Sun™ Identity Manager 8.0
配備に関する技術概要

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

Part No: 820-5432

目 次

はじめに

本書『Sun Java™ System Identity Manager の配備に関する技術概要』では、Sun Java™ System Identity Manager を環境に合わせてカスタマイズするときに使用する参照情報と手順の概要について説明します。

対象読者

Sun Java™ System 『Identity Manager の配備に関する技術概要』は、製品配備のさまざまな段階で Identity Manager を顧客インストール用にカスタマイズするのに必要なワークフロー、ビュー、規則、システム設定、およびその他の設定ファイルを作成および更新するデプロイヤおよび管理者に向けて作成されました。

デプロイヤは、プログラミングに関する予備知識があり、XML、Java、Emacs や IDE (Eclipse または NetBeans など) に精通していることが望まれます。

内容の紹介

『Identity Manager の配備に関する技術概要』は、次の章で構成されています。

- 第 1 章「属性の処理」- アイデンティティー属性と、この機能を使って Identity Manager 配備中のデータフローを整理する方法について説明します。
- 第 2 章「データ読み込みと同期」- アカウント情報を Identity Manager に読み込むための調整とその他の機構の概要を説明します。調整では、Identity Manager で定義されたユーザーのセットと Identity Manager リソースで定義されたアカウントのセットを比較します。
- 第 3 章「データ読み込みのシナリオ」- アカウント情報を Identity Manager に読み込むときに考慮すべきヒントについて説明します。これには、発生する可能性がある問題を示すサンプルシナリオが含まれます。
- 第 4 章「データエクスポータ」- データエクスポータの計画と実装の方法について説明します。
- 第 5 章「ユーザーアクションを設定する」- Identity Manager の管理者インタフェースにカスタムタスクを追加する方法と、インタフェースの 2 つのエリアから実行できるユーザーアクションを設定する方法について説明します。
- 第 6 章「Identity Manager のプライベートラベリング」- 組織のスタイル標準に合わせて IDM の色、ロゴ、およびヘッダーとフッターの内容をカスタマイズする方法について説明します。
- 付録 A「設定オブジェクトを編集する」- 設定オブジェクトの概要と `UserUIConfig` オブジェクトについて説明します。
- 付録 B「国際化サポートを有効にする」- Identity Manager で複数の言語を使用する場合や英語以外の言語を表示する場合の設定に関する情報を提供します。

表記上の規則

この項の表は、本書で使用する次の表記規則について説明しています。

- 表記上の規則
- 記号
- シェルプロンプト

表記上の規則

次の表は、本書で使用する表記上の規則について説明しています。

表 1　表記上の規則

字体または記号	意味	例
`AaBbCc123` (モノスペース)	API および言語の要素、HTML タグ、Web サイトの URL、コマンド名、ファイル名、ディレクトリパス名、画面出力の表示、サンプルコードを示します。	`.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。 `% You have mail.`
`AaBbCc123` (太字のモノスペース)	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は区別して示します。	`% su` `Password:`
AaBbCc123 (イタリック)	実際の名前または値によって置き換えられるコマンドまたはパス名の可変部分。	これらを、*class* オプションと呼びます。 このファイルは、*install-dir*`/bin` ディレクトリにあります。

記号

次の表は、本書で使用する記号の表記規則を示しています。

表 2　記号の表記規則

記号	説明	例	意味
`[ ]`	省略可能なコマンドオプションが入ります。	`ls [-l]`	`-l` オプションは省略可能です。
-	同時に押すキーを連結します。	Control-A	Ctrl キーと A キーを同時に押します。
+	連続して押すキーを連結します。	Ctrl+A+N	Ctrl キーを押し、離してから、以後のキーを続けて押します。
>	グラフィカルユーザーインタフェースで選択するメニュー項目を示します。	「ファイル」>「新規」>「テンプレート」	「ファイル」メニューから「新規」を選択します。「新規」サブメニューから、「テンプレート」を選択します。

