.jsp

また、本リリースの『HP Server Automation Support and Compatibility Matrix』は、次のHPソフトウェアサポートオンラインの製品マニュアルWebサイトからダウンロードできます。

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

ドキュメントの更新情報

このリリースのServer Automation製品の最新のドキュメントは、すべて次のSA Documentation Libraryから入手できます。

http://support.openview.hp.com/selfsolve/document/KM00417675/binary/SA_10_docLibrary.html

SA Documentation Libraryでは、このリリースに関連するガイドライン、リリースノード、サポートマトリクス、およびホワイトペーパーにアクセスできます。また、フルドキュメントセットを一括してダウンロードすることもできます。SA Documentation Libraryは、リリースごとに更新されます。また、リリースノートが更新されたときや、新しいホワイトペーパーが発行されたときにも更新されます。

情報リソースを見つける方法

Server Automationの情報リソースは、次のいずれの方法でもアクセスできます。

方法1: 新しいSA Documentation Libraryから、最新のドキュメントにタイトルとバージョンを指定してアクセスします。

方法2: [All Manuals Download] からローカルディレクトリにフルドキュメントメントセットを保存します。

方法3: サポートされるリリースのHP製品ドキュメントをHPソフトウェアドキュメントポータルで検索します。

各ドキュメントにアクセスするには、次の手順を実行します。

1. SA 10.x Documentation Libraryにアクセスします。

   http://support.openview.hp.com/selfsolve/document/KM00417675/binary/SA_10_docLibrary.html

2. HP Passportの資格情報を使ってログインします。
3. ドキュメントのタイトルとバージョンを指定して、[**go**] をクリックします。

ローカルディレクトリ内の完全なドキュメントセットを使用するには、次の手順を実行します。

1. フルドキュメントセットをローカルディレクトリにダウンロードするには、次の手順を実行します。
   a. SA Documentation Libraryにアクセスします。

      http://support.openview.hp.com/selfsolve/document/KM00417675/binary/SA_10_docLibrary.html

b HP Passportの資格情報を使ってログインします。

c SA 10.1バージョンの [All Manuals Download] タイトルを探します。

d [**go**] リンクをクリックして、ローカルディレクトリにZIPファイルをダウンロードします。

e ファイルを解凍します。

2 ローカルディレクトリ内のドキュメントを探すには、ドキュメントカタログ (docCatalog.html) を使用します。ローカルディレクトリにダウンロードしたドキュメントの索引ポータルが表示されます。

3 ドキュメントセット内のすべてのドキュメントを対象としてキーワードを検索するには、次の手順を実行します。

a ローカルディレクトリ内の任意のPDFドキュメントを開きます。

b [**編集**] > [**高度な検索**] を選択します (またはShift＋Ctrl＋Fキー)。

c [以下の場所にあるすべてのPDF文書] オプションを選択し、ローカルディレクトリを指定します。

d キーワードを入力し、[検索] をクリックします。

HPソフトウェアドキュメントポータルで追加ドキュメントを探すには、次の手順を実行します。

HPソフトウェアドキュメントポータルにアクセスします。

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

このサイトを利用するには、HP Passportへの登録とサインインが必要です。HP Passport IDの登録は、HP Passportのサインインページの [**New users - please register**] リンク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、HPの営業担当にお問い合わせください。改訂状況については、「ドキュメントの更新情報」を参照してください。

製品エディション

Server Automationには、次の2つの製品エディションがあります。

- Server Automation (SA) は、Server AutomationのUltimate Editionです。Server Automationについては、『SAリリースノート』および『SAユーザーガイド: Server Automation』を参照してください。
- Server Automation Virtual Appliance (SAVA) は、Server Automation の Premium Edition です。SAVA の機能については、『SAVA Release Notes』および『SAVAクイックガイド』を参照してください。

ドキュメントの更新情報

次の表は、前回リリースされたエディション以降の本ドキュメントに対する変更を示します。

日付	変更内容
2014年7月9日 (英語版)	SA 10.1に伴う本ドキュメントのオリジナルリリース。

目次

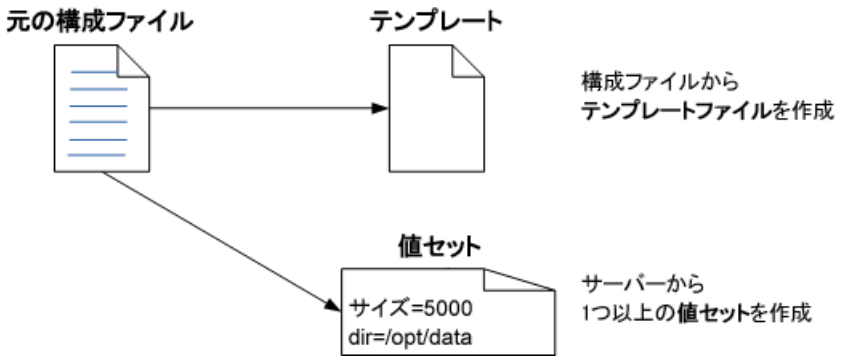
第1章 アプリケーション構成のクイックスタート

この章では、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 (15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 (49ページ) を参照してください。

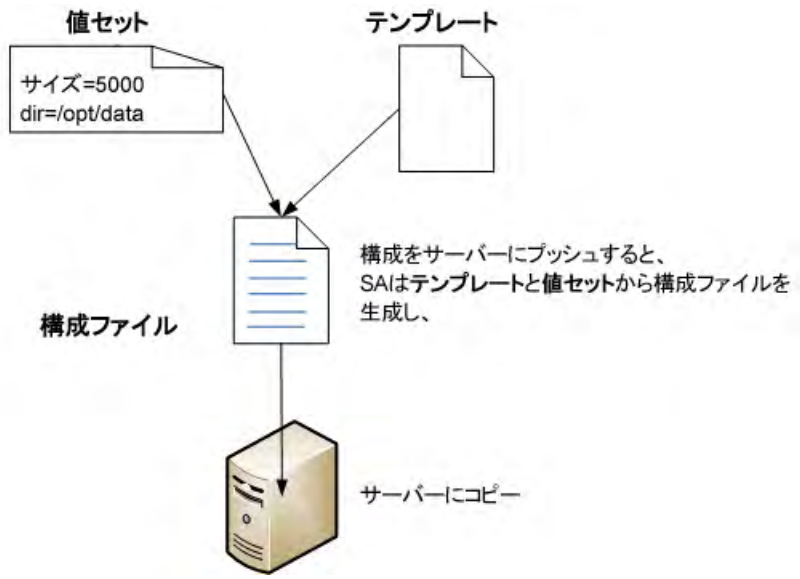
アプリケーション構成を作成して使用するには、まず次の図に示すように**テンプレート**ファイルと**値セット**を作成する必要があります。テンプレートは、構成ファイルのモデルです。値セットは、サーバーにプッシュされる構成ファイルのインスタンスを作成するために、テンプレートとマージされるデータ値です。手順については後で詳しく説明します。

図1　　アプリケーション構成の作成の図



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。手順については後で詳しく説明します。

図2　　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には、次の手順を実行します。

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。参照情報:

   — 構成テンプレートの作成 (17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 (18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 (73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 (139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 (107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について (50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 (34ページ)

   — 構成テンプレートとスクリプトテンプレートについて (50ページ).

   — アプリケーション構成でのスクリプトの実行について (63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 (32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 (16ページ)
- アプリケーション構成オブジェクトについて (49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。参照情報:

- 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 (33ページ)

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。参照情報:

- 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 (36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の使用 (71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。参照情報:

- 既存の構成ファイルからの値セットのインポート (57ページ)
- 値セットについて (51ページ)
- 値セットのレベルと値セットの継承 (52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。参照情報:

- 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 (44ページ)
- 2つの構成テンプレートの比較 (45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。参照情報:

- アプリケーション構成のプッシュ (39ページ)
- アプリケーション構成のサーバーへのプッシュについて (65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 (66ページ) と『SAユーザーガイド: 監査とコンプライアンス』を参照してください。

第2章 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テンプレートの編集と管理:

アプリケーション構成でのスクリプトの使用:

アプリケーション構成のアタッチとデタッチ:

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について (49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。

   a [構成] ノードを開きます。

   b 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。

   c 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 (46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について (50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 (34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 (34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 (46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 (71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 (31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 (139ページ) およびXML構成ファイルの管理 (73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。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 (33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

  詳細については、使用中の構成ファイルからのテンプレートの作成 (19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

  詳細については、テンプレートのインポートと作成 (18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。

  詳細については、ビジュアルエディターによる構成テンプレートの編集 (24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 (27ページ) を参照してください。

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、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。

[**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ます。

2 [**アクション**] メニューで、[**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 (21 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 (24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。

— [**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。

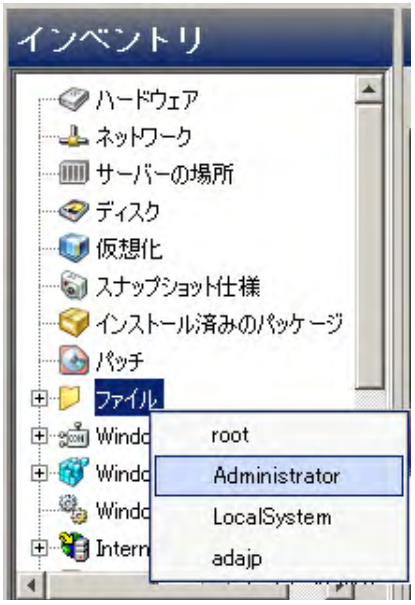
— [**デバイス**] > [**デバイスグループ**] を選択して、デバイスグループリストを表示します。

