Sun Desktop Manager 1.0 開発者ガイド

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 819–6097–10
2006 年、1 月

原典: Sun Desktop Manager 1.0 Developer Guide

Part No: 819-2728

Revision A

060406@14774

目次

はじめに

『Sun Desktop Manager 1.0 開発者ガイド』では、開発者が Sun™ Desktop Manager 1.0 Beta で動作するアプリケーションを作成できるようになるためのガイドラインについて説明します。デフォルトでは Desktop Manager で認識されないソフトウェアアプリケーションの構成を一元管理する方法について、必要な知識を提供します。

このマニュアルを読むと、新しい構成設定の保存場所と表示方法に関する情報を含んだ「テンプレート」というファイルを作成して配備できるようになります。また、設計に関する推奨事項、高度なテンプレートの作成、リファレンスなど、必要なテンプレートの構築に役立つ情報を提供しています。

対象読者

『Sun Desktop Manager 1.0 開発者ガイド』は、Sun Desktop Manager を拡張して追加のアプリケーションや設定を一元的に構成できるようにしたい開発者や先進サイトの管理者を対象にしています。

このマニュアルを読む前に

ユーザーが少なくとも『Sun Desktop Manager 1.0 管理ガイド』の「概要とアーキテクチャー」の章を読んでおり、Desktop Manager を管理し、使用した経験がいくらかあることを推奨します。XML の知識があると役立ちますが、必須ではありません。

マニュアルの構成

第 1 章では、Sun Desktop Manager 1.0 の概要を述べます。

第 2 章では、テンプレートの基本とその作成方法を説明します。

第 3 章では、複雑なテンプレートの作成方法とその使い方について説明します。

第 4 章では、推奨する設計についてガイドラインを提供します。

第 5 章では、構成の概念について説明します。

付録 A では、構成パスのマッピングについて役立つ情報を提供します。

付録 B では、テンプレートの要素と属性のリファレンスを提供します。

付録 C には、テンプレートの DTD が含まれています。

関連情報

Sun の次のマニュアルには、このマニュアルに関連する追加情報が含まれています。

- 『Sun Desktop Manager 1.0 管理ガイド』
- 『Sun Desktop Manager 1.0 インストールガイド』

マニュアル、サポート、およびトレーニング

Sun のサービス	**URL**	内容
マニュアル	http://jp.sun.com/documentation/	PDF 文書および HTML 文書をダウンロードできます。
サポートおよびトレーニング	http://jp.sun.com/supporttraining/	技術サポート、パッチのダウンロード、および Sun のトレーニングコース情報を提供します。

表記上の規則

このマニュアルでは、次のような字体や記号を特別な意味を持つものとして使用します。

表 **P–1** 表記上の規則

字体または記号	意味	例
`AaBbCc123`	コマンド名、ファイル名、ディレクトリ名、画面上のコンピュータ出力、コード例を示します。	`.login` ファイルを編集します。 `ls -a` を使用してすべてのファイルを表示します。 `system%`
`AaBbCc123`	ユーザーが入力する文字を、画面上のコンピュータ出力と区別して示します。	`system%` **`su`** `password:`

表 P–1 表記上の規則　(続き)

字体または記号	意味	例
AaBbCc123	変数を示します。実際に使用する特定の名前または値で置き換えます。	ファイルを削除するには、`rm` *filename* と入力します。
『』	参照する書名を示します。	『コードマネージャ・ユーザーズガイド』を参照してください。
「」	参照する章、節、ボタンやメニュー名、強調する単語を示します。	第 5 章「衝突の回避」を参照してください。 この操作ができるのは、「スーパーユーザー」だけです。
\	枠で囲まれたコード例で、テキストがページ行幅を超える場合に、継続を示します。	`sun% grep '^#define \` `XV_VERSION_STRING'`

コード例は次のように表示されます。

- C シェル

```
machine_name% command y|n [filename]
```

- C シェルのスーパーユーザー

```
machine_name# command y|n [filename]
```

- Bourne シェルおよび Korn シェル

```
$ command y|n [filename]
```

- Bourne シェルおよび Korn シェルのスーパーユーザー

```
# command y|n [filename]
```

[ ] は省略可能な項目を示します。上記の例は、*filename* は省略してもよいことを示しています。

| は区切り文字 (セパレータ) です。この文字で分割されている引数のうち 1 つだけを指定します。

キーボードのキー名は英文で、頭文字を大文字で示します (例: `Shift` キーを押します)。ただし、キーボードによっては `Enter` キーが `Return` キーの動作をします。

ダッシュ (-) は 2 つのキーを同時に押すことを示します。たとえば、`Ctrl-D` は `Control` キーを押したまま `D` キーを押すことを意味します。

◆ ◆ ◆ 第 1 章

Sun™ Desktop Manager 1.0 の概要

この章では、Sun Desktop Manager 1.0 ベータ版の概要を示し、Desktop Manager のテンプレートの作成に必要な概念を紹介します。

新しいアプリケーションを Desktop Manager に導入するには、まずテンプレートを作成する必要があります。また、それらのテンプレートを Desktop Manager に登録する必要があります。「テンプレート」とは、新しい構成設定をどこに保存して、どのように表示するかの情報を格納するファイルです。Desktop Manager はこれらのテンプレートを使用して、構成プロファイルに関して必要な情報をすべて取り込みます。

概要

Desktop Manager は、Sun Desktop Manager の構成を一元管理するために必要な基盤を提供します。現在、Desktop Manager はクライアント側とサーバー側の次のコンポーネントで構成されています。

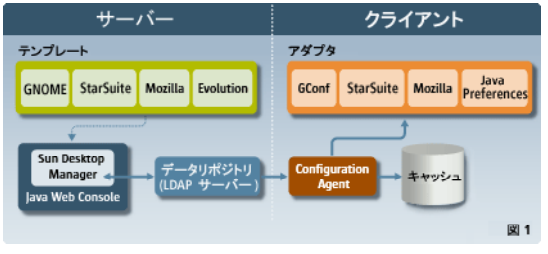


図 1–1 クライアント側とサーバー側のコンポーネント

構成の伝播

ポリシーはすべて、LDAP サーバーのような中央の構成リポジトリに格納されます。「プロファイル」とは、意味的に一貫した構成設定のグループを指します。LDAP サーバーか

らプロファイルデータを取り出したり、データをローカルでキャッシュするのは、各クライアントマシンで実行している Configuration Agent の役目です。Configuration Agent は LDAP サーバーの変更を定期的にチェックし、必要に応じてキャッシュを更新します。さらに、Configuration Agent は関係しているアプリケーションのすべてに通知を送信します。 StarSuite、Mozilla、Evolution、GNOME などのデスクトップアプリケーションは、対応するアダプタを使用してポリシーを読み取ります。これらのアダプタは、キャッシュや Configuration Agent との必要な通信をカプセル化します。

構成の管理

Desktop Manager は、組織、グループ、ユーザーなど、組織のさまざまな階層レベルの構成設定を Web ブラウザを使って表示、定義、実行できる Web ベースの管理ツールです。Desktop Manager は Java Web Console の一部です。Sun Web Console は Web ベースの共通グラフィカルユーザーインタフェース (GUI)、シングルサインオン認証など、Sun の管理ツールのすべてに必要な基盤を提供します。Desktop Manager は、構成リポジトリ内の構成設定の確認、定義、実行と、それらの構成設定を表示するための GUI の提供に テンプレート を使用します。

◆ ◆ ◆ 第 2 章

2 テンプレート

Desktop Manager のテンプレートは、構成リポジトリ内のあらゆる構成設定の場所に関する情報を提供します。また、それらが Desktop Manager の GUI で視覚的に表現される方法についても情報を提供します。テンプレートは文書型定義 (DTD) ファイルに準拠する XML ファイルです。

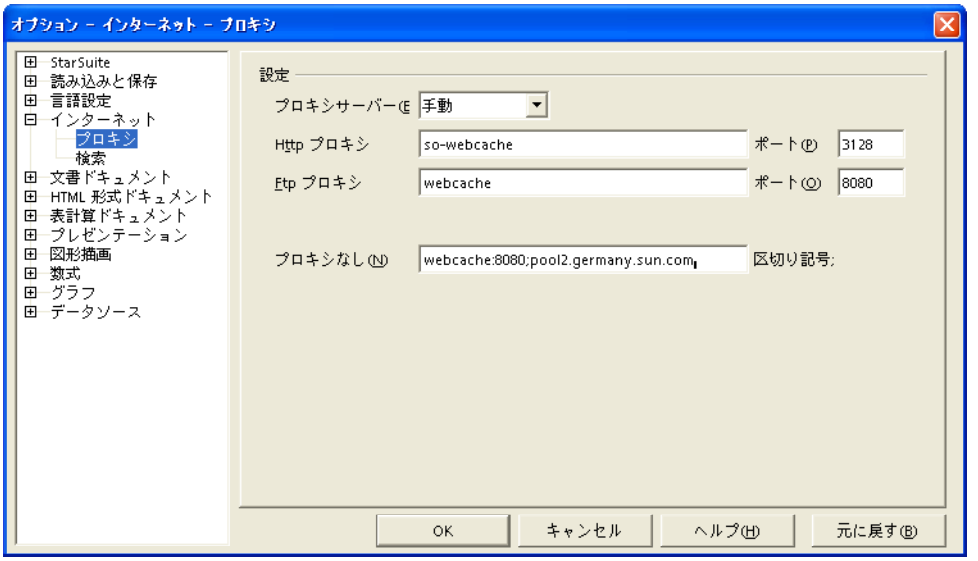
XML を使用すると、GUI の描画エンジン、オペレーティングシステム、およびプログラミング言語に依存しない定義を作成して、構成設定を管理できます。GUI は、テンプレートで指定する要素の意味の依存関係に基づいて描画されます。Desktop Manager のテンプレート形式は一般的なものであるため、GUI のあらゆる設計要求のソリューションになるわけではありません。たとえば、画面での正確な位置決めはサポートされていません。

この章では、テンプレートの典型的な作成過程について説明します。まず、デスクトップアプリケーションの既存の構成ダイアログから始めて、そのダイアログの単純なテンプレートを作成する方法を学びます。ファイルが「コンテンツ区画」に表示されるように、そのファイルを Desktop Manager で使用できるようにする方法についても学びます。

「Hello world!」テンプレート

▼ 「Hello world!」テンプレートの作成

始める前に　たとえば、StarSuite のプロキシ構成設定を Desktop Manager で使用したいとします。



次のテンプレートはGUIの最初の実装を提供します。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE apt:template SYSTEM "policytemplate.dtd">
<apt:template>
  <category apt:name="StarOffice" apt:label="StarSuite">
    <category apt:name="Internet" apt:label="インターネット">
      <page apt:name="Proxy" apt:label="プロキシ">
        <section apt:name="Settings" apt:label="設定">
          <property apt:name="ProxyServer" apt:label="プロキシサーバー"
                    apt:dataPath="org.openoffice.Inet/Settings/ooInetProxyType"
                    oor:type="xs:int">
            <visual apt:type="radioButtons"/>
            <constraints>
              <enumeration oor:value="0" apt:label="なし"/>
              <enumeration oor:value="2" apt:label="手動"/>
            </constraints>
          </property>
          <property apt:name="HTTPProxy" apt:label="Http プロキシ"
                    apt:dataPath="org.openoffice.Inet/Settings/ooInetHTTPProxyName"
                    oor:type="xs:string"/>
          <property apt:name="HTTPPort" apt:label="Http ポート"
                    apt:dataPath="org.openoffice.Inet/Settings/ooInetHTTPProxyPort"
                    oor:type="xs:int"/>
          <property apt:name="FTPProxy" apt:label="Ftp プロキシ"
                    apt:dataPath="org.openoffice.Inet/Settings/ooInetFTPProxyName"
                    oor:type="xs:string"/>
```