シェルプロンプト

次の表は、本書で使用するシェルプロンプトを示しています。

表 3　　シェルプロンプト

シェル	プロンプト
UNIX または Linux の C シェル	*machine-name*%
UNIX または Linux の C シェルのスーパーユーザー	*machine-name*#
UNIX または Linux の Bourne シェルおよび Korn シェル	$
UNIX または Linux の Bourne シェルおよび Korn シェルのスーパーユーザー	#
Windows のコマンド行	C:¥

関連ドキュメントとヘルプ

Sun Microsystems は、Identity Manager をインストール、使用、および設定する際に役立つ次のような追加のドキュメントと情報を提供しています。

- 『Identity Manager インストール』: Identity Manager と関連ソフトウェアをインストールおよび設定する手順と参照情報が記載されています。
- 『Identity Manager Upgrade』: Identity Manager と関連ソフトウェアをアップグレードおよび設定する手順と参照情報が記載されています。
- 『Identity Manager 管理ガイド』: Identity Manager を使用して企業情報システムへのセキュリティー保護されたユーザーアクセスを実現するために、手順、チュートリアル、実例を説明します。
- 『Identity Manager 配備ツール』: Identity Manager のさまざまな配備ツールの使用方法を示す参照情報と手順が記載されています。これらの情報は、Identity Manager サーバーによって提供される規則と規則ライブラリ、共通のタスクとプロセス、辞書サポート、および SOAP ベースの Web サービスインタフェースを対象としています。
- 『Identity Manager ワークフロー、フォーム、およびビュー』: Identity Manager のワークフロー、フォーム、およびビューの使用方法を示す参照情報と手順が記載されています。この中には、これらのオブジェクトをカスタマイズするのに必要なツールに関する情報が含まれます。
- 『Identity Manager リソースリファレンス』: アカウント情報をリソースから Sun Java™ System Identity Manager に読み込んで同期する方法を示す参照情報と手順が記載されています。

- 『Identity Manager Tuning, Troubleshooting, and Error Messages』: Sun Java™ System Identity Manager のチューニングに関するガイダンス、問題の追跡とトラブルシューティングの手順、およびこの製品を処理したときに発生する可能性があるエラーメッセージと例外についての説明を提供する参照情報と手順が記載されています。
- 『Identity Manager Service Provider Edition Deployment』: Sun Java™ System Identity Manager Service Provider Edition の計画と実装の方法を示す参照情報と手順が記載されています。
- Identity Manager ヘルプ : Identity Manager の完全な手順、参照情報、用語の説明を記載したオンラインガイダンス、オンライン情報です。ヘルプにアクセスするには、Identity Manager メニューバーの「ヘルプ」リンクをクリックします。主要なフィールドには、ガイダンス (フィールド固有の情報) があります。

オンライン上の Sun リソースへのアクセス

製品のダウンロード、プロフェショナルサービス、パッチとサポート、および開発者向け追加情報については、次の Web サイトにアクセスしてください。

- ダウンロードセンター
  http://wwws.sun.com/software/download/
- プロフェショナルサービス
  http://www.sun.com/service/sunps/sunone/index.html
- Sun Enterprise サービス、Solaris パッチ、およびサポート
  http://sunsolve.sun.com/
- 開発者向け情報
  http://developers.sun.com/prodtech/index.html

Sun テクニカルサポートへのお問い合わせ

製品のドキュメントで解決できない、本製品に関する技術的な質問については、次のいずれかの方法でカスタマサポートにお問い合わせください。

- オンラインサポート Web サイト http://www.sun.com/service/online/us
- 保守契約に基づいて提供されるサポート電話番号

関連するサードパーティー Web サイト

Sun は、本書に記載されているサードパーティー Web サイトの利用について責任を負いません。Sun は、このようなサイトまたはリソースで得られるあらゆる内容、広告、製品、およびその他素材を保証するものではなく、責任または義務を負いません。Sun は、このようなサイトまたはリソースで得られるあらゆるコンテンツ、製品、またはサービスによって生じる、または生じたと主張される、または使用に関連して生じる、または信頼することによって生じる、いかなる損害または損失についても責任または義務を負いません。