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。

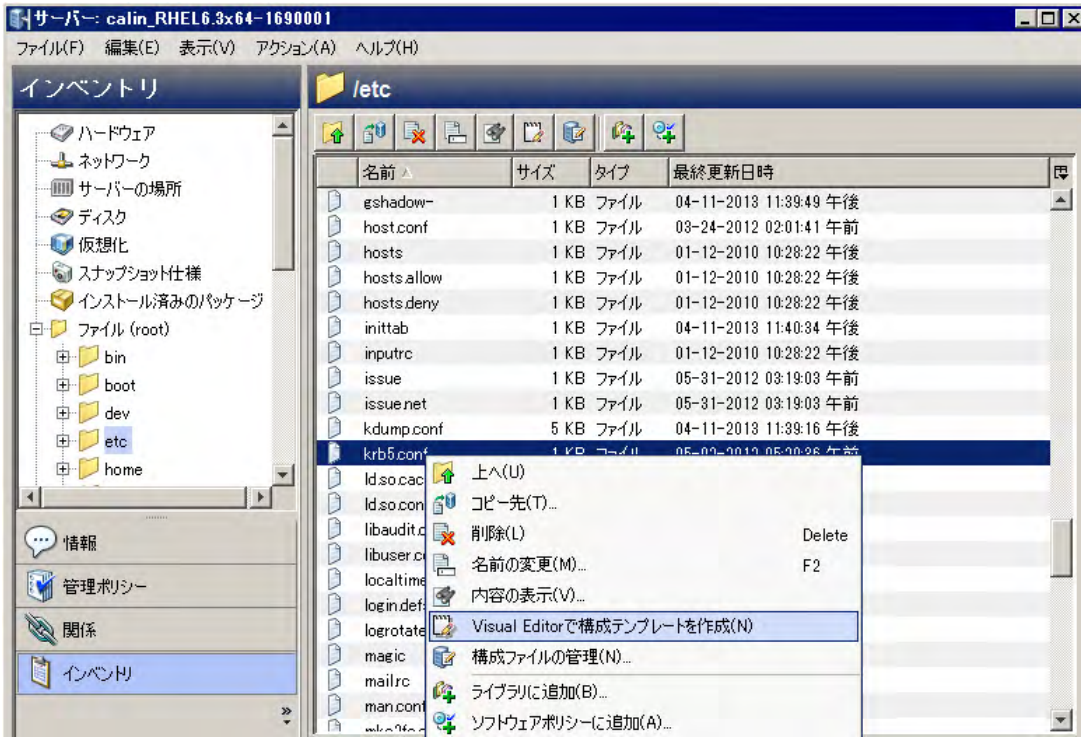
c [**アクション**] メニューから [**開く**] を選択します。

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。

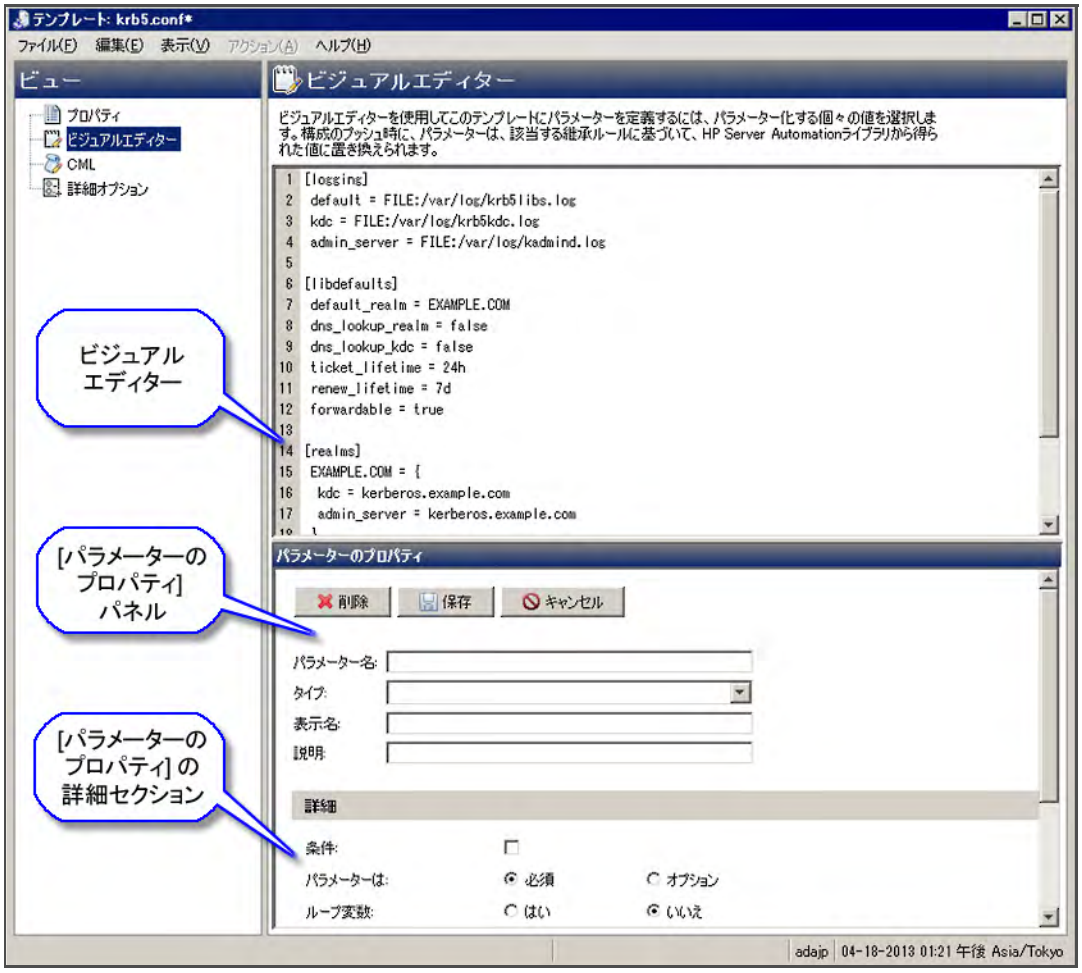


選択した構成ファイルのテンプレートがビジュアルエディターモードで開きます。詳細については、ビジュアルエディターのインタフェース (21ページ) を参照してください。

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。

図3　　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

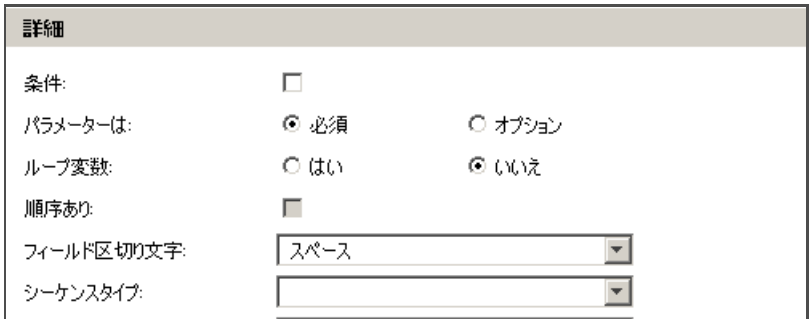
ビジュアルエディターで値を選択してから、[パラメーターのプロパティ]パネルの[**パラメーター名**]フィールドの中をクリックすると、[パラメーターのプロパティ]パネルのフィールドが、元になる構成ファイルの値に基づいて設定されます。

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。

図4　　[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)。

図5　　[パラメーターのプロパティ] フォームの [詳細] セクション

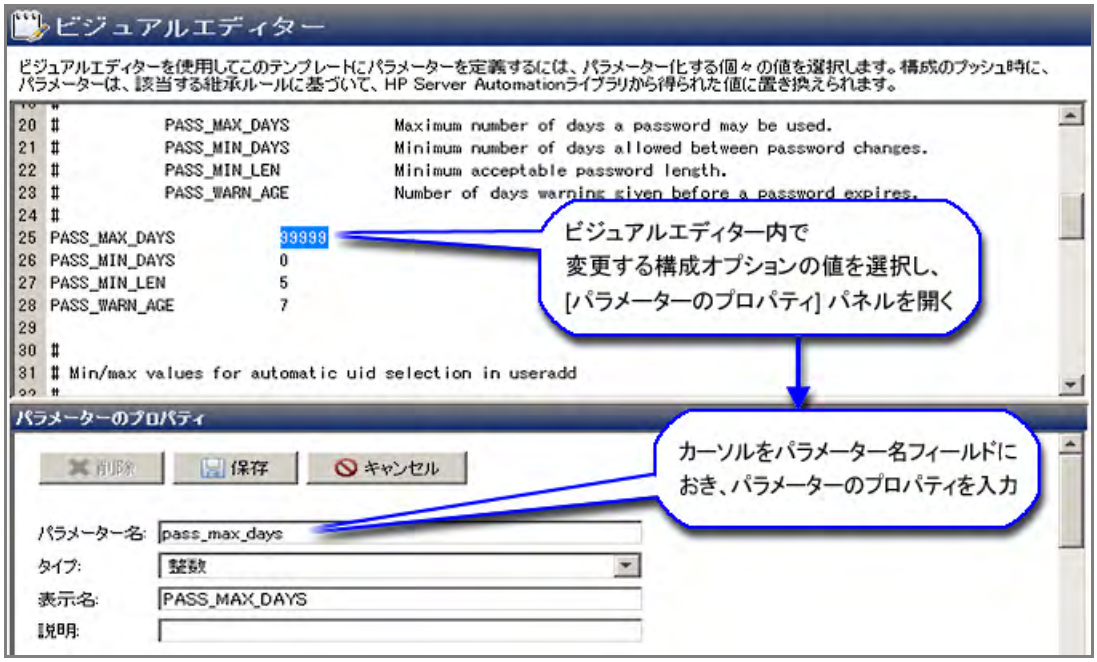


ビジュアルエディターインタフェースの仕組み:

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。

図6　　ビジュアルエディターでの構成テンプレートの編集

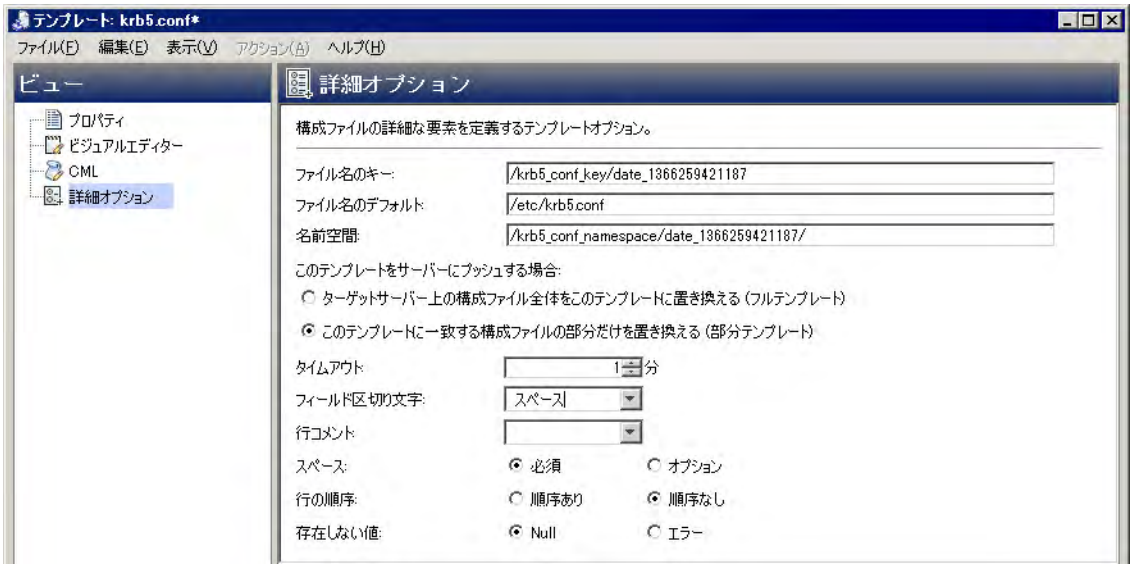


ビジュアルエディターによる構成テンプレートの編集方法については、ビジュアルエディターによる構成テンプレートの編集 (24ページ) を参照してください。

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。

図7　　ビジュアルエディターの詳細オプション



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

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 (18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 (18 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 (19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。

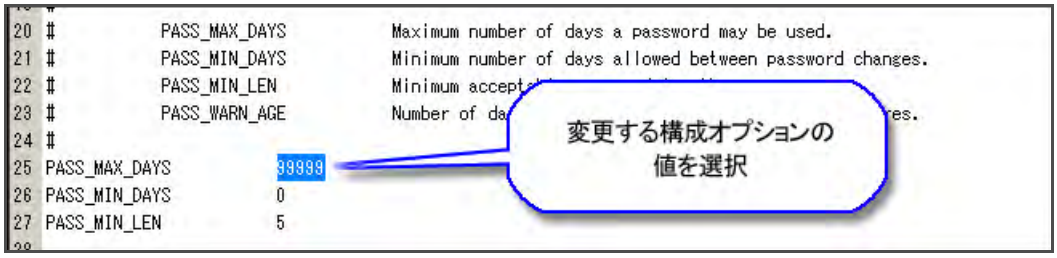
   例については、単純なパラメーターの作成 (24ページ) またはシーケンスパラメーターの作成 (26ページ) を参照してください。

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。

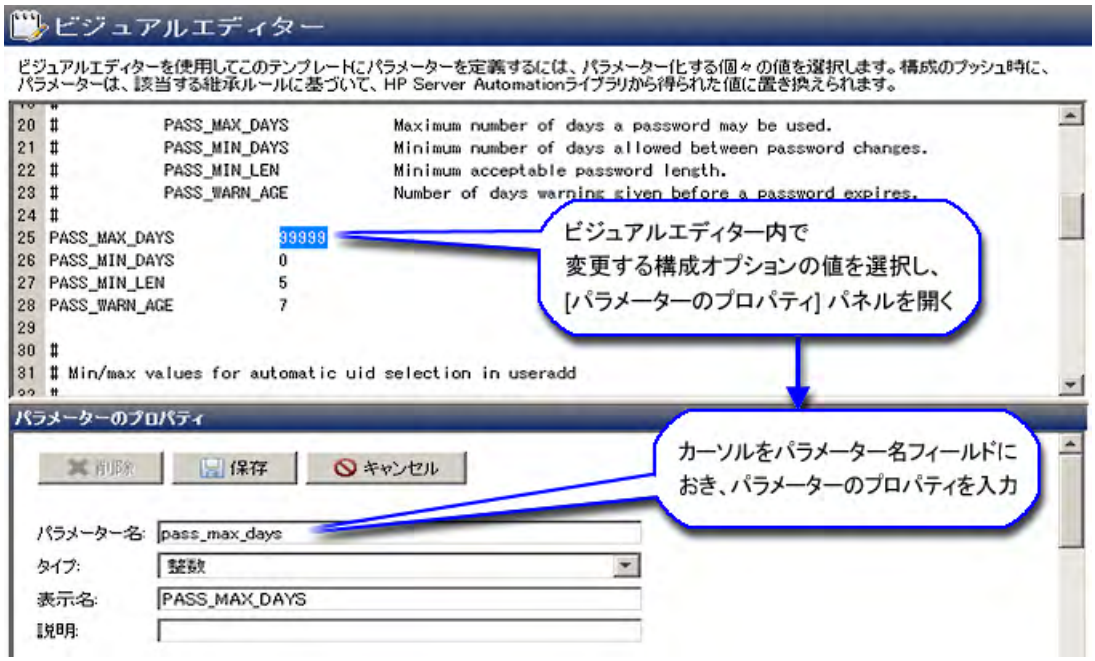
単純なパラメーターの作成

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。

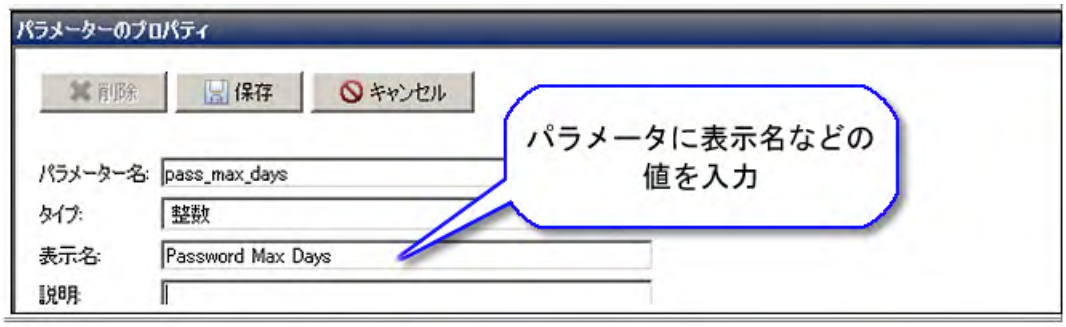
1. 構成をプッシュする際に置き換える構成オプションを選択すると、フォームビューが表示されます。