```
          <property apt:name="FTPPort" apt:label="Ftp ポート"
                    apt:dataPath="org.openoffice.Inet/Settings/ooInetFTPProxyPort"
                    oor:type="xs:int"/>
          <property apt:name="NoProxyFor" apt:label="プロキシなし"
                    apt:dataPath="org.openoffice.Inet/Settings/ooInetNoProxy"
                    oor:type="xs:string"/>
        </section>
      </page>
    </category>
  </category>
</apt:template>
```

新しいテンプレートを Desktop Manager に通知するには、次の手順が必要です。

1 **Desktop Manager** をインストールしたマシンに **root** としてログインします。

2 `/usr/share/webconsole/apoc/packages` に `HelloWorld/templates/StarSuite/Internet/Proxy` というディレクトリを作成します。

3 先ほどリストにした **XML** テンプレートの内容で `proxy.xml` というファイルを作成します。ファイルを `Proxy` ディレクトリにコピーします。

4 ユーザー **"noaccess"** に `Proxy` ディレクトリへの読み取り/実行アクセス権を与えます。

5 ユーザー **"noaccess"** に `proxy.xml` ファイルへの読み取りアクセス権を与えます。

6 **`/usr/sbin/smreg add -a /usr/share/webconsole/apoc`** を実行します。

7 `/usr/sbin/smcwebserver restart` コマンドを使用して **Web** サーバーを再起動します。

Desktop Manager にログインすると、「StarSuite 7」という新しいトップレベルのカテゴリが表示されます。そのカテゴリを下に参照すると、作成したテンプレートで定義した「プロキシ」ページが表示されます。

「Hello world!」テンプレートの検証

テンプレートの最初の 2 行は XML の初期定義です。3 行目には `apt:template` というテンプレートのルート要素があり、これにテンプレートファイルで作成したプロファイルの定義がすべて含まれています。

次の 4 行には、テンプレートの主要構成要素が含まれています。`apt:category` の要素をネストして、構成プロファイルツリーのノードを作成します。構成プロファイルツリーは、Desktop Manager の GUI にプロファイルの階層を視覚的に表します (32 ページの「ツリー」を参照)。`apt:name` 属性を指定すると、属性を使用してその要素が一意に示されます。`apt:label` 属性は、表示されるテキストを GUI のカテゴリとして指定します。`apt:label` 属性を指定しなければ、表示されるテキストは `apt:name` 属性によって定義され

ます。したがって、apt:label 要素は必ず指定してください。この要素はローカライズにも使用されます。詳細は、14 ページの「ローカライズ」を参照してください。

すべての apt:category 要素には、1 つまたは複数の apt:category 要素または apt:page 要素が含まれている必要があります。apt:page 要素は構成プロファイルツリーでリーフを表します。先ほど示した「プロキシ」ページはリーフの一例です。Desktop Manager ではページが単一の HTML ページとして表されるので、これを少なくとも 1 つの apt:section で分割する必要があります。apt:section 要素では、その子要素のすべてが見出し付きのテーブルで表されます。複数のセクションを使用すると、1 ページで設定をグループに分けることができます。

apt:section 要素には、構成設定を表す apt:property 要素が含まれています。「Hello, world!」テンプレートには、ProxyServer、 HTTPProxy、 HTTPPort、 FTPProxy、FTPPort、NoProxyFor の 6 つのプロパティーがあり、各プロパティーに apt:dataPath 属性が含まれています。これは必要な属性で、構成ツリーのデータの場所を指定します。構成ツリーは、構成リポジトリに保存されている構成設定の階層を表します。詳細は、32 ページの「ツリー」 を参照してください。

oor:type 属性は、構成リポジトリ内の構成設定のデータ型を定義します。ProxyServer、HTTPPort、および FTPPort は xs:int 型で、その他のプロパティーは xs:string 型です。整数型と文字列型は、デフォルトでは編集フィールドとして表示されます。

visual 要素は、Desktop Manager にプロパティーの表示方法を指示するために使用します。この要素を指定しなければ、プロパティー ProxyServer は、ラジオボタンのグループではなく編集フィールドを使って描画されます。この要素はオプションです。

ヒント – Desktop Manager の GUI は、ドロップダウンリストを使用する代わりに、整数値 2 候補をラジオボタンのグループとして表す点で、元の StarSuite の GUI と異なります。2 つの値を視覚化するには、ドロップダウンリストよりもラジオボタンを使用した方が操作性が向上します。たとえば、値を変更するために必要なクリックが 1 回か 2 回かの違いがあります。

---

constraints 要素は enumeration サブ要素とあわせて、描画するラジオボタンの数と、選択されるラジオボタンに応じてバックエンドに保存する整数値を指定するために使用します。apt:label 属性は、GUI に表示される各ラジオボタンの文字列を指定します。

ローカライズ

テンプレートで定義した文字列は、すべてローカライズすることが重要です。ローカライズした文字列はリソースファイルから取得されます。resImport は apt:template 要素のサブ要素で、1 つまたは複数のリソースファイルをテンプレートにバインドするために使用されます。次の例のように、リソースの完全修飾パスとそのベース名を指定する必要があります。

```
<apt:template>
<resImport
  apt:packagePath="com.sun.star.apoc.policies.resource.starsuite"/>
```

使用するリソースキーを定義するには、キーの名前を apt:label 属性の値として提供します。次はその例です。

```
<property apt:name="HTTPProxy" apt:label="SO.internet.proxy.name"
          apt:dataPath="org.openoffice.Inet/Settings/ooInetHTTPProxyName"
          oor:type="xs:string">
```

Desktop Manager は、テンプレートに結合されているすべてのリソースファイルから、apt:label 属性で指定されているキーを最初に探します。キーが見つからない場合は、apt:label 属性の値が表示されます。キーが見つかった場合は、リソースファイルから該当する値が取り出されて表示されます。

文字列の取得元となるリソースファイルは、Java に定義されているメカニズムと同様の方法で決定されます。すなわち、resImport 要素で指定されているパッケージのパスと、Web ブラウザで指定されている言語によって、選択されるリソースファイルが決まります。たとえば、ブラウザで選択されている Web ページの言語が en_US で、resImport 要素で指定されているパッケージのパスが com.sun.star.apoc.policies.resource.starsuite とすると、Desktop Manager は次のファイルの順に検索してリソースキーを探します。