ご意見、ご要望の送付先

Sun ではマニュアルの品質向上のため、お客様のご意見、ご要望をお受けしております。

コメントをお送りになる場合は、http://docs.sun.com にアクセスして「コメントの送信」をクリックしてください。オンラインフォームで、ドキュメントのタイトルと Part No. を入力します。Part No. は、マニュアルのタイトルページまたは最上部に記載されている 7 桁または 9 桁の番号です。

たとえば、本書のタイトルは『Sun Java™ System Identity Manager の配備に関する技術概要』であり、Part No. は 820-5432 です。

第 1 章

属性の処理

この章では、Identity Manager 配備で使用される属性の概念的な概要について説明します。この章で説明するトピックは次のとおりです。

関連する章

属性は、Identity Manager のさまざまな処理の中で処理され、このマニュアルセット全体で説明されています。次の章には、属性に関する多くの情報が含まれています。

- ビュー属性、属性の登録、および deferred 属性に関する説明の詳細は、『Identity Manager ワークフロー、フォーム、およびビュー』の「Identity Manager のビュー」の章にあります。
- リソース属性に関する説明は、『Identity Manager リソースリファレンス』にあります。

属性とは

属性は、Identity Manager オブジェクトや外部リソースの特性を定義および処理するために使われる名前と値のペアです。フォーム、ワークフロー、規則などの Identity Manager コンポーネントは、通常の処理でデータアクセスやデータ変換の主要部分として属性を呼び出します。

属性のタイプ

Identity Manager 配備のオブジェクトにはさまざまな属性を入れることができますが、通常は次の節で説明するタイプの属性を使用します。

- 概要の属性
- クエリー可能な属性
- インライン属性
- 拡張属性
- オペレーショナル属性
- ビュー属性
- リソースユーザー属性
- アイデンティティーシステムユーザー属性
- その他の標準的な属性

概要の属性

すべての持続オブジェクトは、概要の属性のセットを公開します。概要の属性は Identity Manager Schema Configuration オブジェクトで設定され、リスト処理の結果として各項目に返される値が含まれます。各 `PersistentObject` サブクラスは、`getSummaryAttributes` メソッドを上書きすることにより、デフォルトの属性セットを拡張できます。

概要の属性は、直列化されて文字列になったときに概要の属性全体の長さが制限されるため、通常は単一の値です。

ユーザーオブジェクト用のこれらの属性は、Identity Manager Schema Configuration オブジェクトを使って直接設定できます。詳細については、143 ページの「IDM Schema Configuration オブジェクト」を参照してください。

注 現在、Identity Manager では `UserUIConfig` の `SummaryAttrNames` セクションは使用されていません。

クエリー可能な属性

すべての持続オブジェクトは、クエリー可能な属性のセットを公開します。クエリー可能な属性には、フィルタや照合に使用する値のセットが含まれ、これらの属性は Identity Manager Schema Configuration オブジェクトで設定されます。クエリー可能な属性は複数の値をとることができます。

注 `UserUIConfig` の `QueryableAttrNames` セクションは、Identity Manager では使用されません。

インライン属性

オブジェクトの型ごとに、クエリー可能な属性を最大 5 個までインライン属性として指定できます。インライン属性は、Identity ManagerRepository Configuration オブジェクトで設定されます。

注 現在、`UserUIConfig` ではインライン属性を設定しません。

属性をインラインとして指定すると、その属性に対するクエリーのパフォーマンスを最適化するようデータストアに要求します。

Identity Manager は、通常、メインオブジェクトテーブルとは別個の属性テーブル内の行としてクエリー可能な属性の各値を格納します。属性テーブルをオブジェクトテーブルに結合することにより、`AttributeCondition` と一致するオブジェクトを選択できます。

ただし、Identity Manager はインライン属性の値をその型のオブジェクトテーブルに直接格納します。属性をインラインとして指定すると、Identity Manager はより効率的な SQL を生成できます。メインオブジェクトテーブル上の列の式は、対応する属性テーブルへの JOIN (または対応する属性テーブルに対する EXISTS 述語) よりも高速になります。これにより、属性に対するすべてのクエリーのパフォーマンスが向上します。