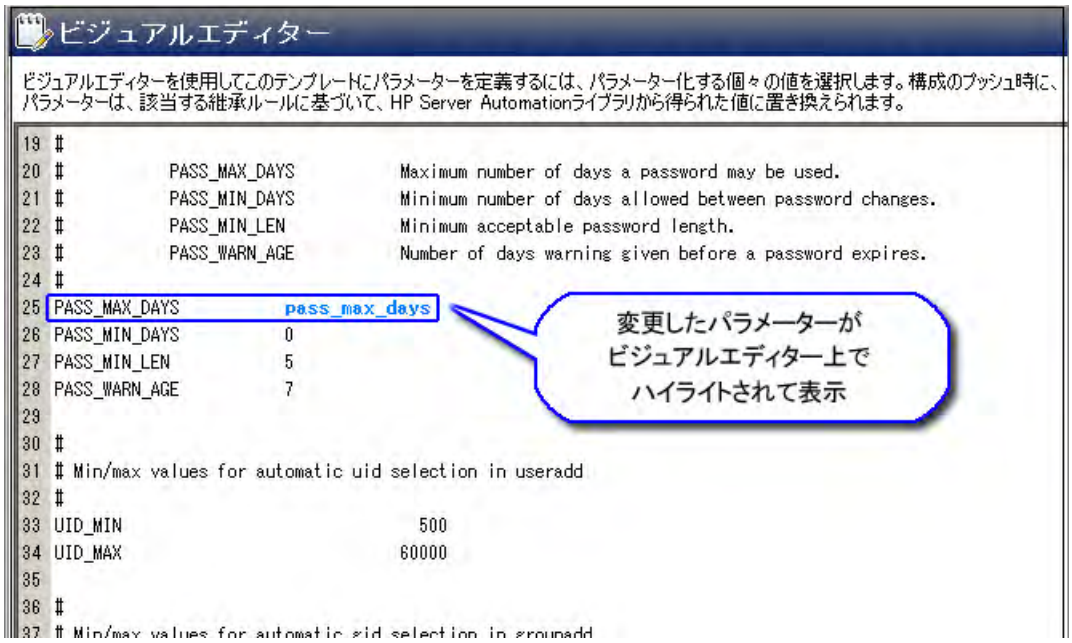
2. [パラメーターのプロパティ] パネルで、[**パラメーター名**] フィールドをクリックすると、マークの付いたオプションで、[**パラメーター名**]、[**表示名**]、[**データ型**] の各フィールドが、元になるテンプレートに基づいて設定されます。



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。

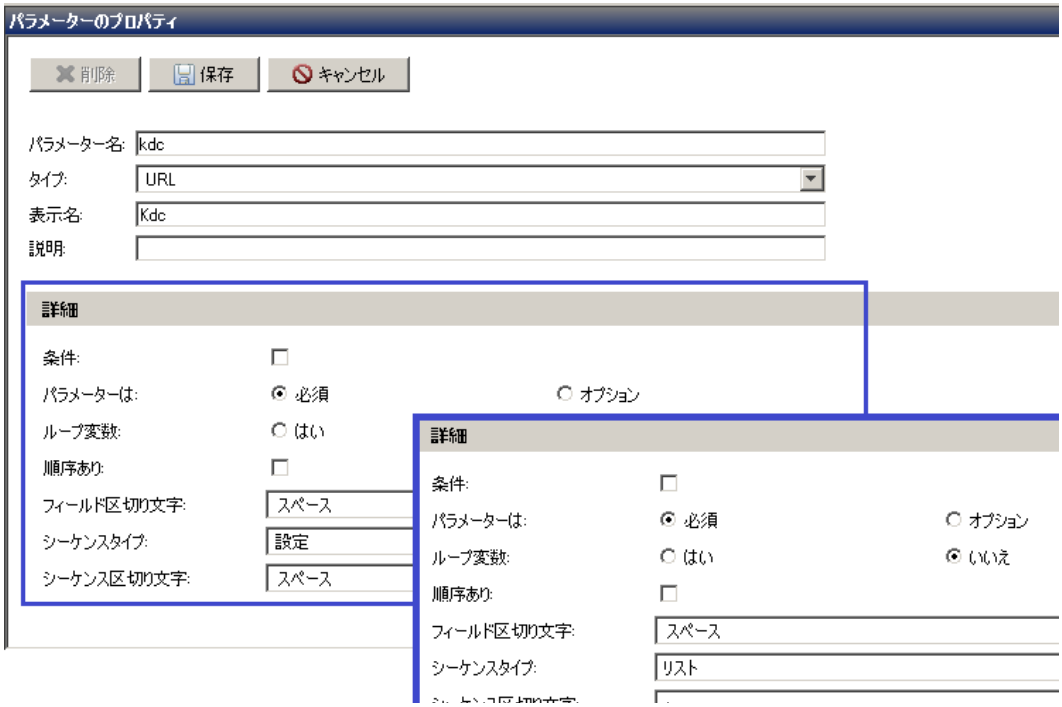


シーケンスパラメーターの作成

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 (24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。

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。

— [**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。

— [**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。

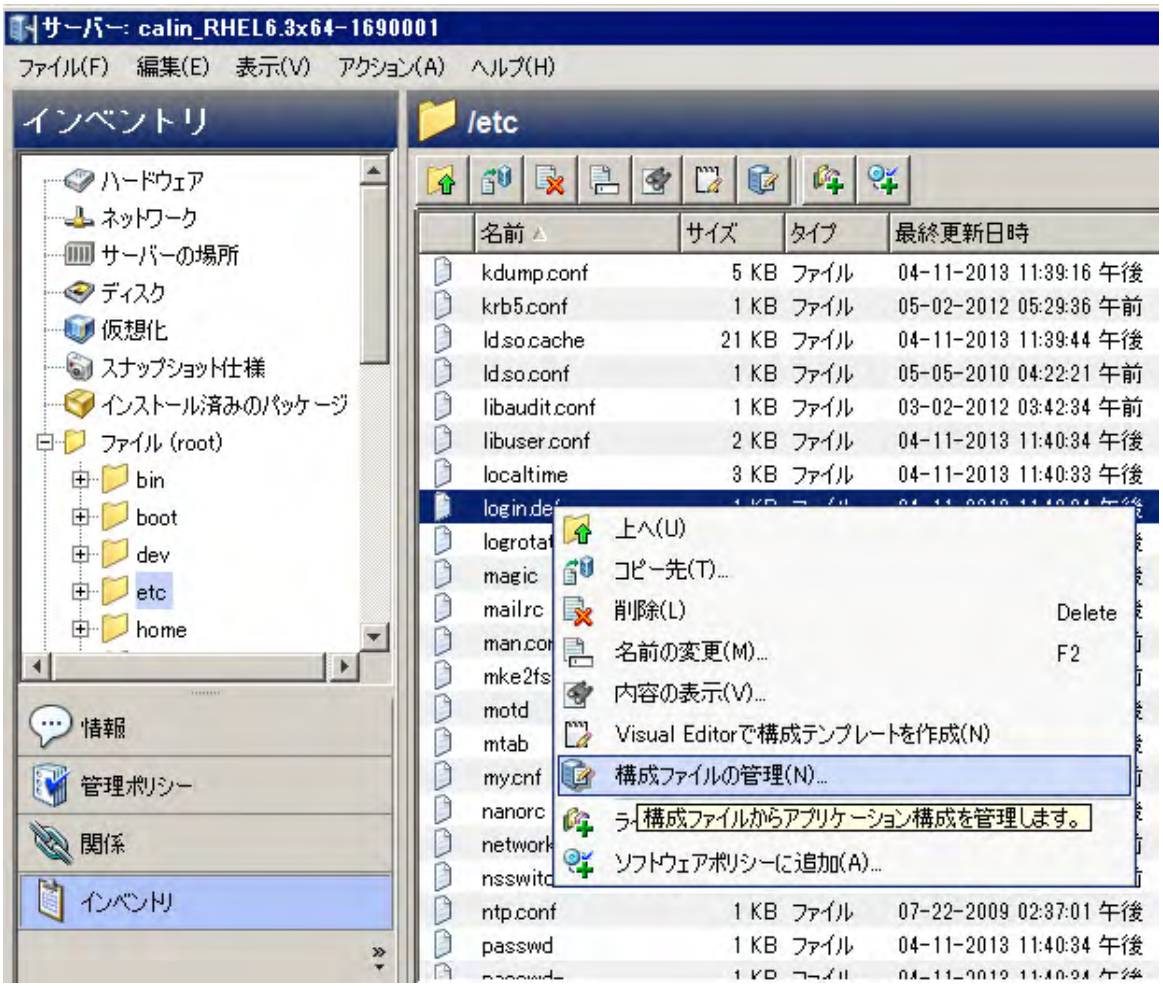
b 内容ペインで、開くサーバーまたはデバイスグループを選択します。

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)。

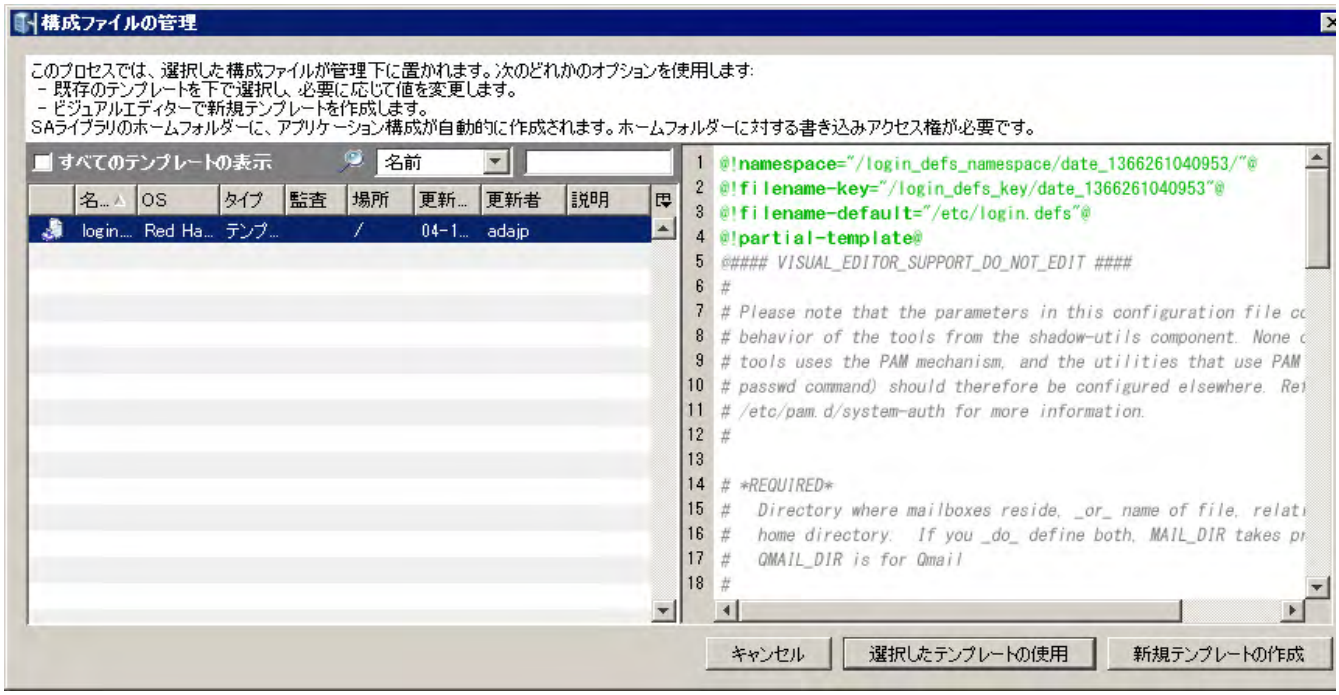
指示に従って、サーバーファイルシステムのルートパスを選択します (通常、Unixではroot、WindowsではAdministrator)。

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。



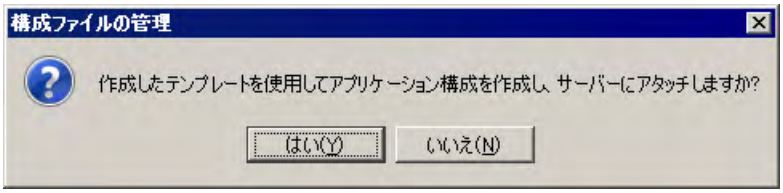
構成ファイル名とパスに一致するSAの既存のテンプレートが候補としてリストされます。サーバープラットフォームに一致するテンプレートだけが候補になります。

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:

- [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。
- [**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。
- [**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 (19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 (24ページ) を参照してください)。
  - [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。

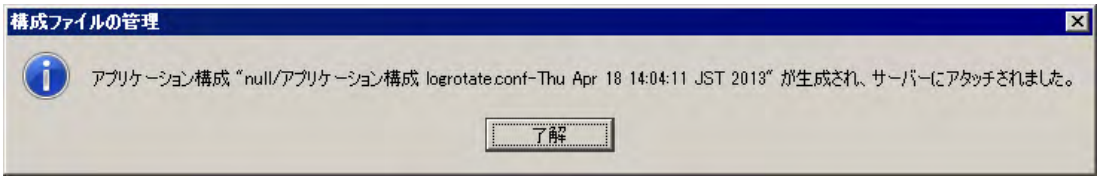


  - オプション:単にテンプレートを保存して後でまた開きたい場合は、[**いいえ**]をクリックします。テンプレートは[プロパティ]タブに指定したディレクトリに保存されます。
  - [**はい**]をクリックすると、アプリケーション構成インスタンスがただちに生成され、サーバーにアタッチされます。構成インスタンスは、値セットエディターに表示されます。