```
./res/com/sun/star/apoc/policies/resource/starsuite_en_US.properties

./res/com/sun/star/apoc/policies/resource/starsuite_en.properties

./res/com/sun/star/apoc/policies/resource/starsuite.properties
```

Desktop Manager では、ローカルのプロファイルパッケージにあるファイルが先に検索されます (16 ページの「プロファイルパッケージ形式」を参照)。見つからない場合は、その他のすべてのパッケージが検索されます。この方法を用いると、ほかのパッケージですでにローカライズされている文字列を、特にカテゴリ名について再利用できます。

リソース検索の詳細は、Java ResourceBundle の API 仕様を参照してください。

ヒント – Java Web Console のアプリケーションはすべて、ログイン中に言語を判別します。アプリケーションに別の言語を使用させるには、いったんログアウトする必要があります。そのあとブラウザで Web ページの言語を変更してから、もう一度ログインします。

オンラインヘルプもローカライズする必要があります。Desktop Manager は、リソースファイルと同じルールに従って HTML ファイルを選択しますが、ヘルプファイルに対してはローカルのプロファイルパッケージのみが検索されるという点が異なります。たとえば、HTML ファイルのパスとして /StarSuite/Internet/Proxy、ブラウザで en_US を指定すると、Desktop Manager はオンラインヘルプファイル ./web/StarSuite/Internet/Proxy_en_US.html を表示します。

プロファイルパッケージ形式

テンプレートは配布コンテナに埋め込まれます。これは Java™ プログラミング言語の「パッケージ」に似ています。パッケージには、GUI ローカライズ用のリソースファイル、オンラインヘルプ用の HTML ファイル、独自のサポートファイルなど、オプションのファイルも含めることができます。

Desktop Manager はテンプレートや必要なオプションファイルにアクセスするために、特殊な形式のディレクトリ名とファイル名を使用しています。このディレクトリ名とファイル名の構造を「プロファイルパッケージ形式」と呼びます。

プロファイルパッケージはすべて `/usr/share/webconsole/apoc/packages` ディレクトリ下の一意のサブディレクトリに格納されます。たとえば「Hello, world!」の場合は、`HelloWorld` を選択した結果、`/usr/share/webconsole/apoc/packages/HelloWorld` ディレクトリになります。

ヒント–パッケージをインストールするソフトウェアの製品名と製品バージョンを使用してください。この方法で名前を付けると、パッケージディレクトリが他と重複することがありません。たとえば、`HelloWorld` よりも `HelloWorld3.1` の方が適切な名前です。

特定のパッケージディレクトリ (`packages` ディレクトリと混乱しないようにしてください) の下に次のサブディレクトリを作成できます。

templates	`templates` サブディレクトリには、プロファイルパッケージのテンプレートがすべて含まれていなければなりません。接尾辞が `.xml` のファイルはテンプレートと見なされます。ファイル名は `page` 要素の `apt:name` 属性の値と相関する必要があります。テンプレートは、そのカテゴリ階層で指定したのと同じディレクトリ階層にある限り、どのように分類してもかまいません。
classes	`classes` サブディレクトリには、プロファイルパッケージのクラスファイルがすべて含まれていなければなりません。接尾辞が `.class` のファイルは Java クラスファイルと見なされます。ファイルの名前は、そのファイルで定義されているクラスの名前と相関する必要があります。ファイルは、そのパッケージ階層で指定したのと同じディレクトリ階層になければなりません。
web	`web` サブディレクトリには、プロファイルパッケージの HTML ヘルプファイルすべてと、プロファイルパッケージが参照しているイメージが含まれていなければなりません。接尾辞が `.html` のファイルは HTML ファイルと見なされます。HTML ファイルの名前は、HTML ファイルを使用しているテンプレートの名前と相関付けてください。HTML ファイルは、HTML ファイルを使用しているテンプレートと同じディレクトリ階層に置きます。

res	res サブディレクトリには、プロファイルパッケージのリソースファイルがすべて含まれていなければなりません。接尾辞が .properties のファイルは Java 対応のリソースファイルと見なされます。リソースファイルの名前とパスは、それを使用するテンプレートの名前とパスに相関付けることができます。また、リソースファイルを 1 つ含んだディレクトリ階層 1 つをすべてのテンプレートに指定することもできます。
lib	lib サブディレクトリには、プロファイルパッケージのライブラリファイルがすべて含まれていなければなりません。接尾辞が .jar のファイルはライブラリと見なされます。ライブラリは Desktop Manager のクラスローダによって自動的に読み込まれます。ライブラリの内容にアクセスするには、 jar ファイルのルートディレクトリを絶対パスのルートディレクトリとして使用します。ライブラリファイルの典型的な用途は、クラスファイルやリソースファイルのコンテナとしての役割、および通常は classes ディレクトリと res ディレクトリ内にあるディレクトリのコンテナとしての役割です。

その他のファイルタイプは Desktop Manager にとって特別な意味はありません。ただし、クラスファイルや HTML ファイルで必要になる場合は、classes ディレクトリまたは web ディレクトリに入れることができます。

図 2–1 では、完成した HelloWorld パッケージを使用してパッケージ形式をさらに詳しく説明しています。

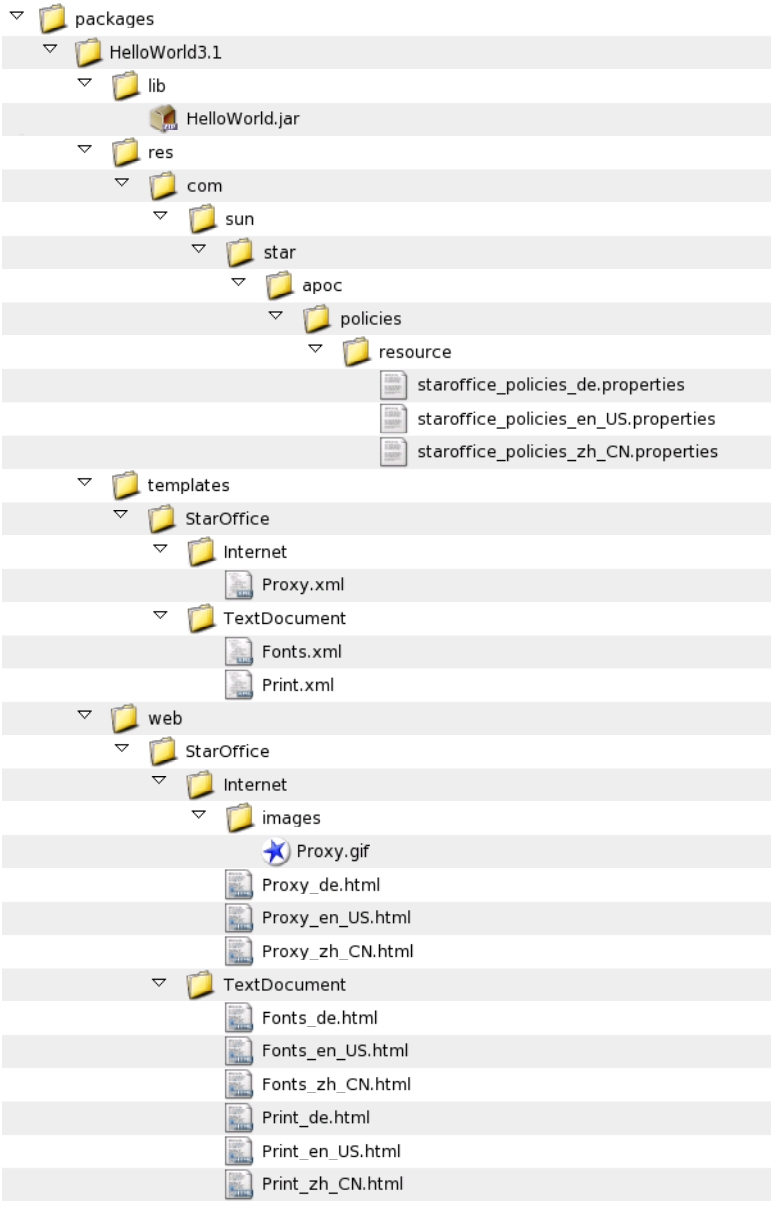


図 2–1 HelloWorld パッケージ

パッケージの配布は、この章で先に定義したルールに従う限り、パッケージの開発者が自由に決めることができます。ファイル一式に正しい場所へのコピー手順を添えて配布

したり、zip ファイルを提供したり、Solaris の場合は pkg ファイル、Linux の場合は .rpm ファイルなど、オペレーティングシステムが提供する配布メカニズムを使用することもできます。それぞれのオペレーティングシステムが提供しているソフトウェアの保守と削除の機能を強化して、エンドユーザーによりよいサポートを提供できる最後の方法をお勧めします。

◆ ◆ ◆ 第 3 章

3 高度なテンプレート

この章では、さらに複雑なテンプレートの作成方法と使い方を説明します。

セット

第 2 章で紹介した「Hello, World!」テンプレートを作成した後、必要に応じて、構成可能なプロキシのリストを動的にします。動的なリストがあると、GOPHER、SOCKS、SSL など、その他のプロトコルを処理するプロキシをユーザーが追加できます。このタスクを行うには、「セット」を使用します。

注 – プロキシページでのセットの使用は、例示のみを目的としています。実装には構成ツリーの元のレイアウトが必要になるため、StarSuite でも OpenOffice.org でもこの例を処理することはできません。

「新規作成...」ボタンをクリックすると、プロキシが追加されます。プロキシで処理する新しいプロトコルの名前を求めるダイアログが表示されます。最初に **FTP** と指定してから、もう一度「新規作成...」をクリックし、2 番目のプロトコルの名前として **HTTP** と入力します。表示される 2つのエントリはリンクです。そのリンクの一方をクリックすると、「コンテンツ区画」にコンテンツが読み込まれます。

この機能は「Hello, world!」を変更することによって実装されます。次のように変更して実装します。`HTTPProxy`、`HTTPPort`、`FTPProxy`、`FTPPort` の 4 つのプロパティーを削除します。そのあと、次のコード例の注釈セクションに示したセット要素を追加します。

```
<?xml version="1.0" encoding="UTF-8"?>
<DOCTYPE apt:template SYSTEM "policytemplate.dtd">
<apt:template>
  <category apt:name="StarOffice" apt:label="StarSuite">
    <category apt:name="Internet" apt:label="インターネット">
      <page apt:name="Proxy" apt:label="プロキシ">
        <section apt:name="Settings" apt:label="プロキシサーバー">
```

```
            <property apt:name="ProxyServer" apt:label="プロキシサーバー"
                      apt:dataPath="org.openoffice.Inet/Settings/ooInetProxyType"
                      oor:type="xs:int">
              <visual apt:type="radioButtons"/>
              <constraints>
                <enumeration oor:value="0" apt:label="なし"/>
                <enumeration oor:value="2" apt:label="手動"/>
              </constraints>
            </property>
            <property apt:name="NoProxyFor" apt:label="プロキシなし"
                      apt:dataPath="org.openoffice.Inet/Settings/ooInetNoProxy"
                      oor:type="xs:string"/>
          </section>
      <!-- Beginning of set element to be added -->
          <set apt:name="ProxyList" apt:label="プロキシリスト"
               apt:dataPath="org.openoffice.Inet/Settings/ooInetProxyList">
            <page apt:name="ProxyPage" apt:label="プロキシ">
              <section apt:name="Proxy" apt:label="ホストおよびポート">
                <property apt:name="HostName" apt:label="ホスト名"
                          apt:dataPath="./$queriedId/HostName"
                          oor:type="xs:string"/>
                <property apt:name="Port" apt:label="ポート"
                          apt:dataPath="./$queriedId/Port"
                          oor:type="xs:string"/>
              </section>
            </page>
          </set>
      <!-- End of added set element -->
        </page>
      </category>
    </category>
</apt:template>
```

set 要素の apt:dataPath 属性は、セットがバックエンドに保存されている場所を指しています。set 要素には page 要素が含まれ、page 要素には section 要素、section 要素には property 要素が含まれています。この階層はカテゴリ要素の下の要素の階層と相関関係があります。同じようにページとして描画されますが、セットテーブルのリンクをクリックしてトリガーする点が異なります。

カテゴリページと比較すると、セットページのプロパティー HostName と Port は apt:dataPath に特殊な表記法を使用しています。パスはドットで始まります。これは、要素の階層を上に検索して最初に見つかったパスの定義に相対するパスという意味です。apt:dataPath で最初の親要素はセット要素のため、Desktop Manager は相対パス、たとえば Port プロパティーを org.openoffice.Inet/Settings/ooInetProxyList/$queriedId/Port に翻訳します。このパスのもう 1 つの特徴は $queriedId 変数です。構成リポジトリ内のデータはすべて他と識別できなければならないので、動的データ構造の各要素に一意の名前を付ける必要があります。$queriedId 変数は、「追加」ボタンがクリックされたときに、その名前のユー

ザーを問い合わせるよう Desktop Manager に指示します。その結果生成されるセット要素は、変数で判別される位置に、指定した名前で格納されます。したがって、セット要素 FTP の場合は、そのポートプロパティーのパスは org.openoffice.Inet/Settings/ooInetProxyList/FTP/Port です。

別の例 : セットを使用して、NoProxyFor プロパティーを表示することもできます。ホスト名の文字列が長くなると、このプロパティーの編集フィールドの使用に問題が生じます。その場合、ユーザーはスクロールして編集フィールドの文字列全体を表示しなければなりません。セットで実装されるプロキシ名のリストによって、この余分のスクロール操作がなくなります。

NoProxyFor プロパティーの文字列の代わりにセットを使用するには、NoProxyFor 要素をそのサブ要素すべてと一緒に削除します。そのあと、次のコード例の注釈セクションに示したセット要素を追加します。

```
<?xml version="1.0" encoding="UTF-8"?>
<DOCTYPE apt:template SYSTEM "policytemplate.dtd">
<apt:template>
  <category apt:name="StarOffice" apt:label="StarSuite">
    <category apt:name="Internet" apt:label="インターネット">
      <page apt:name="Proxy" apt:label="プロキシ">
        <section apt:name="Settings" apt:label="プロキシサーバー">
          <property apt:name="ProxyServer" apt:label="プロキシサーバー"
                    apt:dataPath="org.openoffice.Inet/Settings/ooInetProxyType""
                    oor:type="xs:int">
            <visual apt:type="radioButtons"/>
            <constraints>
              <enumeration oor:value="0" apt:label="なし"/>
              <enumeration oor:value="2" apt:label="手動"/>
            </constraints>
          </property>
        </section>
    <!-- Beginning of set element to be added -->
        <set apt:name="NoProxyFor" apt:label="プロキシなし"
             apt:dataPath="org.openoffice.Inet/Settings/ooInetNoProxySet">
          <page apt:name="HostNamePage">
            <section apt:name="HostNameSection">
              <property apt:name="HostNameProp"
                        apt:dataPath="./$queriedId/HostName" oor:type="xs:string"
                        apt:storeDefault="true">
                <visual apt:type="hidden"/>
                <value>$queriedId</value>
              </property>
            </section>
          </page>
        </set>
    <!-- End of set element to be added -->
        <set apt:name="ProxyList" apt:label="プロキシリスト"
```