インライン属性の特徴は次のとおりです。

- **インライン属性は、単一値である必要があります。**これは、インライン属性の値がオブジェクトテーブルの親行の 1 つの列に保存されるためです。
- **インラインにできるクエリー可能な属性は、1 つの型につき最大 5 個です。**これは、任意の属性値の格納に使用できる列がオブジェクトテーブルに 5 つしかないためです。

- **同じクエリー可能な属性のセットが、1 つの型の全インスタンスでインラインとして指定されます。**これは、列の値と属性の名前の対応がこの設定のみで指定されるためです。つまり、ある型に対するインライン属性の設定は、どの属性がどの列に格納されているかをリポジトリが識別するための唯一の方法です。

拡張属性

拡張属性は、組み込み以外の属性 (ユーザーの employeeNumber など) です。employeeNumber を使用してクエリーを実行したいという顧客がほとんどであるため、設定を介してこの属性をクエリー可能な拡張属性として追加することができます。

注 将来リリースされる Identity Manager の新しいコア属性との競合を避けるために、拡張属性に配備固有の接頭辞を付けることをお勧めします。

たとえば、employeeNumber を記録するために拡張属性をユーザーに追加するときに、acme_employeeNumber などの会社に関連付けられた接頭辞を使用します。将来の Identity Manager リリースに employeeNumber という組み込みユーザー属性が組み込まれても、2 つの属性は区別されます。それ以外の場合は、組み込み属性が優先されます。

拡張属性は組み込みではないため、これらの属性は IDM Schema Configuration オブジェクトの <IDMAttributeConfigurations> セクションに存在する必要があります。このセクションでは属性名、構文 (string、int、date など)、および属性が単一値と複数値のどちらであるかが取得されます。名前を付けられた属性 (MemberObjectGroups など) は複数のオブジェクトクラス中に存在することがあるため、IDMObjectClassConfiguration では、どの属性がどのクラスに存在するのかが取得されます。

注 IDM Schema Configuration オブジェクトは、IDMSchemaConfig authType で保護されます。

ユーザーまたはロールの Identity Manager スキーマを参照または編集する必要のある管理者は、IDMSchemaConfig AdminGroup (機能) を割り当てる必要があります。コンフィギュレータのユーザーには、デフォルトでこの AdminGroup が割り当てられています。

User 拡張属性の詳細については、『Identity Manager ワークフロー、フォーム、およびビュー』の「Identity Manager のビュー」章で、ユーザービューの accounts[lighthouse] 属性に関する説明を参照してください。

組み込み属性および拡張属性は、クエリー可能または概要として公開できます。一部の組み込み属性には REFERENCE 構文がありますが、拡張属性は REFERENCE にすることが許可されていません。

効果的なスキーマの <Comments> セクションには、関連するオブジェクトクラスの拡張属性のほかに、使用可能な内部属性に関する情報が含まれています。Identity Manager デバッグページで「Display Schema」ボタンをクリックし、リストから ObjectClass スキーマを選択すると、この情報を参照できます。

注

拡張属性はユーザー、ロール、およびロールの拡張でのみサポートされます。

ユーザーおよびロールの一部の組み込み属性参照は、デフォルトでクエリー可能または概要ではありませんが、次の属性を公開できます。

- ユーザーの場合 :
  - MemberAdminGroups
  - adminRoles
  - adminGroupsRule
- ロールおよびロールの拡張の場合 : role_applications

属性定義の場合、IDMObjectClass スキーマを参照するには、デバッグページで「Display Schema」ボタンをクリックします。管理者が IDMObjectClass スキーマを参照するには、IDMSchemaConfig に対する参照権限が必要です。

オペレーショナル属性

Identity Manager では、リポジトリが正常に機能するために必要ないくつかの属性が事前に定義されています。ID、type、および name は特に重要です。