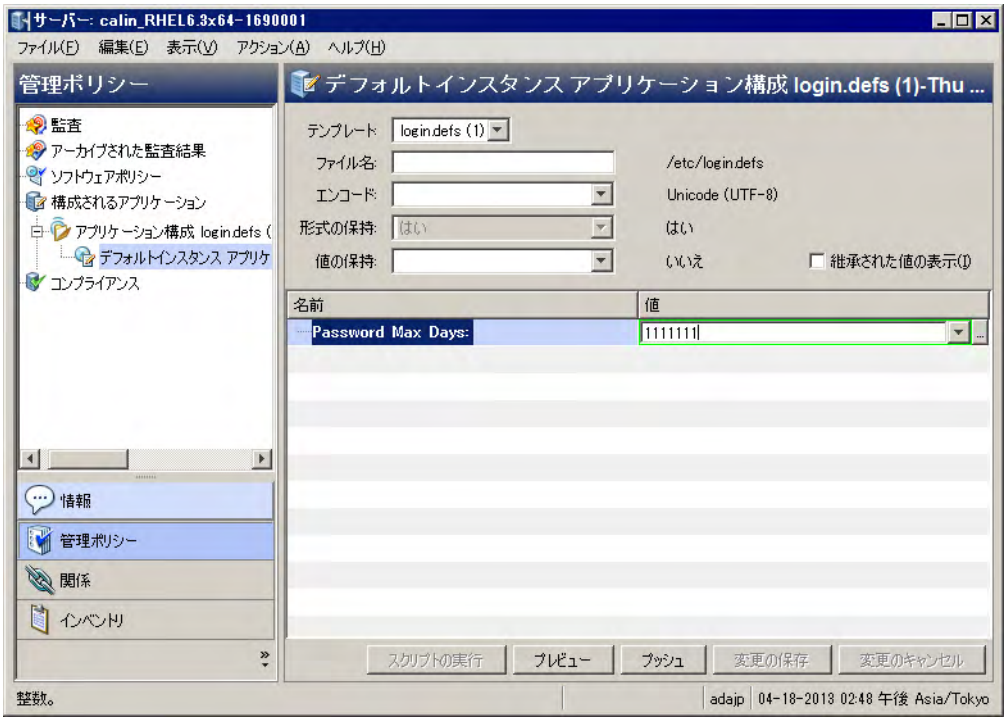
5 [**選択したテンプレートの使用**] をクリックして、選択したテンプレートを使用して新しいアプリケーション構成を作成します(この手順では、自分のホームフォルダーへのアクセス権が必要です)。

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。

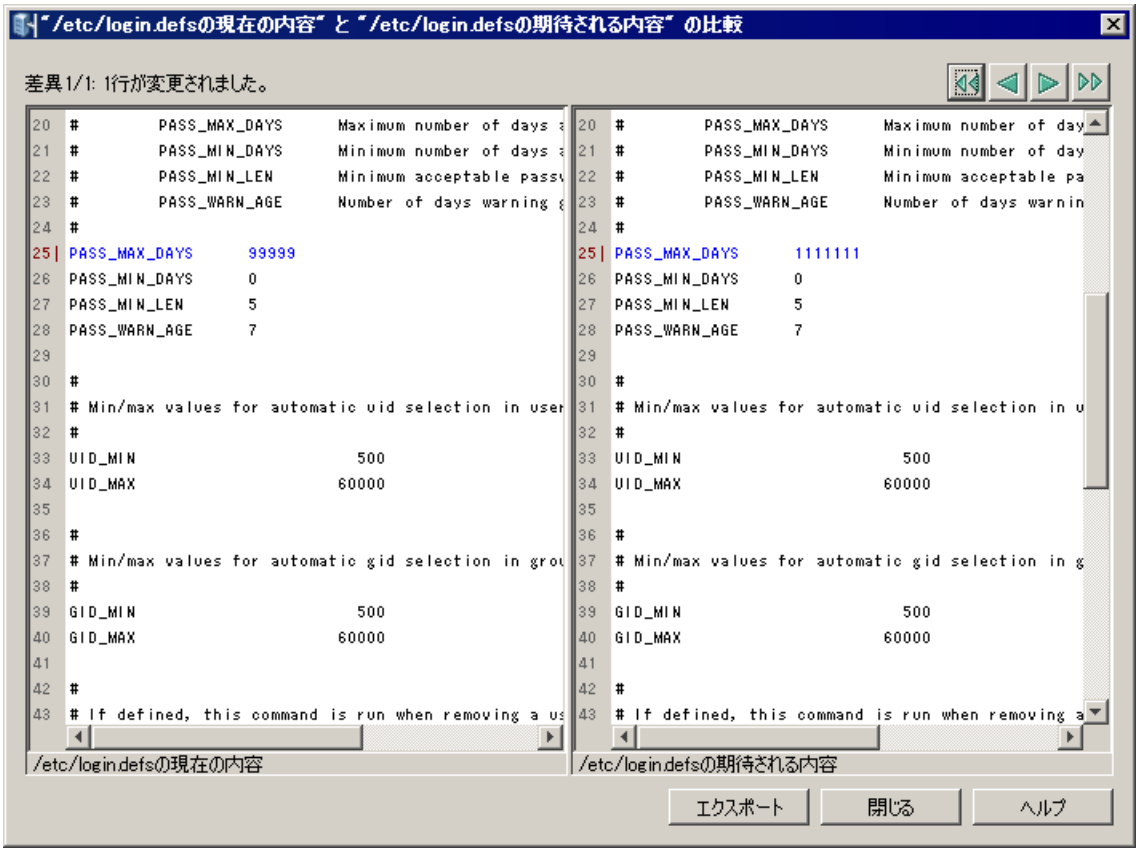


6 [**了解**] をクリックします。構成インスタンスは、値セットエディターに表示されます。



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 (55ページ))。

(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の編集モードに戻ります。



8 [**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 (139ページ) およびXML構成ファイルの管理 (73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について (50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の手順の手順4 (17ページ) 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 (174 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 (64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 (139ページ) を参照してください。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について (63ページ) も参照してください。

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 (17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64ページの表4 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 (17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 (33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。これらのタイプの説明は、64ページの表3 にあります。

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 (64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 (39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、『SA ユーザーガイド: Server Automation』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。

   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、『SAユーザーガイド: Server Automation』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、値セットについて (51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 (61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、値セットについて (51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、サーバーレベルの値セットエディター (61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。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 (36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、手順 5 と手順 6 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 (60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について (65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**]  をクリックして、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します。
- [**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 (40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] 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 (139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 (39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。

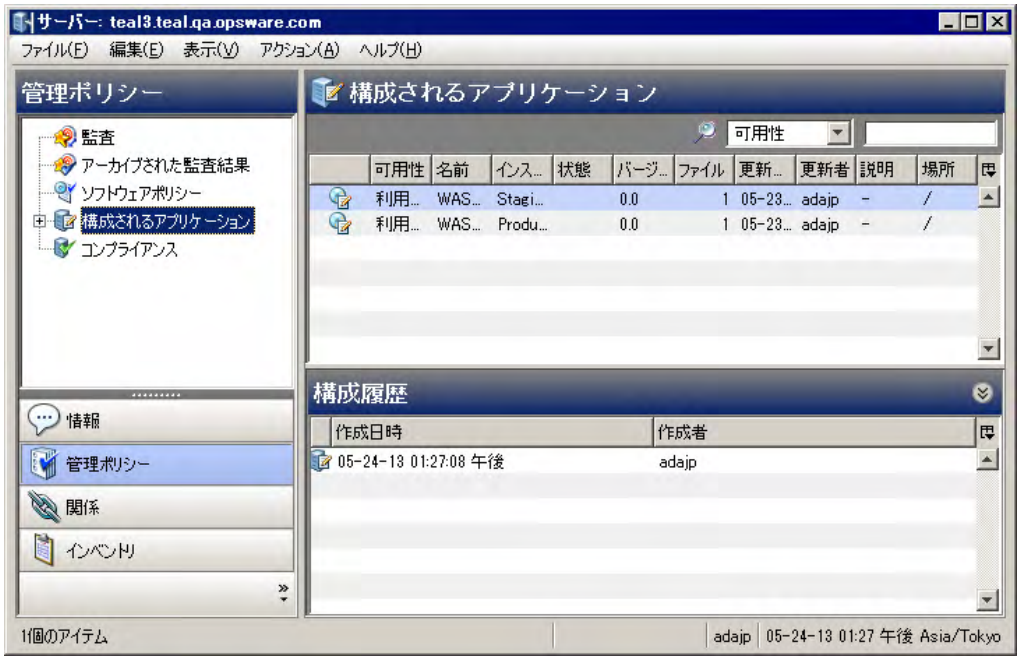
2 [すべての管理対象サーバー] を選択し、目的のサーバーを見つけます。

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。

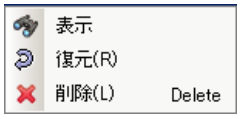
4 [管理ポリシー] タブを選択します。

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。

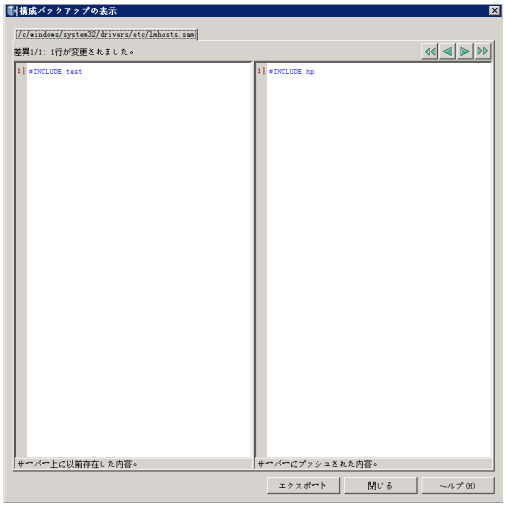


7 履歴のインスタンスを右クリックして [**表示**] を選択すると、スナップショットが表示されます。



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。

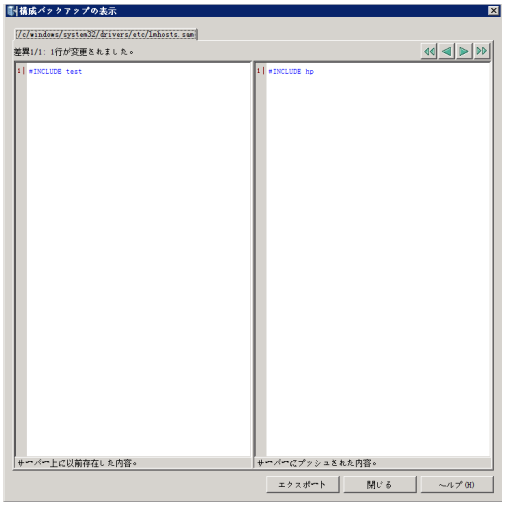
— [選択したプッシュより後(やり直し)]を選択すると、選択した履歴インスタンスの値に構成ファイルが復元されます。つまり、選択したインスタンスが再プッシュされます。

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**] > [Public] 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。

2つの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 (31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 下のローカリゼーションテンプレートの適用の手順に従って、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、[+] ボタンを選択するか、[**アクション**] > [**追加**] 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。

第3章 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について (63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の使用 (71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 (71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 (66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 (41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について

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。

下の図8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。

図8　　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について

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 (139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 (73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、値セットについて (51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 (32ページ)を参照してください。

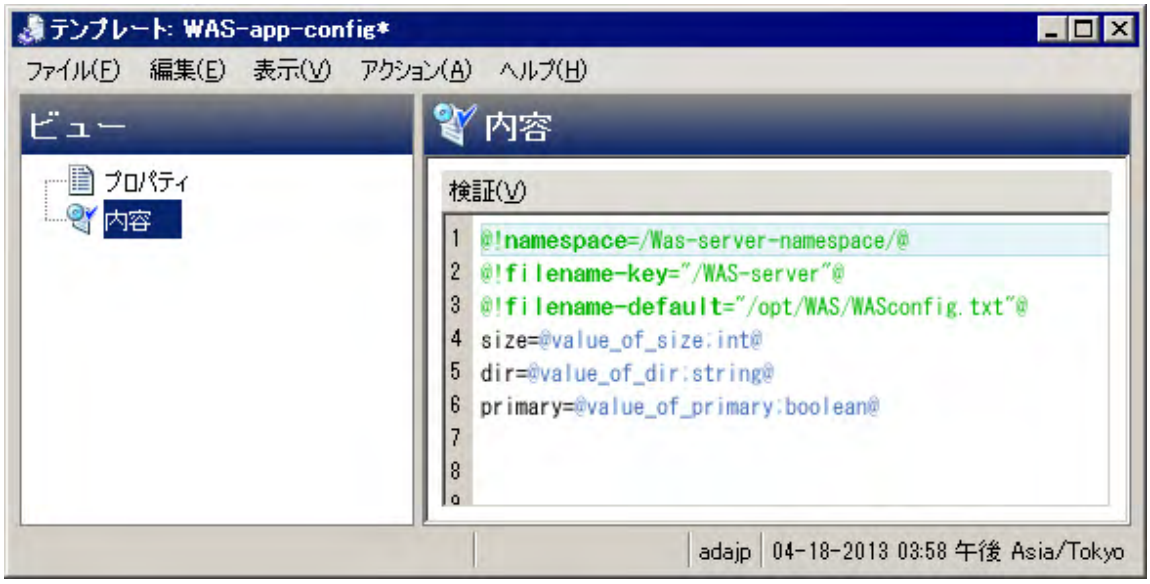
構成ファイルを管理するには、管理対象の各ファイルに対して構成テンプレートを作成し、アプリケーション構成に追加します。詳細については、アプリケーション構成の作成 (16ページ)を参照してください。