```
            apt:dataPath="org.openoffice.Inet/Settings/ooInetProxyList">
          <page apt:name="ProxyPage" apt:label="プロキシ">
            <section apt:name="Proxy" apt:label="ホストおよびポート">
              <property apt:name="HostName" apt:label="ホスト名"
                        apt:dataPath="./$queriedId/HostName"
                        oor:type="xs:string"/>
              <property apt:name="Port" apt:label="ポート"
                        apt:dataPath="./$queriedId/Port"
                        oor:type="xs:string"/>
            </section>
          </page>
        </set>
      </page>
    </category>
  </category>
</apt:template>
```

テーブルのエントリはリンクではなく、一価要素の単なるリストを表すようになります。これを行うには、`apt:storeDefault` 属性と `visual` 要素を `value` 要素とあわせて使用します。`value` 要素は構成設定のデフォルト値を定義できます。デフォルト値は、デフォルトでは構成リポジトリに保存されません。`apt:storeDefault` 属性は、そのデフォルトを上書きして、バックエンドにデフォルト値を自動的に保存するよう Desktop Manager に指示します。この場合、デフォルト値は新しいセット要素が追加されるときにユーザーがダイアログで入力する値です。`visual` 要素の `apt:type` 属性を「hidden (非表示)」と指定すると、そのページの唯一のセクションは空のままになります。セットのページが空の場合は、ページを表示する意味がないので、Desktop Manager はリンクを提供しません。

アクションハンドラ

アクションハンドラは、イベントが発生するときにユーザー定義のアクションを実行するために使用されます。その時点で、使用可能なアクションハンドラは XML ハンドラ 1 つだけです。XML ハンドラはクライアント側のブラウザで JavaScript コードを生成します。

XML ハンドラを使用すると、まだテンプレートに含まれていない StarSuite/ OpenOffice.org「プロキシ」ダイアログの機能を実装することもできます。「プロキシサーバー」オプションに「なし」の値を選択すると、編集フィールドが無効になります。

次のテンプレートの注釈領域は、「プロキシサーバー」オプションが「手動」か「なし」に設定されている場合に、編集フィールドを有効または無効にするために、元の「Hello, world!」テンプレートに加える必要のある変更を示しています。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE apt:template SYSTEM "policytemplate.dtd">
<apt:template>
```

```
<category apt:name="StarOffice" apt:label="StarSuite" >
  <category apt:name="Internet" apt:label="インターネット">
    <page apt:name="Proxy" apt:label="プロキシ">
      <section apt:name="Settings" apt:label="設定">
        <property apt:name="ProxyServer" apt:label="プロキシサーバー"
                  apt:dataPath="org.openoffice.Inet/Settings/ooInetProxyType"
                  oor:type="xs:int"
            <!-- The following line should be added to original "Hello, world!" template -->
                  apt:xmlHandler="switchState">
          <visual apt:type="radioButtons"/>
          <constraints>
            <enumeration oor:value="0" apt:label="なし"/>
            <enumeration oor:value="2" apt:label="手動"/>
          </constraints>
        </property>
        <property apt:name="HTTPProxy" apt:label="Http プロキシ"
                  apt:dataPath="org.openoffice.Inet/Settings/ooInetHTTPProxyName"
                  oor:type="xs:string"/>
        <property apt:name="HTTPPort" apt:label="Http ポート"
                  apt:dataPath="org.openoffice.Inet/Settings/ooInetHTTPProxyPort"
                  oor:type="xs:int"/>
        <property apt:name="FTPProxy" apt:label="FTP Proxy"
                  apt:dataPath="org.openoffice.Inet/Settings/ooInetFTPProxyName"
                  oor:type="xs:string"/>
        <property apt:name="FTPPort" apt:label="FTP Port"
                  apt:dataPath="org.openoffice.Inet/Settings/ooInetFTPProxyPort"
                  oor:type="xs:int"/>
        <property apt:name="NoProxyFor" apt:label="No Proxy For"
                  apt:dataPath="org.openoffice.Inet/Settings/ooInetNoProxy"
                  oor:type="xs:string"/>
      </section>
  <!-- Beginning of section to be added to original "Hello, world!" template -->
      <xmlHandler apt:name="switchState">
        <event apt:type="onChange" />
        <action>
          <choose>
            <when apt:test="ProxyServer.value=0">
              <command>HTTPProxy.enabled=false</command>
              <command>HTTPPort.enabled=false</command>
              <command>FTPProxy.enabled=false</command>
              <command>FTPPort.enabled=false</command>
              <command>NoProxyFor.enabled=false</command>
            </when>
            <otherwise>
              <command>HTTPProxy.enabled=true</command>
              <command>HTTPPort.enabled=true</command>
              <command>FTPProxy.enabled=true</command>
              <command>FTPPort.enabled=true</command>
```

```
                <command>NoProxyFor.enabled=true</command>
              </otherwise>
            </choose>
          </action>
        </xmlHandler>
    <!-- End of section to be added -->
      </page>
    </category>
  </category>
</apt:template>
```

apt:xmlHandler 属性をプロパティー ProxyServer に追加すると、同じ名前 (ここでは「switchState」) の xmlHandler 要素がそのプロパティに関連付けられます。

アクションハンドラはイベントによってトリガされます。アクションハンドラの対象となるイベントは、イベント要素の apt:type 属性で定義します。この時点で使用できるイベントは、onChange イベントだけです。このイベントは、ユーザーがプロパティーの新しいデータを入力したときに発行されます。前の例では、ProxyServer プロパティーの値が変わったときに XML ハンドラをトリガするためにイベントが使用されています。

アクション要素には、イベント要素で指定されているイベントが発生したときに実行されるアクションが含まれています。前の例で、最初のアクションは ProxyServer プロパティーの値をチェックし、それに応じてほかの編集フィールドの状態を変更することです。これには、choose、when、および otherwise 要素を使用します。ProxyServer プロパティーを「なし」に設定した場合は、すべての編集フィールドが無効になります。ProxyServer プロパティーを「手動」に設定した場合は、編集フィールドが有効になります。

ヘルプ

ヘルプにはオンラインヘルプとインラインヘルプの 2 種類があります。

オンラインヘルプは、ユーザーがマストヘッドの「ヘルプ」リンクをクリックしたときに別のウィンドウに表示されるコンテキスト依存の詳しいヘルプです。前回の操作を「コンテンツ区画」で行なった場合は、現在「コンテンツ区画」に表示されているテンプレートで定義されているオンラインヘルプのドキュメントが表示されます。前回の操作をそれ以外の場所で行なった場合は、一般的なオンラインヘルプが表示されます。page 要素の apt:onlineHelp 属性は、HTML ヘルプファイルをテンプレートに結合するために使用されます。この場合、たとえば次のように、ファイルの完全修飾パスとそのベース名を指定する必要があります。

```
<page apt:name="Proxy">
      apt:label="プロキシ"
          <apt:onlineHelp="/StarSuite/Internet/proxy">
```

これは ./web/StarSuite/Internet/proxy.html を参照します。

HTML ファイルでは絶対パスと相対パスを使用できます。HTML ファイルと一緒に images というディレクトリのイメージを配置する場合は、HTML ファイルで次のいずれかの注釈を使用できます。

- `<img src="/StarSuite/Internet/images/image.gif">`
- `<img src="./images/image.gif">`

インラインヘルプは、カテゴリ、ページ、プロパティーの各ページに補足情報を提供する短いテキストです。インラインヘルプは category 要素では「コメント」列、page 要素ではページタイトルの下、 property 要素では構成設定の下に表示されます。ヘルプテキストは、この category、 page、 property の 3 要素のいずれかの apt:inlineHelp 属性によって指定されます。次に例を示します。

```
<property apt:name="HTTPProxy"
          apt:label="Http プロキシ"
          apt:inlineHelp="Specify no proxy (None) or a manually defined proxy (manual)."
          apt:dataPath="org.openoffice.Inet/Settings/ooInetHTTPProxyName"
          oor:type="xs:string"
</property>
```

ローカライズを円滑にするため、ラベルにリソースキーを与えておきます。

◆ ◆ ◆ 第 4 章

4 設計に関する推奨事項

アプリケーションのテンプレートを作成する前に、Desktop Manager が一元管理する対象の構成設定を決定する必要があります。もっとも単純な解決法は、使用する可能性のある構成設定をすべて含むテンプレートを作成することです。しかし、この方法では不必要な作業が増える可能性があり、決して使用されることのない設定を Desktop Manager が表示するという結果になりがちです。

ガイドライン

テンプレートの作成中にどのオプションを選ぶかは、対象とするアプリケーションによりますが、使用するオプションの目安として次のリストを参考にしてください。

- セキュリティを扱うオプション
- ロックダウンを扱うオプション
- ホスト、ポート、URL、パスなどのリソースを参照するオプション
- フォントや色などの企業イメージの定義に使用するオプション

カテゴリとページの具体的な名前は、テンプレートの作成者が自由に決めます。守るルールは次の 2 つだけです。

- 一意のパスを作成できるように、ツリーの全レベルで重複のない名前にする
- 最初のカテゴリで、アプリケーションとそのバージョンを識別する。たとえば、「StarSuite 6.0」のように指定する

`apt:section` 要素を選ぶか、 `apt:page` 要素を選ぶかは、次の 2 つの要件によって異なります。すなわち、1 ページ当たりの構成項目数を 6 つ以内に制限して、「コンテンツ区画」でのスクロールを避けます。既存の GUI を Desktop Manager の GUI にマッピングしている場合は、使い易さを認識によって最適化するために、できるだけ正確にアプリケーションの GUI を Desktop Manager の GUI にマッピングします。この 2 つの要件が相反する場合は、後者を選んでください。アプリケーションの GUI からの Desktop Manager の GUI の逸脱を考慮している場合は、それを小さなものにすべきです。

GUI に表示されるテキストはすべて見た目が一貫している必要があります。GUI の設計要素に表示するテキストには、次の大文字使用のガイドラインを適用してください。

- インラインヘルプとオンラインヘルプでは、文単位で大文字を使用する。常に大文字を使用する固有名詞、略語、頭字語がテキストに含まれている場合を除いて、各文の最初の文字だけを大文字にする
- 文中や文末に正しく句読点を入れる
- 完全な文でない長い句を用いるのを避ける。完全な文ではない句を用いる必要がある場合は、最後に句読点は不要
- ラベル、タイトル、チェックボックスのテキスト、メニュー、リスト項目などは大文字を見出し形式で使用する。見出し形式を適用するには、不定詞 (a、an、the)、接続詞 (and、or、but、so、yet、nor など)、および 3 文字以内の前置詞 (in など) を除いて、各単語の最初の文字を大文字にする。ただし、冒頭と末尾の単語はその品詞に関わらず常に最初を大文字にする
- ラベルの後にセミコロンを付けない
- アプリケーション内で一貫させる

一般に、パッケージには必要なものが揃っていますが、次の 2 つの例外があります。ほかのパッケージのリソースを再利用できます(14 ページの「ローカライズ」を参照)。また、ほかのパッケージのチューザ定義を `apt:extendsChooser` 属性で再利用できます (43 ページの「Basic Data 要素: property, value, constraints」を参照)。この種の参照は慎重に使い、多用しないでください。ほかのパッケージを参照すると、そのパッケージと依存関係が生じますが、依存しているパッケージが Desktop Manager のすべてのインストールに含まれているとは限りません。

Desktop Manager は、リソースを回復するために、インストールされている全パッケージを走査する可能性があるため、リソースキーはパッケージ内だけではなくほかのパッケージと比べた場合も一意のものでなければなりません。カテゴリ階層を使用してローカルのリソースキーに接頭辞を付けるか、Java のパッケージ構造に似た構造や製品名を使うなど、独自の階層的接頭辞を作成します。

オンラインヘルプは、標準パッケージに付属の HTML ファイルに似た設計にする必要があります。ブラウザの検出と CCS の定義が正しく行われるように、次の行を含めます。

```
<script type="text/javascript" src="/com_sun_web_ui/js/browserVersion.js">
</script>
<script type="text/javascript" src="/com_sun_web_ui/js/stylesheet.js">
/com_sun_web_ui/js/stylesheet.js"></script>
```

タイトルのレイアウトには、"help-header-1"、"help-header-2"、および "help-header-3" のスタイルを使用します。

◆ ◆ ◆ 第 5 章

5 構成の概念

このマニュアルで紹介する各種ツリーは、管理者ガイドに含まれているものとは異なります。Desktop Manager を使用するうえで、構成ツリーと構成プロファイルツリーという 2 種類のツリーに関する知識は不要のため、管理者ガイドは構成ツリーについては触れていません。

階層

クライアント側から見ると、アプリケーションは 3 つの別々のデータソース (階層) から構成データを取得します。これらは、デフォルト階層、ユーザー階層、およびプロファイル階層です。

ユーザー階層とデフォルト階層は、クライアントのアプリケーションが現在使用している既存のデータソースです。デフォルト階層はアプリケーションと一緒に配備され、配備後は、ほとんど変わることはありません。これはアプリケーションと一緒にローカルに格納されています。ユーザー階層は、特定のユーザーがアプリケーションの設定に加えた変更を保存します。これはローカルまたは共有の場所に格納されています。

プロファイル階層は、構成リポジトリに一元的に格納されています。このリポジトリは Desktop Manager が管理する構成設定を含んでいます。これらの設定はサーバーでは、組織、役割、ユーザー、ホストなどの要素に関連付けられています。特定のユーザーやホストに代わってConfiguration Manager がこれらにアクセスし、ユーザーやホストには読み取り専用になっています。

Desktop Manager は、プロファイル階層の構成設定のみを読み書きできます。デフォルト階層またはユーザー階層のコンテンツには、Desktop Manager からアクセスできません。すべての階層の値の取得と組み合わせは、クライアントのアプリケーション構成システムが担当します。33 ページの「結合」を参照してください。

ツリー

Desktop Manager は、「ツリー」 とも呼ばれる、4 つの異なる階層構造を取り扱います。Desktop Manager がどのように機能するかを理解するには、各種ツリーを区別することが重要です。

最初のツリーは「組織ツリー」 (図 5–1 のグレーの部分で、組織単位間の関係を表しています。このツリーの最初のレベルは組織自体を表します。その次のレベルは、たとえば部門や課を表すことができます。最後のレベルは、これらの部署のメンバーを表すことができます。

2 番目のツリーは「ドメインツリー」で、ドメインやホストなどのネットワーク要素間の関係を表しています。このツリーの最初のレベルは、ネットワーク全体を表します。その次のレベルは、たとえば各種サブネットを表し、最後のレベルはこれらのサブネット内の実際のホストを表します。

Desktop Manager では、上記の 2 つのツリーが現在 LDAP サーバーのコンテンツ (企業構造の代表的なリポジトリ) を解釈することによって取得されています。LDAP でツリー内の各場所は「要素」と呼ばれます。LDAP サーバー内のエントリは、Desktop Manager によって認識される要素、すなわち「組織」、「役割」、「ユーザー」、「ドメイン」、および「ホスト」にマッピングされます。

3 番目のツリーは「構成ツリー」で、図 5–1 の青で表した部分です。構成ツリーは、バックエンドの構成設定を階層的にグループ化します。構成ツリーのトップレベルはコンポーネントです。コンポーネントは、1 つのソフトウェアコンポーネントを構成する構成設定で構成されています。コンポーネントの下にある要素はすべてノードかプロパティーです。ノードにはノードまたはプロパティーを含むことができます。プロパティーには構成設定が含まれています。構成設定のそれぞれはパスで表されます。たとえば、`org.openoffice.Office.Common/ExternalMailer/Program` は、「Common」コンポーネントの下の「External Mailer」ノードにある「Program」構成設定を表します。

組織ツリーとドメインツリーの各要素は独自の構成ツリーを持つことができるため、「ツリーのツリー」が 2 つ になります (構成ツリーが含まれた組織ツリーと、構成ツリーが含まれたドメインツリー)。

4 番目のツリーは「構成プロファイルツリー」で、図 5–1 の黄色の部分です。構成プロファイルツリーは、参照や編集がしやすいように構成設定を視覚的に分類するために使用します。これは、構成ツリーの階層から完全に独立した階層を定義して行います。構成プロファイルツリーに表示される実際の値は、構成ツリーの構成設定の場所を参照して取得されます。図 5–1 の矢印を参照してください。このようにすると、GUI とバックエンドのデータの異なる設計要件を分離できます。たとえば、構成設定の位置はバックエンドよりも GUI の方が早く変わります。

構成プロファイルツリーのトップレベルにはアプリケーションがあり、次のレベルはそのアプリケーションの各種モジュールやサブモジュール、最後のレベルは実際の構成設定に対応しています。多数のオプションを処理する StarSuite™ や Mozillaのような構成システムでも同様に表されます。たとえば、HomeUrl オプションは「設定」ダイアログの Mozilla/Navigator/HomeUrl にあります。

注 – このマニュアルで紹介する各種ツリーは、『Sun Desktop Manager 1.0 管理ガイド』に含まれているものとは異なります。Desktop Manager を使用するうえで、構成ツリーと構成プロファイルツリーという 2 種類のツリーに関する知識は不要のため、管理者ガイドでは構成ツリーについては触れていません。

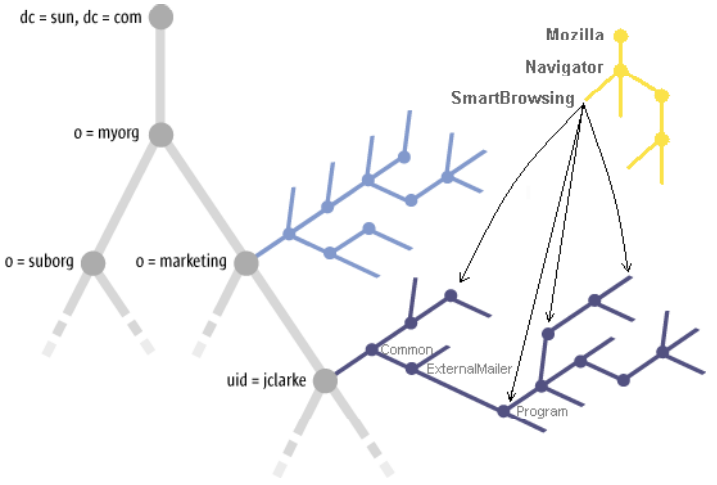


図 5–1 「ツリーのツリー」

結合

特定の要素に最終的に使用される構成設定は、その要素の構成定とその親要素の構成設定をクライアント側でマージして決定されます。たとえば、あるユーザー用の設定値では、そのユーザーに割り当てられたプロファイルとともに、そのユーザーが所属している組織に割り当てられたプロファイルが考慮されます。マージは継承によって実現します。すなわち、ユーザーは組織構造の上のレベルで指定されている設定を継承します。このプロセスについては、図 5–2 を参照してください。「marketing」組織の設定をそのメンバーの 1 人である「jclarke」が継承する仕組みを表しています。ユーザー「jclarke」の構成設定が、継承された設定の一部を上書きします。

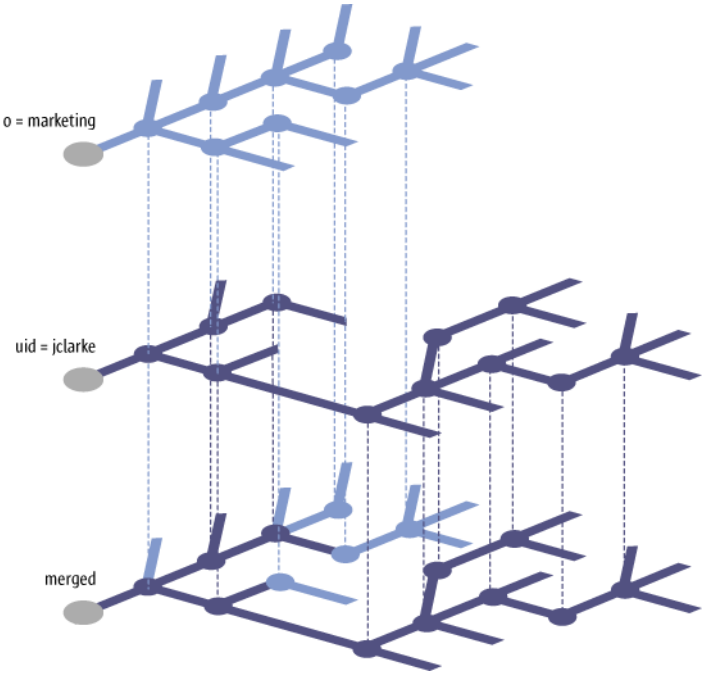


図 5–2 マージ

構成設定がプロファイル階層にマージされる場合と同様に、3 つの階層がマージされて、最終的な構成設定一式が形成されます。ユーザー階層はプロファイル階層より優先され、プロファイル階層はデフォルト階層より優先されます。ユーザー階層の構成設定がマージ処理中には考慮されないように、またユーザー自身のクライアントマシン上で Desktop Manager を使って管理者によって行われた設定に上書きすることが許可されないように、構成設定をプロファイル階層にマーキングすることが可能です。これを「保護」と呼んでいます。

ユーザーベースとホストベースの構成

組織団体ツリーとドメインツリーを使用した作業の考え方は同じです。この 2 つの大きな違いは、組織ツリーがユーザーで構成され、ドメインツリーがホストで構成されることです。ユーザーとホストを 2 つの別々のツリーに置くことによって、Desktop Manager はユーザーベースおよびホストベースの構成を提供することが可能です。

クライアント側では、ユーザーベースの構成設定は、ユーザー名に基づいて組織ツリーから取得されます。ホストベースの構成設定は、ユーザーが現在作業を行なっているホストの IP またはホスト名に基づいて、ドメインツリーから取得されます。ユーザー設定はホスト設定のあとでマージされます。すなわち、ユーザーの設定がホストの設定より

優先されます。たとえば、この 2 種類の構成を提供すると、ローミングユーザーはユーザーベースの構成 1 つを使用できるだけでなく、作業中のホストに最適なプロキシ構成を利用することもできます。

◆ ◆ ◆ 付 録 A

構成パスのマッピング

Desktop Manager の構成リポジトリの重要な設計照準の 1 つは、できるだけ多くの既存の構成形式による構成データのリポジトリとして機能できるような柔軟な設計です。

リポジトリに使用される構成形式は、構成データを「コンポーネント」に分割します。コンポーネントとは構成設定の集合です。1 つのコンポーネントに属する設定は、通常は一緒に使用され、相互に関連している可能性があります。多くの場合、特定のクライアントアプリケーション、ソフトウェアモジュール、アプリケーションドメインなどに関連付けられています。コンポーネントはその名前で識別され、たとえば `org.openoffice.Inet` のような構造的な名前を使用してパッケージの階層にグループ化されます。

各コンポーネントは階層構造になっています。この構造は「ノード」と「プロパティー」で構成されています。ノードは、他のノードやプロパティーのコンテナの役割をする構成要素です。プロパティーは階層のリーフ要素で、1 つまたは複数の値が含まれています。ノードとプロパティーは名前で識別されるため、その親ノード内で一意でなければなりません。一意な名前があれば、そのコンポーネントとパス (たとえば `org.openoffice.Inet/Settings/ooInetProxyType`) によって、どのノードやプロパティーでも参照できます。

他の構成システムでは異なる構成形式が使用されています。他の構成システムの構成データは、APOC 構成形式を使用して構成レジストリに保存されますが、アプリケーションに対応する構成形式で渡す必要があります。APOC 形式を、アプリケーションが使用している形式に 1 対 1 でマッピングする必要があります。構文のマッピングは、対応 APOC アダプタによってサイレントに実行されます。テンプレート開発者に残された唯一のタスクは、構成データを構成プロファイルツリーに保存するときに正しい構成パスのマッピングを使用することです。これらのマッピングは、中央の構成リポジトリでクライアントのさまざまな構成システムの `config` オプションを区別するために必要です。次の構成システムには、アダプタによって定義および考慮される具体的なマッピングが定義されています。