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。下の図9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 (97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 (107ページ) 、CMLリファレンス (139ページ) を参照してください。

図9　　CMLで作成された構成テンプレートの例



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 (73ページ) を参照してください。

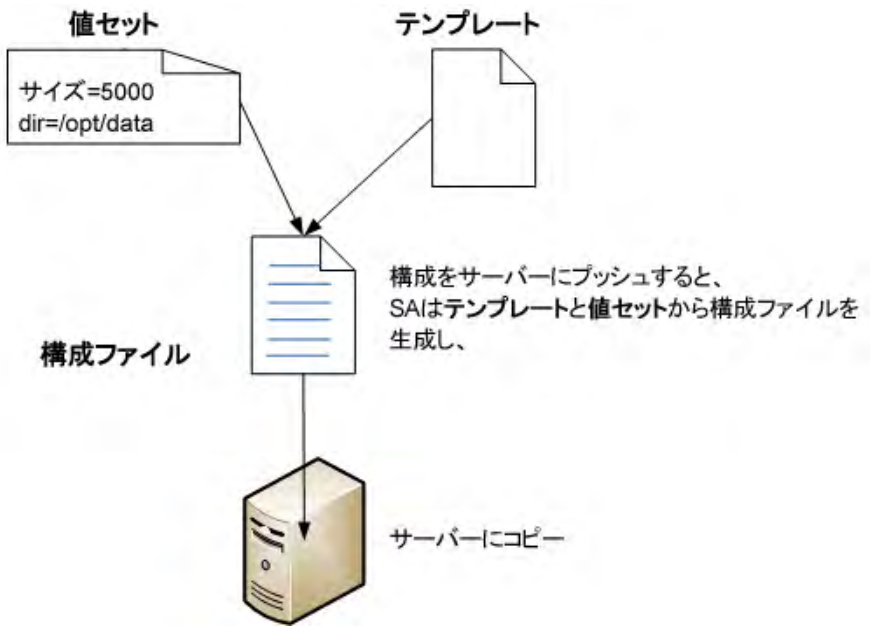
スクリプトテンプレート

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について (63ページ) 、スクリプトからのテンプレートの作成 (34ページ) を参照してください。

値セットについて

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。

下の表 1 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。

表1　　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。詳細については、アプリケーションレベルでの値の設定 (58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。詳細については、ファシリティレベルでの値の設定 (59 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。詳細については、カスタマーレベルでの値の設定 (59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。詳細については、グループレベルでの値の設定 (60 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。	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。テンプレートを選択します。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 (61 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。詳細については、サーバーレベルでの値の設定 (62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。詳細については、サーバーインスタンスレベルでの値の設定 (63 ページ) を参照してください。

下の表 2 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。いちばん下の行は、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。
6. 詳細については、値セットエディターでの値の設定 (55ページ) を参照してください。

値セットエディターについて

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。

各継承レベルで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 (57ページ)、グループレベルの値セットエディター (59ページ)、サーバーレベルの値セットエディター (61ページ) を参照してください。

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。カスタム属性の詳細については、『SA User Guide: Application Automation』を参照してください。
  - 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。詳細については、『SA User Guide: Application Deployment Managert』を参照してください。
- 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。

図 10 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。

  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の意味

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。

図10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。

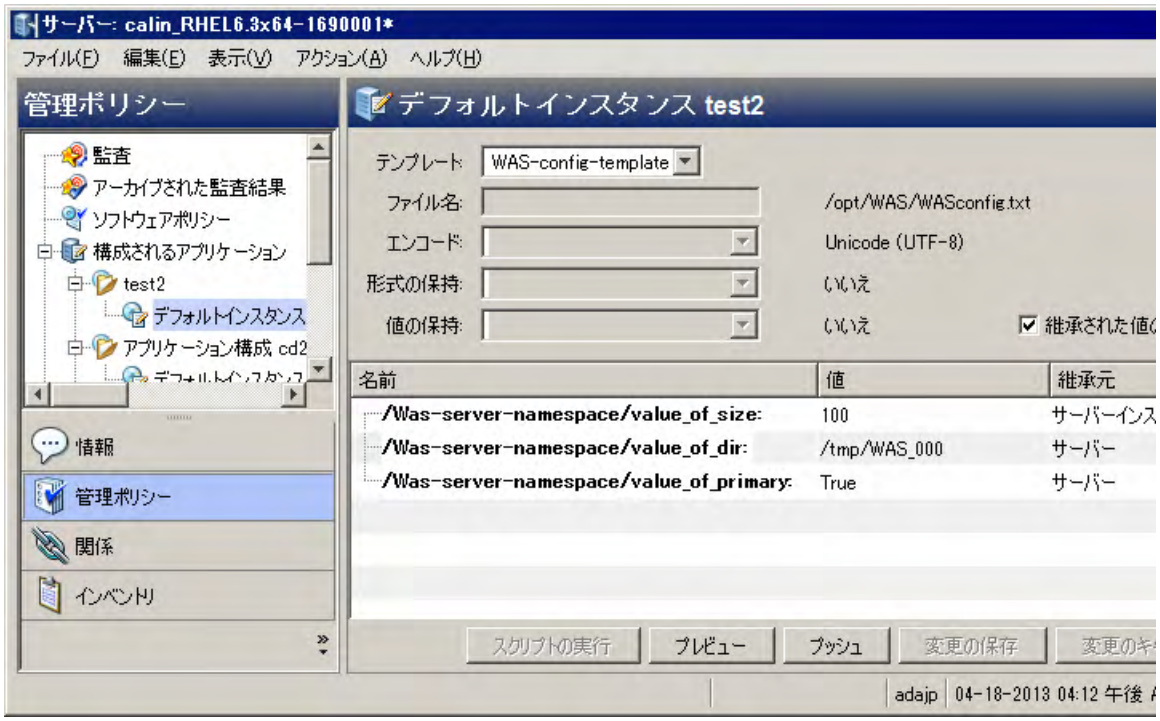
- **名前**: テンプレートファイル内の変数の名前を示します。名前は、シンプルタイプ、シンプルタイプのリスト、または多次元リストです。多次元リストは、その親の下に表示されます。必須の要素は太字で表示されます。多次元リストをダブルクリックすると、表示と非表示を切り替えることができます。リストタイプの値にエントリを追加するには、親を右クリックして、アイテムの **[追加]** を選択します。必須フィールドは、構成テンプレートで設定されます。必須フィールドは空にすることができません。空である場合、アプリケーション構成のプレビューやプッシュは実行できません。

- **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。

図10　　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] ボタンを選択します。

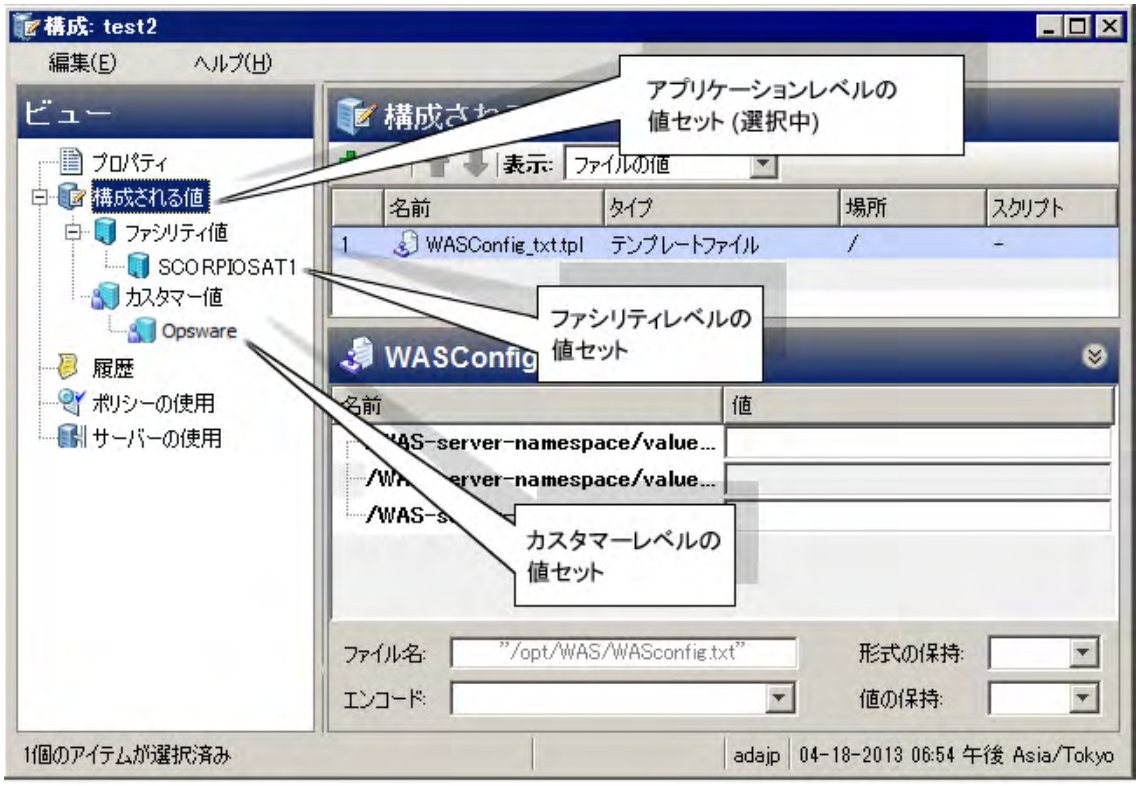
アプリケーション、ファシリティ、カスタマーレベルの値セットエディター

下の図11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。詳細については、アプリケーションレベルでの値の設定 (58ページ) を参照してください。

- [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。詳細については、ファシリティレベルでの値の設定 (59ページ) を参照してください。
- [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。詳細については、カスタマーレベルでの値の設定 (59ページ) を参照してください。

図11　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、図11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。詳細については、値セットエディターでの値の設定 (55ページ) も参照してください。
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. [ファイル]>[保存]を選択して変更内容を保存します。

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、図11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。詳細については、値セットエディターでの値の設定 (55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. [ファイル] > [保存] を選択して変更内容を保存します。

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、図11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 (55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. [ファイル] > [保存] を選択して変更内容を保存します。

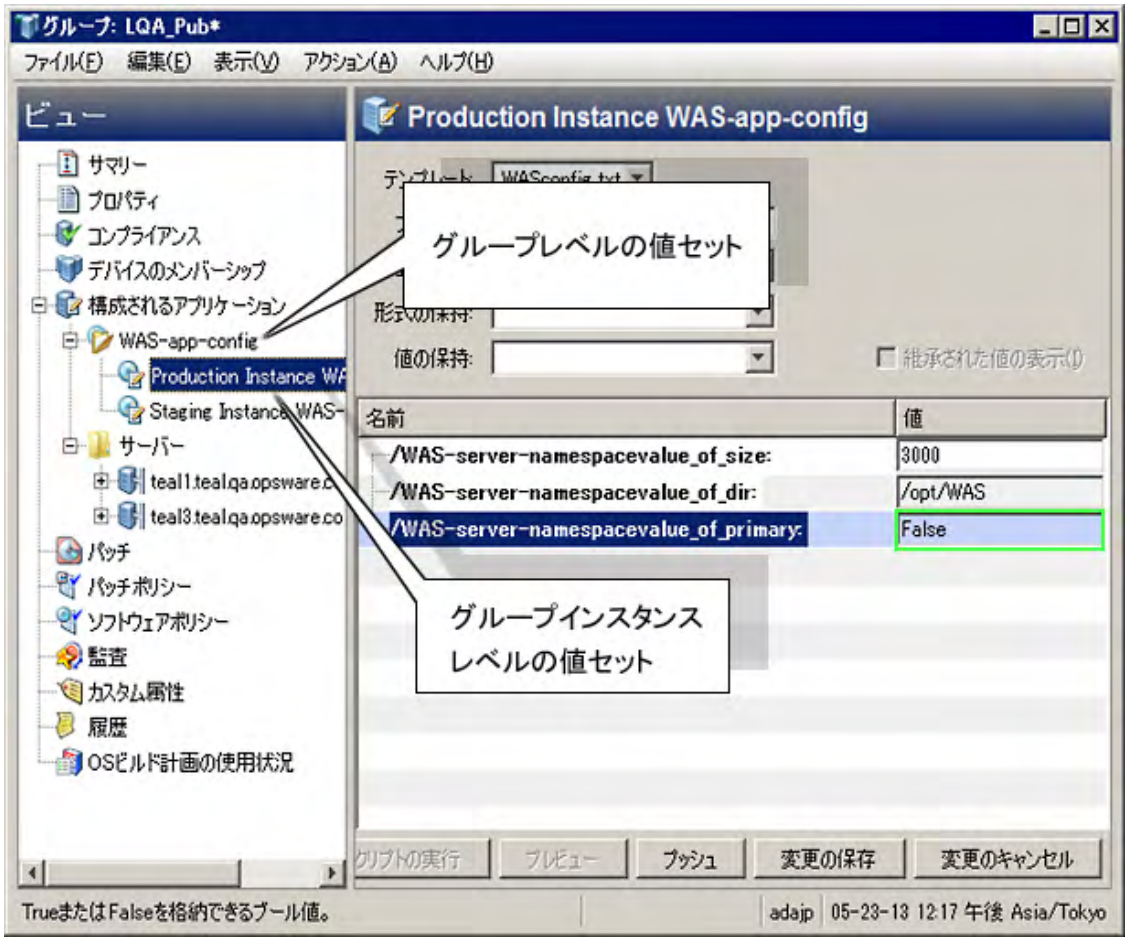
グループレベルの値セットエディター

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)。

下の図12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 (36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。詳細については、グループレベルでの値の設定 (60ページ) を参照してください。
- 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。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 (61ページ) を参照してください。