- StarSuite/OpenOffice.org Registry (OOR。StarSuite と OpenOffice.org に使用)
- GNOME Configuration (GConf。GNOME アプリケーションに使用)
- Java Preferences (Java プログラムに使用)

- Mozilla Preferences (Mozilla に使用)

StarSuite/OpenOffice.org Registry (OOR)

OOR キー命名スキーマは Desktop Manager 構成リポジトリで使用されるスキーマなので、この構成システムでは適用の作業は必要ありません。

GNOME Configuration (GConf)

GConf の構成要素を APOC のコンポーネントとパスに変換するために、次のマッピングが実行されます。

- GConf 関連のコンポーネントにはすべて `org.gnome` の接頭辞が付けられる
- `/apps/<subdir>/...` はコンポーネントの接尾辞 `apps.<encoded subdir>` にマッピングされる
- `/desktop/<subdir>/...` はコンポーネントの接尾辞 `desktop.<encoded subdir>` にマッピングされる
- `/system/<subdir>/...` はコンポーネントの接尾辞 `system.<encoded subdir>` にマッピングされる
- `/extra/<subdir>/...` はコンポーネントの接尾辞 `extra.<encoded subdir>` にマッピングされる
- `/extra/<keyname>/...` はコンポーネントの接尾辞 `extra` にマッピングされる
- 命名規則に従っていないキーは、`subdir` と `ooc` があれば、コンポーネントの接尾辞 `ooc.<encoded subdir>` にマッピングされる
- `/schemas/<keypath>` はコンポーネント部品のスキーマにマッピングされ、上記の規則が残りのキーに適用される
- コンポーネント部品にマッピングされた GConf キーの subdirs はコンポーネント部品の制約に従ってエンコードされる

例 A–1 GNOME Configuration

- `/apps/myapplication/sampleSub.Dir/sampleSetting` は `org.gnome.apps.myapplication/sampleSub.Dir/sampleSetting` になる
- `/desktop/sampleDir/sampleSub.Dir/sampleSetting` は `org.gnome.desktop.sampleDir/sampleSub.Dir/sampleSetting` になる
- `extra/sampleSetting` は `org.gnome.extra/sampleSetting` になる
- `/sample.Dir/sampleSetting` は `org.gnome.ooc.sample.Dir/sampleSetting` になる
- `/schemas/apps/gnome-setting/sampleSubDir/sampleSetting` は `org.gnome.schemas.apps.gnome-setting/sampleSubDir/sampleSetting` になる

Java Preferences

Java Preferences のノード / キーのペアを APOC のコンポーネントとパスに変換するために、最初の 3 つのノードパス要素 (3 つより少ない場合はすべてのノードパス要素) が `java.prefs` に付加されてコンポーネント名が形成され、残りのノードパス要素とキーでパスが形成されます。ユーザー設定のみが考慮されます。

例 **A–2** Java Preferences

- ノード `/com/sun/star/configuration`、キー `someKey` は `java.prefs.com.sun.star/configuration/someKey` になる
- ノード `/com/acme/widget`、キー `someKey` は `java.prefs.com.acme.widget/someKey` になる
- ノード `/sample.Dir`、キー `someKey` は `java.prefs.sample.Dir/someKey` になる

Mozilla Preferences

Mozilla の構成要素を APOC のコンポーネントとパスに変換するために、最初の要素 (名前に複数の要素が含まれていると想定) が `org.mozilla` に付加されてコンポーネント名が形成され、残りの名前がノードのパスとして使用されます。要素が 1 つだけの設定は、`org.mozilla.ooc` のそれぞれの名前の下に保存されます。

例 **A–3** Mozilla Preferences

- `mail.server.default.isSecure` は `org.mozilla.mail/server/default/isSecure` になる
- `sampleSetting` は `org.mozilla.ooc/sampleSetting` になる

◆ ◆ ◆ 付 録 B

要素の辞書

この付録では、テンプレートで使用可能な要素と属性すべてのリファレンスを提供します。

Header 要素: apt:template, resImport, helpImport

```
<!ELEMENT apt:template (resImport*, category)>
<!ATTLIST apt:template
    xmlns:apt CDATA #FIXED "http://www.sun.com/jds/apoc/2004/template"
    xmlns:oor CDATA #FIXED "http://openoffice.org/2001/registry"
    xmlns:xs  CDATA #FIXED "http://www.w3.org/2001/XMLSchema"
    xmlns:xsi CDATA #FIXED "http://www.w3.org/2001/XMLSchema-instance"
>

<!ELEMENT resImport EMPTY><!ATTLIST resImport
    apt:packagePath NMTOKEN #REQUIRED
>
```

ルート要素テンプレートには 2 つのサブ要素 `resImport` と `category` があり、これらについては、42 ページの「Structure 要素: category, page, section」で説明しています。

`resImport` 要素はリソースファイルのインポートに使用します。インポートされたリソースバンドルのリソースキーは、すべてテンプレートに通知されます。リソースキーを使用するリソースを、たとえば `apt:label` 属性にインポートする必要があります。42 ページの「Structure 要素: category, page, section」を参照してください。`apt:packagePath` 属性は、パスを使用してリソースの場所を指定します。区切り文字はドット (.) です。ファイルの接尾辞 (`.properties`) は、ISO 言語コード (ISO-639) および ISO 国コード (ISO 3166) と同様に指定してはいけません。パスのルートディレクトリは、`package` の下にある `res` ディレクトリです。14 ページの「ローカライズ」も参照してください。

Structure 要素: category, page, section

```
<!ELEMENT category (category | page)>
<!ATTLIST category
    apt:name ID #REQUIRED
    apt:scope (user | host | global) #IMPLIED
    apt:label NMTOKEN #IMPLIED
    apt:inlineHelp NMTOKEN #IMPLIED
>
<!ELEMENT page ((section | set)+, xmlHandler*)><!ATTLIST page
    apt:name ID #REQUIRED
    apt:scope (user | host | global) #IMPLIED
    apt:label NMTOKEN #IMPLIED
    apt:inlineHelp NMTOKEN #IMPLIED
    apt:onlineHelp CDATA #IMPLIED
>
<!ELEMENT section (property+)>
<!ATTLIST section
    apt:name ID #REQUIRED
    apt:scope (user | host | global) #IMPLIED
    apt:label NMTOKEN #IMPLIED
>
```

category 要素は、構成プロファイルツリーでページの一意な配置を定義するために使用します。その最初の属性は apt:name 属性です。name 属性は、要素の一意な名前を定義するために使用します。大きいテンプレートの方向付けと要素の参照を円滑にします。

category 要素の 2 番目の属性は apt:scope です。scope 属性は、構成設定を適用できるツリーを指定します。スコープが "user" の場合は、組織ツリーのみに構成設定が適用されます。スコープが "host" の場合は、ドメインリーのみに構成設定が適用されます。スコープが "global" の場合は、両方のツリーに構成設定が適用されます。デフォルト設定は "global" です。要素は、独自のスコープを定義する場合を除いて、親要素からスコープを継承します。要素のスコープが "user" で、ドメインツリーに接続している構成プロファイルツリーが「コンテンツ区画」に表示されている場合、その要素はユーザーに表示されません。要素のスコープが "host" で、組織ツリーに接続している構成プロファイルツリーが表示されている場合も、同様です。

category 要素の 3 番目の属性は apt:label です。label 属性は、ユーザーに表示可能な要素の名前を指定し、ローカライズをサポートしています。label 属性で指定される文字列は、最初にリソースバンドルで検索されます。文字列と一致するキーが見つかった場合は、その値が GUI に表示されます。どのリソースバンドルにも文字列と一致するキーがない場合は、その文字列が GUI に表示されます。label 属性を指定しなければ、name 属性で指定された文字列が GUI に表示されます。属性を両方とも定義しない場合は、出力が表示されません。

category 要素の 4 番目の属性は apt:inlineHelp です。inlineHelp 属性は、GUI に表示されるヘルプテキストを指定します。ヘルプはカテゴリ名の右にある「コメント」列に表示されます。前述の label 属性と同様に、ローカライズをサポートしています。

カテゴリ階層の終わりに 1 つだけページ要素があります。この要素はオプション 1 ページを表し、これには、category 要素で認識される 4 つの属性 name、 scope、label、inlineHelp が含まれています。inlineHelp 属性の値はページタイトルの下に表示されます。label 属性の値はページのタイトルとして表示されます。カテゴリ名とページ名は、構成プロファイルツリーでページの一意な場所と名前を定義します。

apt:onlineHelp 属性は、オンラインヘルプを含む HTML ファイルを Desktop Manager で利用できるようにします。この要素によって参照される HTML ページは、ユーザーが Desktop Manager のマストヘッドにある「ヘルプ」のリンクをクリックすると、コンテキスト依存のヘルプとして表示されます。apt:filePath 属性は、パスを使用してヘルプファイルの場所を指定します。区切り文字はスラッシュ ("/") です。ファイルの接尾辞 (.html) は、ISO 言語コード (ISO-639) および ISO 国コード (ISO 3166) と同様に指定してはいけません。パスのルートディレクトリは、package ディレクトリの下にある web ディレクトリです。14 ページの「ローカライズ」も参照してください。

ページには任意の数のセクションやセットを含めることができ、そのあとにオプションとして xmlHandlers のリストを付加することもできます。したがって、page 要素にはサブ要素の section、set (48 ページの「動的データ要素 : set」を参照) と xmlHandler (49 ページの「Interaction 要素: xmlHandler, event, action, choose, command」を参照) が含まれます。

section 要素は、そのプロパティーのサブ要素すべてをテーブルに似たレイアウトで視覚的にグループ化します。これには category 要素から認識される 3 つの属性 name、scope、label が含まれています。label 属性の値はセクションタイトルとして表示されます。

Basic Data 要素: property, value, constraints

```
<!ELEMENT property (constraints?, value*, visual)>
<!ATTLIST property
    apt:name ID #REQUIRED
    apt:scope (user | host | global) #IMPLIED
    apt:label NMTOKEN #IMPLIED
    apt:inlineHelp NMTOKEN #IMPLIED
    apt:dataPath CDATA #REQUIRED
    oor:type (xs:boolean | xs:short | xs:int | xs:long | xs:double |
              xs:string | xs:hexBinary |
              oor:any | oor:boolean-list | oor:short-list | oor:int-list |
              oor:long-list | oor:double-list | oor:string-list | oor:hexBinary-list)
              #IMPLIED
    apt:storeDefault (true | false) #IMPLIED
    apt:xmlHandler IDREF #IMPLIED
    apt:extendsProperty CDATA #IMPLIED
>
<!ELEMENT visual (checkBox | chooser)?>
<!ATTLIST visual
 apt:type (textField | password | textArea | radioButtons | comboBox | stringList |
           colorSelector | hidden) #IMPLIED
```

```
>

<!ELEMENT checkBox EMPTY>
<!ATTLIST checkBox
 apt:labelPost NMTOKEN #IMPLIED
>

<!ELEMENT chooser EMPTY>
<!ATTLIST chooser
 apt:labelPopup NMTOKEN #IMPLIED
 apt:listDataPath CDATA #IMPLIED
 apt:extendsChooser CDATA #IMPLIED
>
<!ELEMENT constraints (enumeration*, length?, minLength?, maxLength?, minInclusive?,
                       maxInclusive?, minExclusive?, maxExclusive?)>
<!ELEMENT enumeration EMPTY>
<!ATTLIST enumeration
    oor:value CDATA #REQUIRED
    apt:label NMTOKEN #IMPLIED
>
<!ELEMENT value (#PCDATA)>
<!ATTLIST value
    xsi:nil (true | false) #IMPLIED
    oor:separator CDATA #IMPLIED
>
```

property 要素は、チェックボックス、ラジオボタン、編集フィールドなどの GUI 要素を通して構成設定を視覚化します。これには、category 要素で認識される 4 つの属性 name、scope、label、inlineHelp が含まれています。インラインヘルプは、「値」列の入力フィールドの下に (または値文字列の下に編集不可の形で) 表示されます。label 属性の値は GUI 要素のラベルとして表示されます。カテゴリ名、ページ名、セクション名、プロパティー名は、構成プロファイルツリーでページの一意な場所と名前を定義します。

属性apt:dataPath は、プロパティーの値を保存するデータのバックエンドの場所を指すパスを定義します。dataPath 属性の値は、コンポーネントの絶対パス (たとえば org.openoffice.Office.Common/ExternalMailer/Program) です。付録 A を参照してください。property 要素の dataPath 属性は、データのバックエンドプロパティーを指す必要があります。データのバックエンドノードを指すと、実行時エラーが発生します。

apt:type は、リポジトリの構成データの種類を指定するために使用します。次のタイプが定義されます。

xs:boolean	ブール値 (true/false)
xs:short	16 ビットの整数
xs:int	32 ビットの整数

xs:long	64 ビットの整数
xs:double	浮動小数点数 (IEEE 64 ビット double の値範囲)
xs:string	プレーンテキスト (印刷可能な Unicode 文字のシーケンス)
xs:hexBinary	未解釈オクテットのシーケンス、16 進数エンコード
oor:any	上記すべてのタイプを含む
oor:*-list	上記いずれかのタイプのリスト

以上のタイプは StarSuite/OpenOffice.org Registry (OOR) 形式で定義されているタイプに似ています。再利用できるように、テンプレートでは可能な限り OOR 形式の構文を使用しています。タイプの詳細は、`http://util.openoffice.org/common/configuration/oor-document-format.html` で OpenOffice.org Registry Format (OOR) に関する文書を参照してください。

属性 `apt:storeDefault` は Desktop Manager にデフォルトデータをデータのバックエンドに保存するように指示します。デフォルトデータは `value` 要素によって定義され (以下を参照)、ユーザーにデフォルトを表示するために使用します。ユーザーが値を変更しない場合や、「コンテンツ区画」で「デフォルトの適用」アクションを実行してデフォルトデータの保存を明示的に要求した場合は、デフォルトデータがリポジトリに保存されません。`storeDefault` 属性の値を true に設定すると、ユーザーが値を変更しない場合や「デフォルトの適用」を実行した場合でも、デフォルトデータが保存されます。

property 要素には 3 つのサブ要素 `constraints`、`value`、`visual` があります。

visual 要素は、GUI のプロパティーの表示タイプを定義します。表示タイプには、`checkBox`、`radioButtons`、`comboBox`、`stringList`、`textField`、password、`textArea`、`chooser`、`colorSelector`、`hidden` があります。GUI の各要素には、編集と編集不可の 2 種類の表示形態があります。編集不可の表示形態は、Desktop Manager を使用する管理者がそのプロパティーに対する書き込み権を持っていない場合に表示されます。

`hidden` プロパティーは、視覚的な GUI 要素を描画しませんが、プロパティーに関連付けられた値をブラウザの非表示フィールドに渡します。この機能は、たとえばフロントエンドで入力された 1 つの値をバックエンドの複数の場所で保存しなければならない場合に便利です。

表示タイプは `visual` 要素の `apt:type` 属性によって定義されます。ただし、`checkBox` と `chooser` は例外です。これら 2 つの GUI 要素を正しく表示するには追加情報が必要なので、これらの要素にはその情報を含める独自のサブ要素があります。

checkbox プロパティーはチェックボックスの前後に文字列を表示します。これは `checkBox` 要素で表します。`checkBox` GUI 要素を編集不可の形態 (上の表を参照) で表示するには、さらに 2 つ文字列が必要になります。したがって、`checkBox` GUI 要素には 4 つ文字列が必要で、次のように表示されます。

1. チェックボックスの前。この文字列は`property` 要素の `label` 属性で定義される。

2. チェックボックスの後。この文字列は checkBox サブ要素の apt:labelPost 属性で定義される。この属性を定義しなければ、label 属性で定義された文字列に「.post」が付加される。この文字列はリソースファイルでキーとして検索される。
3. チェック付きのチェックボックスではなく、チェックボックスが編集不可の表示形態の場合。文字列は constraints 要素の最初の enumeration サブ要素の label 属性で定義される。制約を指定しなければ、property 要素 label 属性で定義された文字列に接尾辞「.checked」が付加される。この文字列はリソースファイルでキーとして検索される
4. チェックなしのチェックボックスではなく、チェックボックスが編集不可の表示形態の場合。文字列は constraints 要素の 2 番目の enumeration サブ要素の label 属性で定義される。制約を指定しない場合は、property 要素 label 属性で定義された文字列に接尾辞「.unchecked」が付加される。この文字列はリソースファイルでキーとして検索される

chooser プロパティーを使用すると、エントリのリストの値が確定されます。これは chooser 要素で表します。コンボボックスと違って、エントリのリストは編集可能です。このリストは chooser 要素の apt:dataPath 属性で指定したバックエンドの場所に保存されます。

「編集」ボタンをクリックすると、ポップアップウィンドウが開き、リストを編集するための GUI が提供されます。ポップアップウィンドウの内容のタイトルは、apt:chooser 要素の labelPopup 属性で定義します。constraints 要素の enumeration サブ要素を使用すると、リストのデフォルト値を指定できます (下の制約に関する説明を参照)。

apt:extendsChooser プロパティーは、別の chooser 要素を参照するために使用します。このプロパティーを使用すると、以前に定義した chooser 要素を簡単に再利用できます。その参照先 chooser で定義されている要素と属性のすべてが、参照元 chooser で定義されているかのように解釈されます。参照元 chooser で定義されているサブ要素と属性が、参照先 chooser の要素や属性を上書きします。プロパティーのパスは、参照先 chooser の検索に使用されます。プロパティーのパスは、ルートカテゴリからプロパティまでのパスの全要素の apt:name の値をスラッシュ ("/") で区切って連ねたものです。次に例を示します。/StarSuite/Internet/Proxy/Settings/MyChooser

表示タイプが指定されていない場合は、type 属性から GUI 要素が引き出されます。タイプが xs:boolean の場合は、チェックボックスが使用されます。タイプがリストの種類 (たとえば oor:short-list)、xs:hexBinary、または oor:any の場合は、テキスト領域が表示されます。その他のタイプの場合は、編集フィールドが使用されます。表示タイプもデータ型も指定されていない場合は、編集フィールドが表示され、バックエンドのデータ型が xs:string であるとします。

constraints 要素は入力フィールドに制約を加えます。たとえば、ユーザーが保存できる値を 1 から 5 までの整数に限定する場合は、oor:type 属性を「xs:int」と一緒に提供するだけでは不十分です。minInclusive 制約 1 と maxInclusive 制約 5 を指定して、必要な制約を与えます。

checkbox プロパティーと一緒に使用する場合は、列挙制約にもう 1 つの用途があります。最初の enumeration constraints サブ要素は、チェックボックスがオンになっている場合にバックエンドに保存する値を定義し、2 番目の enumeration constraints サブ要素

は、チェックボックスがオフになっている場合にバックエンドに保存する値を定義します。制約を指定しなければ、保存されるデフォルト値は true と false です。編集不可の表示形態で GUI に表示されるデフォルトの文字列は、「使用する」と「使用しない」です。

列挙制約は、`radiobutton` プロパティーと `combobox` プロパティーの場合は必須である以外は、意味が `checkbox` プロパティーと同じです。これらの要素の内容は開発者が自由に決めることができます。GUI に表示される名前は、`constraint` 要素の `label` 属性で定義します。列挙制約では `label` 属性の省略が可能です。その場合は、`property` 要素の `label` 属性で定義した文字列に接尾辞を付加して追加のリソースを指定します。

たとえば、プロパティーのラベルが「securityList」で、列挙制約に「1」、「2」、「3」の値が含まれていても列挙制約のラベルが定義されていないドロップダウンボックスがあるとします。ユーザーに表示される文字列は、リソースキーの「securityList.1」、「securityList.2」、「securityList.3」のそれぞれを検索して決定されます。

chooser プロパティーのリストエントリはローカライズされません。その結果、列挙制約の `apt:label` 属性はこれらのプロパティーには影響しません。

その他の制約については、OpenOffice.org Registry Format (OOR) の文書の Property Constraints を参照してください。

`value` 要素には、プロパティーのデフォルト値が含まれています。このデフォルト値は、デフォルト階層に含まれている値と同じにするか、ここで新たに指定することができます。これは、データバックエンドにデータが見つからない場合に、Desktop Manager が表示する値を定義します。

`value` 要素の定義は、OOR 形式で与えられる定義と似ています。 value 要素にはサブ要素がなく、`nil`、 `separator`、 `lang` という 3 つの属性があります。`xsi:nil` 属性を true に設定すると、プロパティーの値が「値なし」として定義されます。`oor:separator` 属性は、value 要素にリスト型の値が含まれている場合に、リストトークンの区切り文字として使う文字列を指定するために使用します。

ヒント – `stringList` プロパティーのリストエントリは、`oor:string-list` 型の値として保存されます。デフォルトの区切り文字は空白文字 1 つです。

`apt:xmlHandler` 属性については、49 ページの「Interaction 要素: xmlHandler, event, action, choose, command」 を参照してください。

動的データ要素 : set

```
<!ELEMENT set (page)>
<!ATTLIST set
    apt:name ID #REQUIRED
    apt:scope (user | host | global) #IMPLIED
    apt:label NMTOKEN #IMPLIED
    apt:labelPopup NMTOKEN #IMPLIED
    apt:dataPath CDATA #REQUIRED
    apt:elementNamePath CDATA #IMPLIED
>
```

これまでに紹介した要素はすべて、バックエンドの静的コンテンツを処理しました。set 要素は、動的コンテンツを処理するために使用します。これは section 要素のようにページのサブ要素で、プロパティーのセットをテーブルに表示します。

これには、category 要素で認識される 3 つの属性 name、scope、label が含まれています。label 属性の値はテーブルのタイトルとして表示されます。

apt:dataPath 属性は、要素のセットを含んだデータのバックエンドノードを指すパスを定義します。dataPath 属性の値は、org.openoffice.Office.Commands/Execute/Disabled 形式の絶対パスです (付録 A を参照)。set 要素の dataPath 属性は、バックエンドのノードを指す必要があります。バックエンドのプロパティーを指すと、エラーが発生します。

セットの子孫の dataPath 属性には動的な部分が含まれている必要があります。この動的な部分は、セットのメンバーであるバックエンドのノードの場所を指定します。バックエンドのセットのメンバーにアクセスするためには、名前が異なる必要があります。これを実現するには、変数が使用されます。