図12　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、“Production Instance WAS-appconfig”インスタンスの値を表示



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、上の図12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[ビュー]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。
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。詳細については、値セットエディターでの値の設定 (55ページ)も参照してください。
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、上の図12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。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 (36ページ)を参照してください。
2. 左側の[ビュー]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。
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、上の図12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 (55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

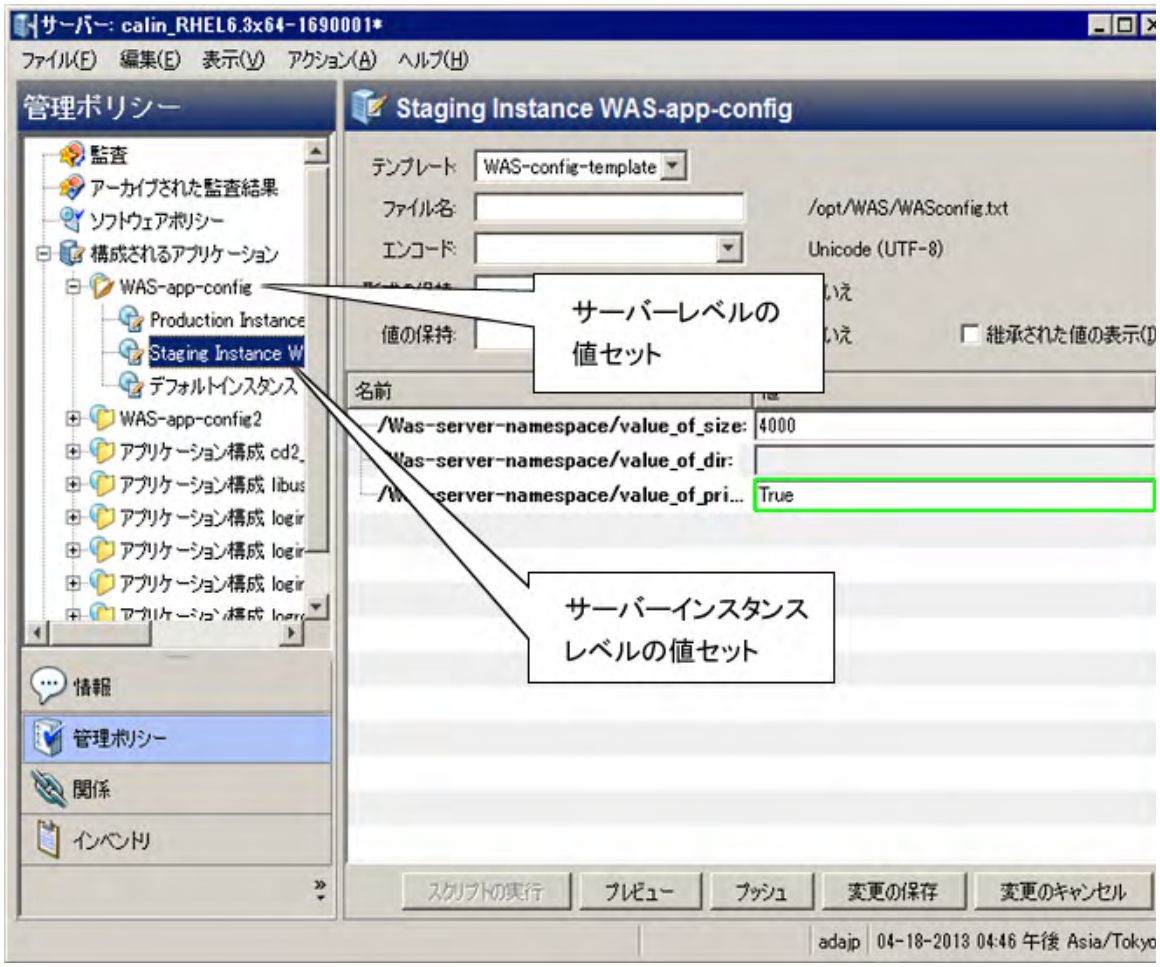
サーバーレベルは、サーバーの値を定義します。これは、下のレベルによってオーバーライドされない限り、指定されたサーバーのアプリケーション構成のすべてのインスタンスに適用されます。

下の図13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。詳細については、サーバーレベルの値セットエディター (61ページ)を参照してください。

- “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。詳細については、サーバーインスタンスレベルでの値の設定 (63ページ) を参照してください。

図13 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、“Staging Instance WAS-appconfig”インスタンスの値を表示



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、上の図13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 (36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。詳細については、値セットエディターでの値の設定 (55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、上の図13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 (36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。詳細については、値セットエディターでの値の設定 (55ページ) も参照してください。

8 [変更の保存] ボタンを選択します。

アプリケーション構成でのスクリプトの実行について

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。

⚠ 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

下の表 3 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。

表3　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. 表3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 (34ページ) も参照してください。

下の表 4 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。

表4　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。
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。

表4　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 .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、表4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 (34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について

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の使用 (71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 (34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 (35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 (39ページ)を参照してください。

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 (174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、『SA User Guide: Application Automation』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、『SA User Guide: Application Automation』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、[監査可能]プロパティがチェックされていないテンプレートに対して表示されます。詳細については、構成テンプレートの作成(17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
- 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

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。

図14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。

図14　　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

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 (70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 (68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 (68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 (69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

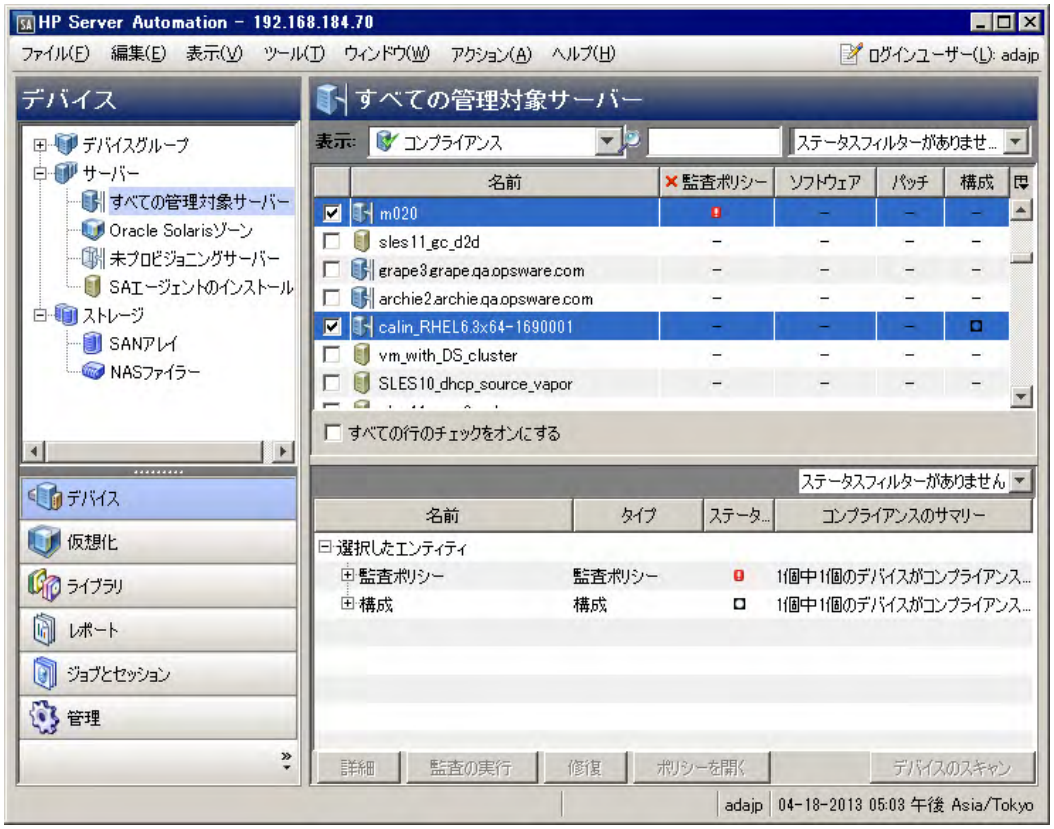
複数のサーバーのアプリケーション構成コンプライアンスを表示するには、次の手順を実行します。

1 SAクライアントのナビゲーションペインから、[**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、図15のように下の詳細ペインに表示されます。

図15　　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、図 16 のように下の詳細ペインに表示されます。

図16 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

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。

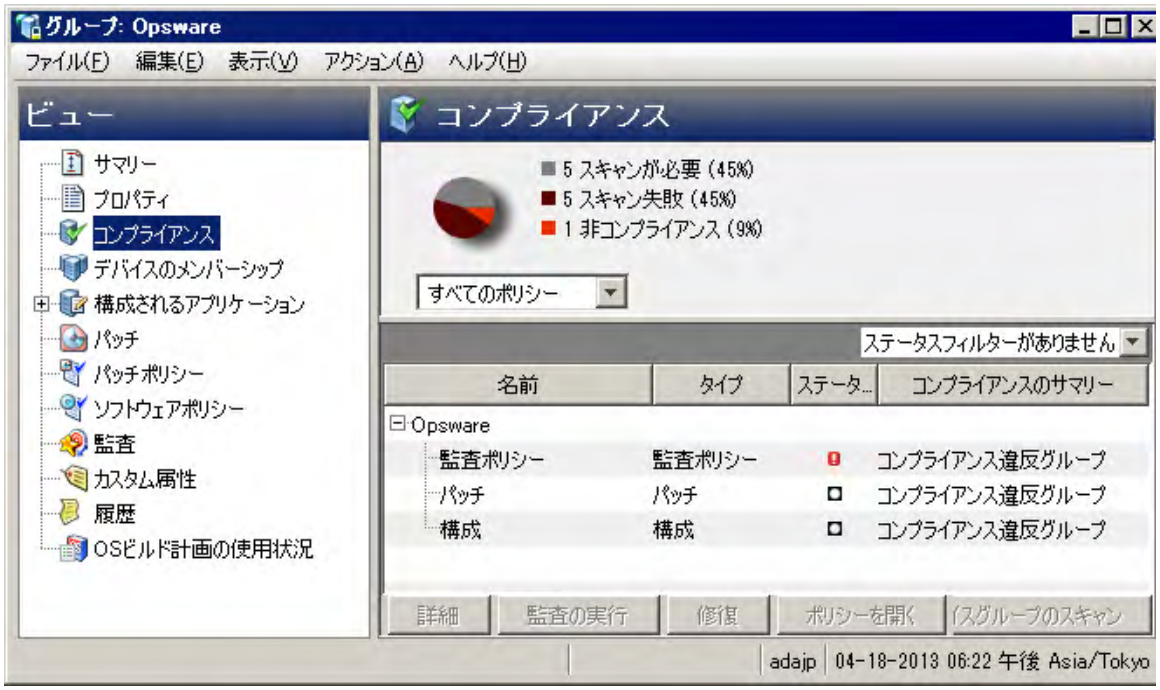
1 SAクライアントのナビゲーションペインで、**[デバイス]** > **[デバイスグループ]** を選択します。

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。図17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。

図17　　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ユーザーガイド: 監査とコンプライアンス』の「監査と修復」を参照してください。特に、「アプリケーション構成ルールの構成」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の使用

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ユーザーガイド: ソフトウェア管理』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ユーザーガイド: ソフトウェア管理』を参照してください。

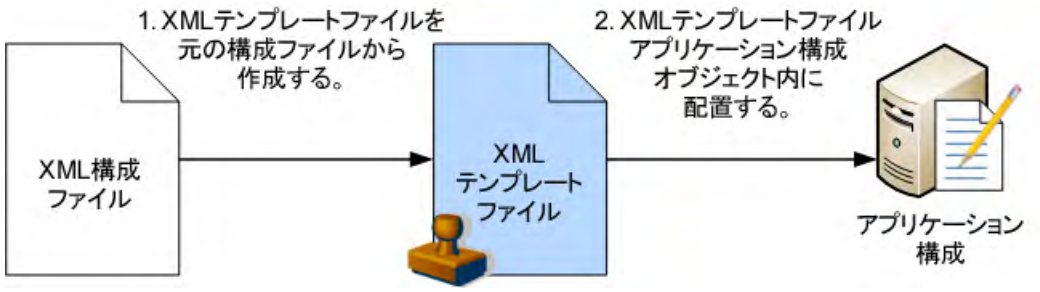
第4章 XML構成ファイルの管理

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。

この章では、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。

図18 XML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 (83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 (89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 (80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 (17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 (80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。

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。

```
<!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.
-->
```

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。

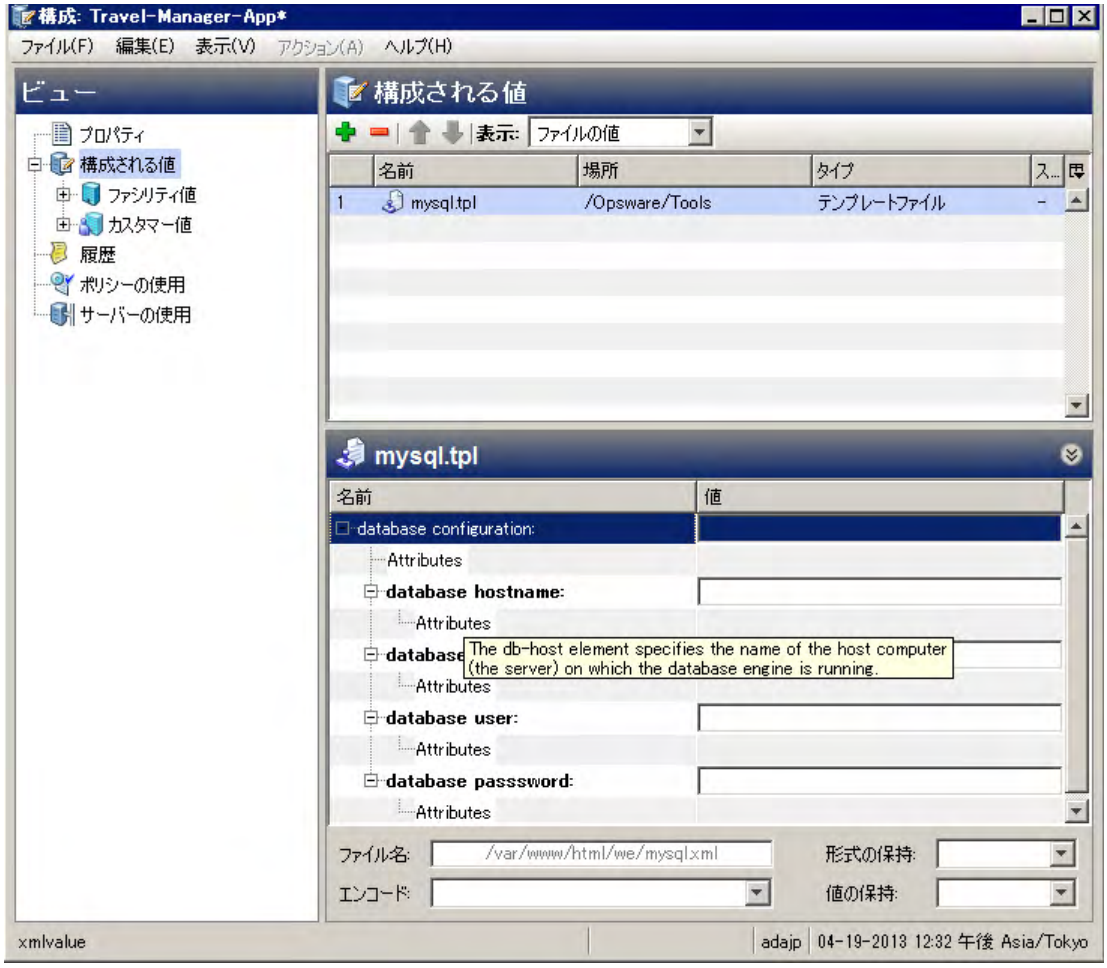
```
<!--
ACM-TIMEOUT = 1
ACM-FILENAME-KEY = /files/TravelManager
ACM-FILENAME-DEFAULT = /var/www/html/we/mysql2.xml
ACM-NAMESPACE = /TravelManager/
ACM-DOCTYPE = db-config
ACM-DOCTYPE-SYSTEM-ID = mysql.dtd
ACM-DOCTYPE-PUBLIC-ID = -//Williams Events//Travel Manager//EN
-->

<!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、図19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。

図19　　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

表5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。

表5　　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> 必須。デフォルト値なし。	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。
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。

表5　XMLおよびXML-DTDテンプレート設定（続き）

属性	説明
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> オプション XML-DTDテンプレートのみ。	printableは、SAクライアントでの要素または属性のラベルの値を設定します。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。

第5章 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 (73 ページ)で説明したTravel Managerの例に基づいています。

非DTD XMLファイルmysql.xml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。
- **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。

6 [**ファイル**] > [**保存**] を選択します。

7 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。

2 次のXMLをコピーして内容ペインに貼り付けます。