変数にはドル記号の接頭辞 *$variable_name*。有効な変数名は queriedId と silentId です。$queriedId 変数が指定された場合、Desktop Manager はユーザーに一意の ID を照会する追加の編集フィールドを表示します。$silentId 変数が指定された場合、ID はユーザーから照会されず、Desktop Manager は一意の ID を自身で生成します。

次に例を示します。property 要素の dataPath 属性の値が org.openoffice.Office.Commands/Execute/Disabled/$queriedId/Command であるとします。ユーザーが新しいセット要素を作成した場合は、セットメンバーの名前も要求されます。GUI に表示される実際の質問の文字列は apt:labelPopup 属性で指定します。この属性を省略した場合は、「新しい項目の名前を入力してください」 というプロンプトが表示されます。

パスの文字列の長さを最小限にするには、セットまたはプロパティーの dataPath 属性の値として相対パスを指定することも可能です。相対パスの例は ./$queriedId/Command です。絶対パスは、テンプレート要素ツリーを上に移動し、相対パスにその祖先の dataPaths の接頭辞を付けて構成します。たとえば、プロパティーの親がセットであるとします。このセットは dataPath の値 org.openoffice.Office.Commands/Execute/Disabled を指定します。Desktop Manager はこのパスを相対パス ./$queriedId/Command と組み合わせ、絶対パス org.openoffice.Office.Commands/Execute/Disabled/$queriedId/Command にします。

dataPath 属性 (set 要素と property 要素) をサポートしているセットのすべての子孫がその dataPath 属性の値として絶対パスを指定している場合は、セットの dataPath 属性を指定する必要はありません。

再帰的なセットの構造 (セットのセット) も処理できます。これは、セット要素に page 要素を含め、それにまた set 要素を含めることで実現します。サブセットの dataPath 属性では、スーパーセットに相対するパスを指定できます。

セットのメンバーがバックエンドプロパティーの場合、バックエンドプロパティーの oor: name 属性は、GUI でラベルとして表示されます。バックエンドプロパティーの値は GUI 要素のラベルとして表示されます。

セットのメンバーがバックエンドのノードの場合は、バックエンドノードの oor:name 属性は、リンクの名前として表示されます。apt:elementNamePath 属性を使用すると、この命名スキームを上書きできます。elementNamePath は、バックエンドのノードに相対するパスを指定します。このパスは、バックエンドのプロパティーを指している必要があり、その値はリンクの名前として表示されます。このようなリンクをユーザーがクリックすると、「コンテンツ区画」が更新されて、セットの page サブ要素で指定されたページが表示されます。

Interaction 要素: xmlHandler, event, action, choose, command

```
<!ELEMENT xmlHandler (event+, action+)>
<!ATTLIST xmlHandler apt:name ID #REQUIRED>

<!ELEMENT event EMPTY>
<!ATTLIST event apt:type (onChange) #IMPLIED>

<!ELEMENT action (choose|command)+>

<!ELEMENT choose (when+, otherwise?)>

<!ELEMENT when (command+)>
<!ATTLIST when apt:test CDATA #REQUIRED>

<!ELEMENT otherwise (command+)>

<!ELEMENT command (#PCDATA)>
```

xmlHandler 要素は、クライアント側で JavaScript コードを実行する場合に使用します。コードの実行は、環境の変化によってトリガーされます。このような変化はイベントで伝播されます。イベントは、状態が変わった場合にプロパティーによって発行されます。イベントタイプは変化の種類を示します。

XML ハンドラを定義するには、apt:template 要素の xmlHandler サブ要素を指定します。これは、apt:name 属性で定義された名前が必要です。XML ハンドラは、property 要素の

apt:xmlHandler 属性を指定 (XML ハンドラの名前をその値として使用) して、プロパティーが発行したイベントを待機するように登録されます。

event 要素は、xmlHandler が待機するイベントを、その apt:type 属性を使用して指定する場合に使用します。ここでは、onChange 要素のみを定義します。このイベントは、ユーザーがその値を変更した場合にプロパティーによって発行されます。プロパティーの値は、たとえば、ユーザーが編集フィールドでキーを入力したり、チェックボックスをオフにしたり、リストボックスでエントリを選択してこの入力を変更した瞬間に変化します。GUI 要素にフォーカスを移したり、GUI 要素からフォーカスを外しても、このイベントはトリガーされません。

1 つまたは複数のプロパティーでイベントのハンドラが登録され、これらのプロパティーのいずれかでそのイベントが発生した場合に、action 要素で定義されているコードが実行されます。action 要素には、 choose 要素か command 要素が少なくとも 1 つ含まれています。

command 要素は、クライアント側で実行される命令を指定します。現時点で含めることができるのは代入式のみです。代入式の左辺および右辺でも計算は許可されていません。代入式のスキーマは次のとおりです。<variable>=<value>.

ドット表記 <property>.<qualifier>. <property> はプロパティーの名前である必要があります。<qualifier> は次の 2 つの値を持つことができます。「value」および「enabled」。value 修飾子はそのプロパティーの値を提供します。変数の値は、プロパティーで指定されているタイプと互換性がある限り、読み取ってどの値にでも確定できます。enabled 修飾子には、プロパティーが有効になっている (フォーカスを受け取れる) 場合は「true」が含まれ、そうでない場合は「false」が含まれます。これを読み取って「true」または「false」に設定できます。例: **propname.enabled=false**.

choose 要素は XSLT で定義される choose 要素に似ており、コマンドを条件付きで実行できます。when 要素を少なくとも 1 つ含んでいる必要があり、末尾に otherwise 要素を 1 つ含むことができます。

when 要素には、 1 つまたは複数の command サブ要素と 1 つの apt:test 属性があります。test 属性では、ブール値に評価される式を指定する必要があります。

式は、変数、数字、文字列、および次のトークンで構成できます。

=	等しい
!=	等しくない
<	より小さい
>	より大きい
<=	より小さい、または等しい
>=	より大きい、または等しい

( )	挿入要素
not()	ブール演算子 NOT
and	ブール演算子 AND
or	ブール演算子 OR
true	ブール肯定
false	ブール否定

例: **`(propname.enabled!=false) and not(propname.value='foo')`**.

式が「true」に評価される場合は、`when` 要素のコマンドが実行され、choose ステートメントが実行されます。「false」に評価される場合は、次の `when` 要素が評価されます。どの `when` 要素も true ではなく、`otherwise` 要素が指定されている場合は、`otherwise` 要素のコマンドが実行されます。

◆ ◆ ◆ 付 録 C

テンプレートの DTD

```
<?xml version="1.0" encoding="UTF-8"?>

<!ELEMENT apt:template (resImport*, category)>
<!ATTLIST apt:template
    xmlns:apt CDATA #FIXED "http://www.sun.com/jds/apoc/2004/template"
    xmlns:oor CDATA #FIXED "http://openoffice.org/2001/registry"
    xmlns:xs  CDATA #FIXED "http://www.w3.org/2001/XMLSchema"
    xmlns:xsi CDATA #FIXED "http://www.w3.org/2001/XMLSchema-instance"
>
<!ELEMENT resImport EMPTY>
<!ATTLIST resImport
    apt:packagePath NMTOKEN #REQUIRED
>

<!ELEMENT category (category | page)>
<!ATTLIST category
    apt:name ID #REQUIRED
    apt:scope (user | host | global) #IMPLIED
    apt:label NMTOKEN #IMPLIED
    apt:inlineHelp NMTOKEN #IMPLIED
>

<!ELEMENT page ((section | set)+, xmlHandler*)>
<!ATTLIST page
    apt:name ID #REQUIRED
    apt:scope (user | host | global) #IMPLIED
    apt:label NMTOKEN #IMPLIED
    apt:inlineHelp NMTOKEN #IMPLIED
    apt:onlineHelp CDATA #IMPLIED
>

<!ELEMENT section (property+)>
<!ATTLIST section
    apt:name ID #REQUIRED
```

```
    apt:scope (user | host | global) #IMPLIED
    apt:label NMTOKEN #IMPLIED
>

<!ELEMENT set (page)>
<!ATTLIST set
    apt:name ID #REQUIRED
    apt:scope (user | host | global) #IMPLIED
    apt:label NMTOKEN #IMPLIED
    apt:labelPopup NMTOKEN #IMPLIED
    apt:dataPath CDATA #REQUIRED
    apt:elementNamePath CDATA #IMPLIED
>

<!ELEMENT property (constraints?, value*, visual)>
<!ATTLIST property
    apt:name ID #REQUIRED
    apt:scope (user | host | global) #IMPLIED
    apt:label NMTOKEN #IMPLIED
    apt:inlineHelp NMTOKEN #IMPLIED
    apt:dataPath CDATA #REQUIRED
    oor:type (xs:boolean | xs:short | xs:int | xs:long | xs:double |
              xs:string | xs:hexBinary | oor:any | oor:boolean-list |
              oor:short-list | oor:int-list | oor:long-list | oor:double-list |
              oor:string-list | oor:hexBinary-list) #IMPLIED
    apt:storeDefault (true | false) #IMPLIED
    apt:xmlHandler IDREF #IMPLIED
    apt:extendsProperty CDATA #IMPLIED
>

<!ELEMENT visual (checkBox | chooser)?>
<!ATTLIST visual
    apt:type (textField | password | textArea | radioButtons | comboBox |
              stringList | colorSelector | hidden) #IMPLIED
>

<!ELEMENT  checkBox EMPTY>
<!ATTLIST checkBox
    apt:labelPost NMTOKEN #IMPLIED
>

<!ELEMENT  chooser EMPTY>
<!ATTLIST chooser
    apt:labelPopup NMTOKEN #IMPLIED
    apt:listDataPath CDATA #IMPLIED
>

<!ELEMENT value (#PCDATA)>
```

```
<!ATTLIST value
    xsi:nil (true | false) #IMPLIED
    oor:separator CDATA #IMPLIED
>

<!ELEMENT constraints (enumeration*, length?, minLength?, maxLength?, minInclusive?,
                       maxInclusive?, minExclusive?, maxExclusive?)>

<!ELEMENT enumeration EMPTY>
<!ATTLIST enumeration
    oor:value CDATA #REQUIRED
    apt:label NMTOKEN #IMPLIED
>

<!ELEMENT length EMPTY>
<!ATTLIST length oor:value CDATA #REQUIRED
>
<!ELEMENT minLength EMPTY>
<!ATTLIST minLength oor:value CDATA #REQUIRED
>
<!ELEMENT maxLength EMPTY>
<!ATTLIST maxLength oor:value CDATA #REQUIRED
>
<!ELEMENT minInclusive EMPTY>
<!ATTLIST minInclusive oor:value CDATA #REQUIRED
>
<!ELEMENT maxInclusive EMPTY>
<!ATTLIST maxInclusive oor:value CDATA #REQUIRED
>
<!ELEMENT minExclusive EMPTY>
<!ATTLIST minExclusive oor:value CDATA #REQUIRED
>
<!ELEMENT maxExclusive EMPTY>
<!ATTLIST maxExclusive oor:value CDATA #REQUIRED
>
<!ELEMENT xmlHandler (event+, action+)>
<!ATTLIST xmlHandler apt:name ID #REQUIRED>

<!ELEMENT event EMPTY>
<!ATTLIST event apt:type (onChange) #IMPLIED>

<!ELEMENT action (choose|command)+>

<!ELEMENT choose (when+, otherwise?)>

<!ELEMENT when (command+)>
<!ATTLIST when apt:test CDATA #REQUIRED>
```

```
<!ELEMENT otherwise (command+)>
<!ELEMENT command (#PCDATA)>
```