```
<!--
ACM-NAMESPACE = /TravelManager
ACM-FILENAME-KEY = /files/TravelManager
ACM-FILENAME-DEFAULT = /var/www/html/we/mysql.xml
-->
```

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 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。
- **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 (61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。

下の図20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。

図20 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)

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、[**構成の複製**] 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 (52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、上の非 DTD XML ファイル mysql.xml の例 (83ページ) 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 (55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 (55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の意味 (56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。

第6章 XMLチュートリアル2 - XML-DTD構成テンプレートの作成

この章では、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 (73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、下の図21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 (77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から、[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 (89ページ) 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 (89ページ) 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。

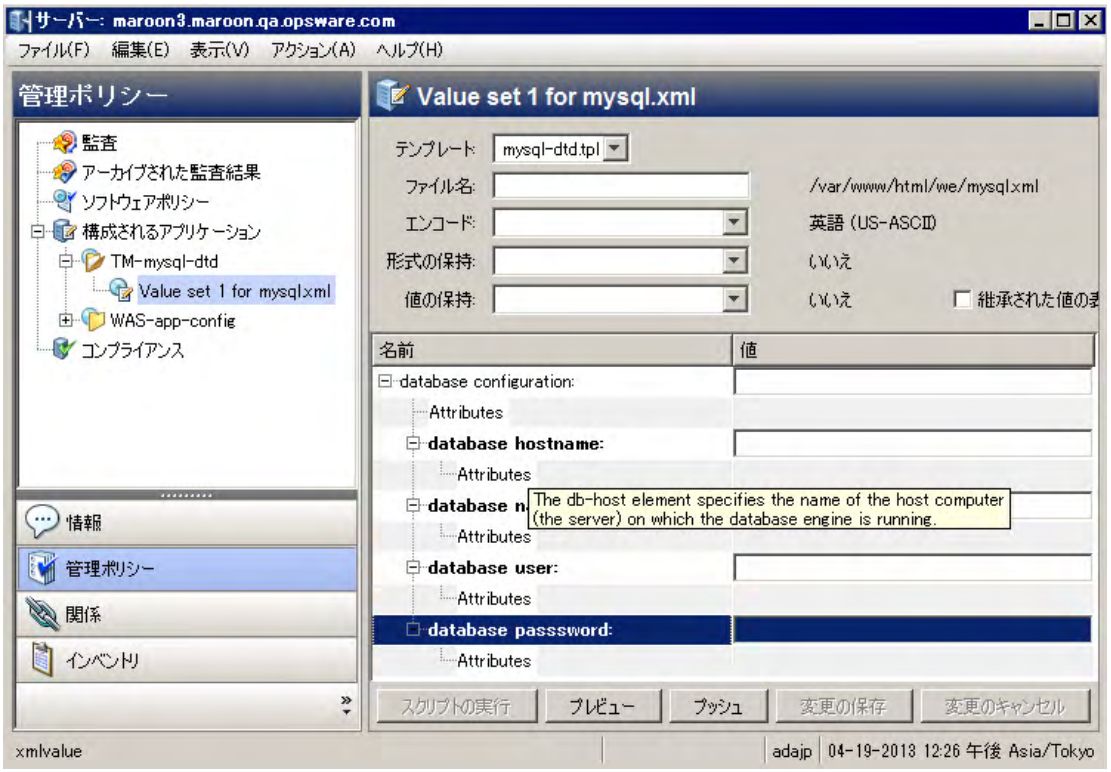
4 [値] 列の下の任意の値を右クリックして、[**値のインポート**] メニューを選択します。

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。

下の図21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。

図21　　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　図21に示すように、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] ボタンを選択します。

4　[プレビュー] ボタンを選択して、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] ボタンを選択します。[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] ボタンを選択します。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。

第7章 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

この章では、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 (32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。

表6 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します。

アプリケーションレベルのデフォルト値の設定

アプリケーションレベルのデフォルト値を設定するには、次の手順を実行します。

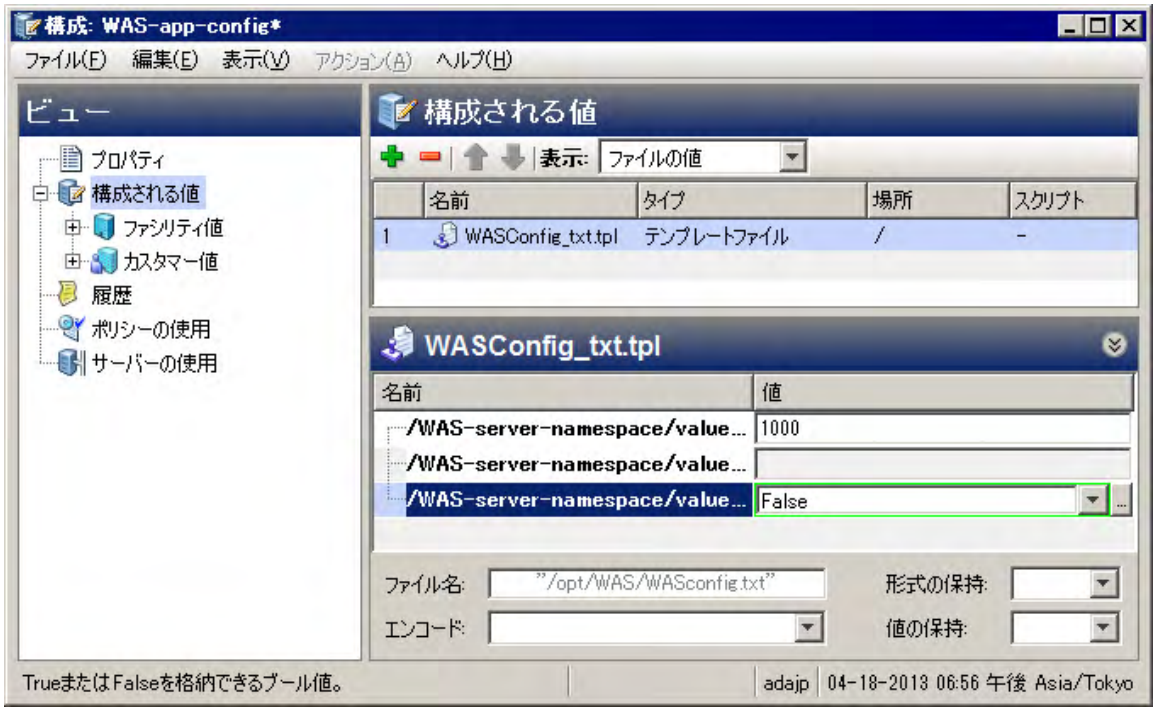
1 上で作成したアプリケーション構成オブジェクトを開きます。

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。

プライマリサーバー RHEL001に対して**サーバーレベルのデフォルト値を設定**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。

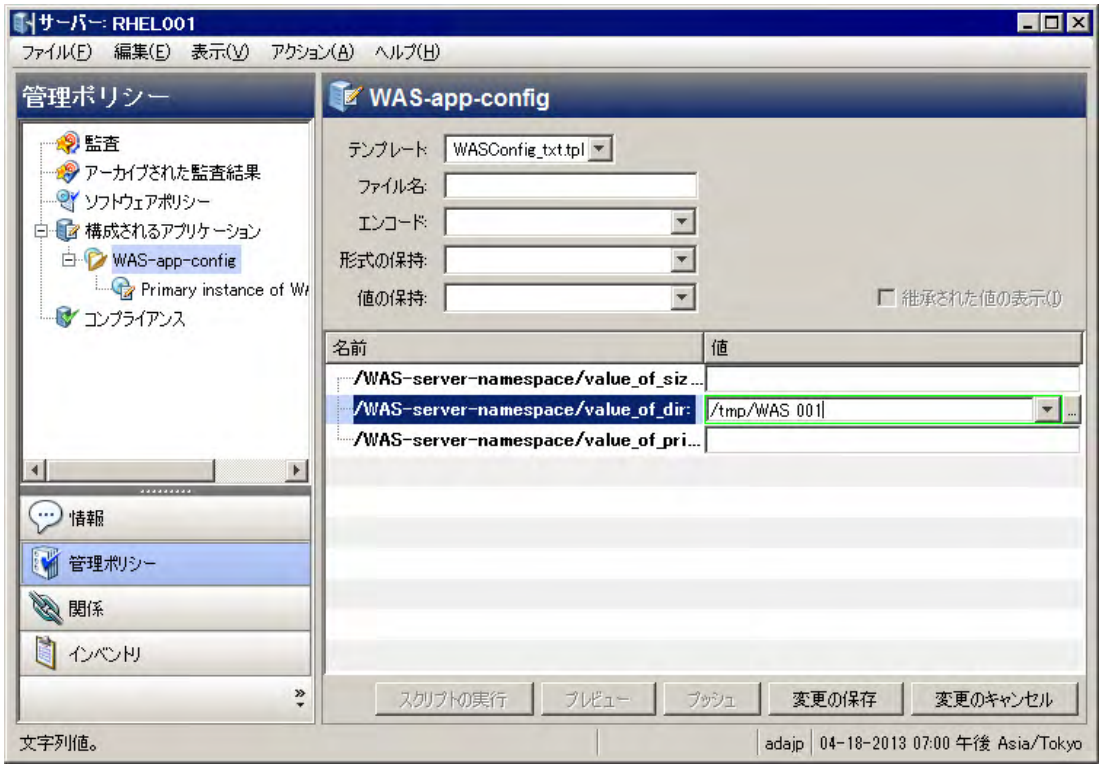
2 RHEL001サーバーを選択し、[アクション] > [開く] を選択します。

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。

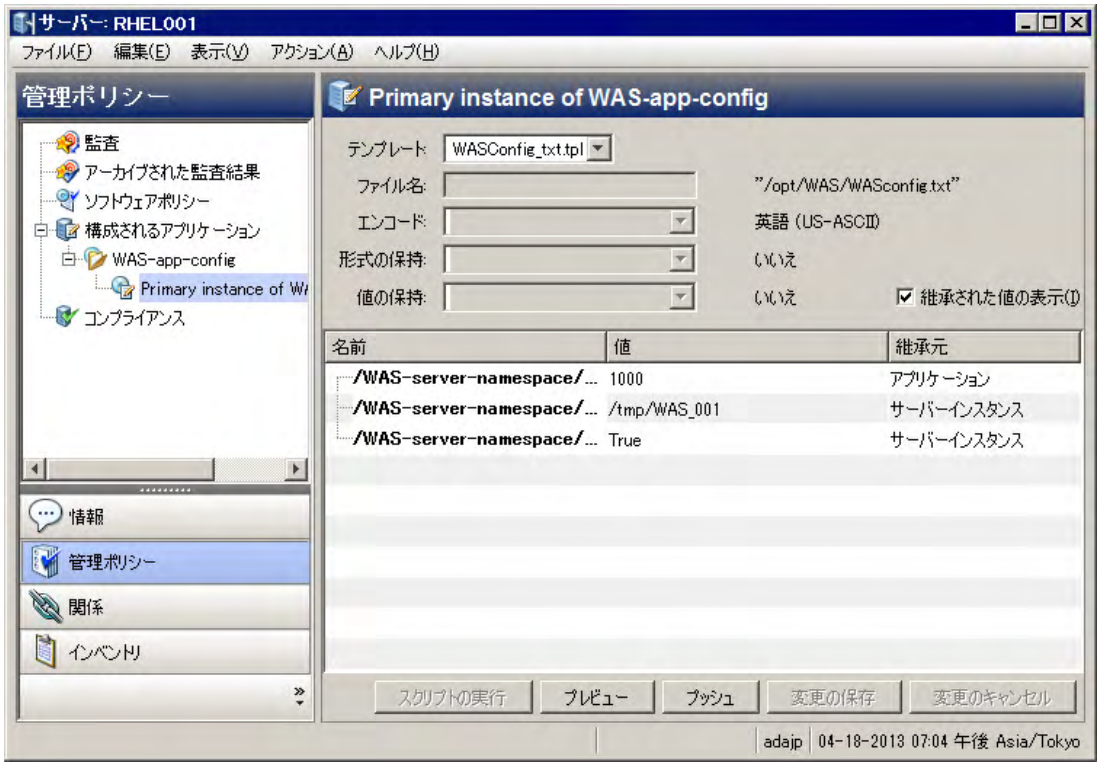
10 [継承された値の表示] を選択して、アプリケーションレベルのデフォルトとサーバーレベルのデフォルトから継承される値を表示します。“value_of_size” と “value_of_primary” はアプリケーションレベルから、“value_of_dir”はサーバーレベルから継承されます。

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。

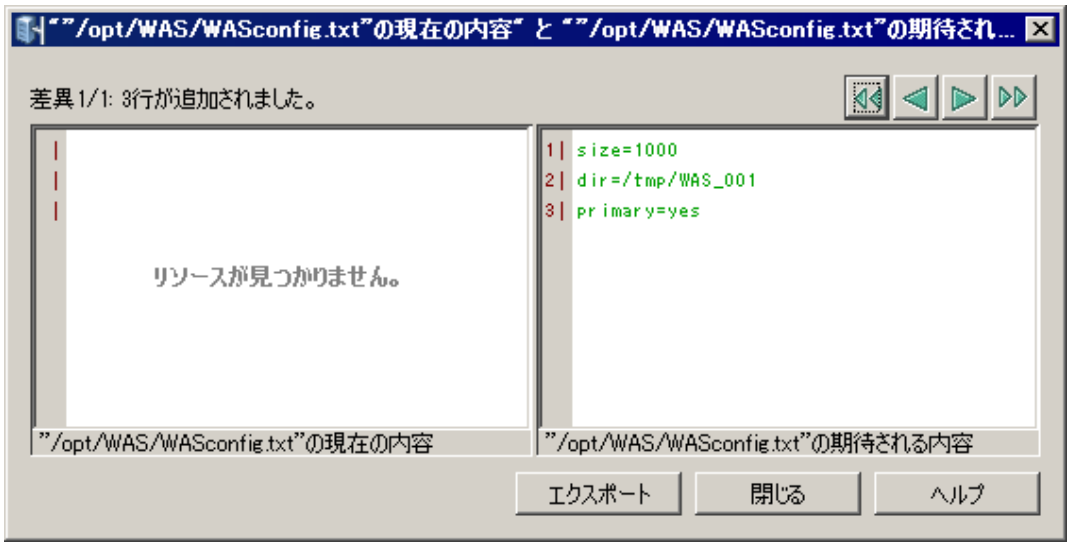
セカンダリサーバー RHEL008に対してサーバーレベルのデフォルト値を設定するには、次の手順を実行します。

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] 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“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。

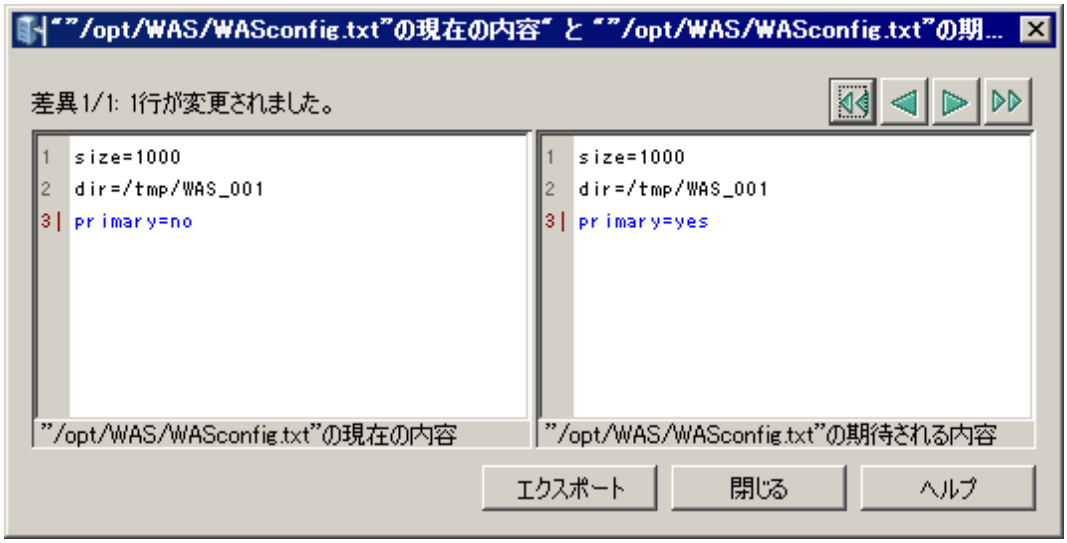
10 [継承された値の表示]を選択して、アプリケーションレベルのデフォルトとサーバーレベルのデフォルトから継承される値を表示します。“value_of_size” と “value_of_primary” はアプリケーションレベルから、“value_of_dir”はサーバーレベルから継承されます。

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

構成変更をサーバー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 (107ページ) を参照してください。

第8章 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 (121ページ) 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 (125ページ) 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 (121ページ) 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

表8にセットアップ命令の説明を示します。

表8　　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。 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。

表9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。

表9　　[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。
	`[`
	角かっこ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。
	`[`
	角かっこ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。

3 次に、構成ファイルの [Options] セクションに現れることができ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。表10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。

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
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 = @reject_response_url;string;r"(HTTP_URLSCAN_STATUS_HEADER)|(HTTP_URLSCAN_ORIGINAL_VERB)|(HTTP_URLSCAN_ORIGINAL_URL)";optional@	構文: 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。

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
`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とがあります。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@
```

表11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。

表11　[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。
	`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@
.@.@
```

表12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。

表12　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。
	`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=":"@
@.@:
```

表13に、これら2つのタグの構文を示します。

表13　[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@
@.@
```

表14に、これらのタグの構文を示します。

表14 [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。
	`;`
	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]@
```

表15に、これらのタグの構文を示します。

表15 [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。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 (32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。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 (33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。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 (36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。詳細については、アプリケーション構成のプッシュ (39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 (97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
.jpg
.jpeg
.gif
```

```
[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]@
```

第9章 CML入門

この章では、**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(139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(97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(107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 (137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。詳細については、namespaceタグ (128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。詳細については、filename-keyタグ (129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。詳細については、filename-defaultタグ (129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 (139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 (160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。

説明

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。

説明

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。

説明

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 (152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。

表16　　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 (131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 (131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 (132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フィールドの設定 (55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 (161ページ) も参照してください。

第10章 CMLリファレンス

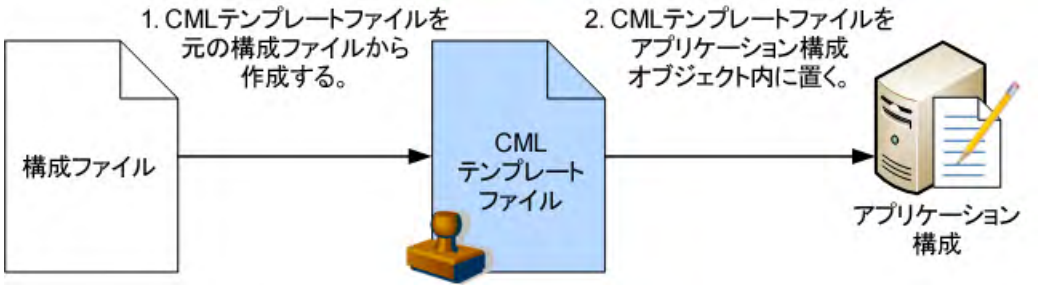
構成モデリング言語 (CML) は、**構成ファイルのテンプレート**を作成して、SAから管理できるようにするためのものです。CMLテンプレートは、構成ファイルの形式をモデル化するために作成する独立したファイルであり、構成ファイル内の可変の値をサーバーごとに異なる値に設定できます。

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について (63ページ) を参照してください。

図22 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 (73ページ) を参照してください。

構成テンプレートについて

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、値セットについて (51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}@
```

CMLタグには、空白と‘@’のいずれも指定できません。

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 (142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、下の@!namespace CMLタグによる名前空間の定義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 (141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 (141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@##
- 置換タグ: @

- 命令タグ: @!
- ブロック (またはグループ) タグ: @[@...@]@
- ループタグ: @*
- ループターゲットタグ: @.
- 条件タグ: @?
- DTDタグ: @~

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 (152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 (158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 (160ページ) とCMLの通常オプション属性 (162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 (160ページ) とCMLの通常オプション属性 (162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 (160ページ) とCMLの通常オプション属性 (162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。

```
@[{レベル}][ [;{オプション}[;{オプション}]...]@{CMLタグのセット} @[{レベル}]]@
```

ブロック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。

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 (160ページ) とCMLの通常オプション属性 (162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。詳細については、ループターゲットタグ: @. (149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 (152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 (158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 (160ページ) とCMLの通常オプション属性 (162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: @* (148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 (173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 (77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 (152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 (158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 (160ページ) とCMLの通常オプション属性 (162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

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

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

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

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。

詳細については、プッシュタイムアウト値の変更 (41ページ) も参照してください。

@!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。
- 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で設定できるので、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することが重要です。

ACMには、継承スコープにある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: 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: 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、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: 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”

シーケンスのマージ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。このようなマージは、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、構成テンプレートでプライマリキーオプションを使用する方法があります。

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、構成テンプレートでCML primary-keyオプションを使用する方法があります。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
```

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マージ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マージし、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

表17に、何種類かのCMLタグを対象としてCMLの文法を示します。

表17　　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 "@"
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 "@"

表17　CMLの文法（続き）

CMLタグ/要素	説明
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"
タイプ要素	"int" / "string" / "ip" / "port" / "file" / 等...
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 ">"

表17 CMLの文法（続き）

CMLタグ/要素	説明
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