リポジトリに格納されているすべての PersistentObject には、グローバルに一意な内部識別子 (ID) があります。ID 値は時間と空間を越えて一意であり、生成された ID 値は再使用されません。事前に定義された Identity Manager オブジェクトの中には、プログラム定数として定義された既知の識別子を持つものがあります。これらは、擬似 ID と呼ばれます。リポジトリでは、オブジェクトの ID が変更されないことが保証されます。

同じ型のオブジェクトは、通常、同じ Java クラスにマップされます。つまり、直列化復元されるときに同じ Java クラスのインスタンスとして構築されます。型と Java クラスが 1 対 1 で対応しない場合は、少なくとも同じ型のすべてのオブジェクトが同じ機構を使って、対応する Java クラスを検索します。たとえば、一部の型のオブジェクトが完全修飾クラス名を含む class 属性を公開するなどです。

オブジェクトの名前は、型の中で一意である必要があります。つまり、ある型の 1 つのオブジェクトは特定の名前しか持つことができません。ただし、型の異なる別のオブジェクトが同じ名前を持つことは可能です。これにより、型ごとに従属する名前空間が効果的に定義されます。オブジェクトの名前は変更できますが、オブジェクトの ID は変更できません。

ビュー属性

ビューとは、リポジトリに格納された 1 つ以上のオブジェクトを組み合わせて構成した名前と値のペア、またはリソースから読み取った名前と値のペアの集合です。ビュー属性では、文字列などの個々の値、リストなどの集合、または別のオブジェクトの参照を値として使用できます。

Identity Manager の管理者インタフェースまたはユーザーインタフェースからユーザーアカウントの作成または変更を実行する場合は、ユーザービューを間接的に処理します。ワークフロープロセスも、ユーザービューと連携しています。要求をワークフロープロセスに渡すと、属性がビューとしてプロセスに送信されます。ワークフロープロセスの中で手動プロセスを要求すると、ユーザービューの属性を表示して詳細に変更できます。

ビューの処理については、『Identity Manager ワークフロー、フォーム、およびビュー』の「Identity Manager のビュー」の章で詳細に説明しています。

リソースユーザー属性

リソースユーザー属性 は、Identity Manager のアカウント属性をスキーママップ (右側) のリソースアカウント属性にマップします。属性のリストはリソースごとに異なります。使用していない属性は、スキーママップページから削除できます。ただし、属性の追加には、アダプタのコードを編集する必要がある場合があります。

リソースユーザー属性は、アダプタがリソースと通信しているときにだけ使用されます。

リソースユーザー属性の処理については、『Identity Manager リソースリファレンス』で詳細に説明しています。

アイデンティティーシステムユーザー属性

アイデンティティーシステムユーザー属性は、リソースユーザー属性に対応する Identity Manager 内部の値を定義します。アイデンティティーシステムユーザー属性は、規則、フォーム、およびその他の Identity Manager 固有の機能で使用できます。Identity Manager では、スキーママップの左側にこれらの属性が表示されます。

アイデンティティーシステムユーザー属性の処理については、『Identity Manager リソースリファレンス』で詳細に説明しています。

その他の標準的な属性

その他の標準属性の一部を使用すると、オブジェクト (`MemberObjectGroups`、`subType`、`authType` など) へのアクセスを制限したり、履歴情報 (作成者、作成日など) を表示したりできます。

MemberObjectGroups

すべての持続オブジェクトは、少なくとも 1 つのオブジェクトグループに属しています。このような複数値属性の各値は、ObjectGroup オブジェクトの ID です。

ObjectGroup は、Identity Manager の管理者インタフェースとユーザーインタフェースでは組織として公開されます。ObjectGroup メンバーシップは、セッションレベルでの認証 (つまり、管理者およびユーザーのリポジトリオブジェクトへのアクセス) を制御しますが、リポジトリ自体はオブジェクトグループのメンバーシップを無視します。

creator、*createDate*、*lastModifier*、および *lastModDate*

これらの値は、各オブジェクトの履歴情報を記録します。これらの属性はリポジトリが管理しますが、リポジトリが使用することはありません。

PropertyList

すべての持続オブジェクトは、任意のプロパティーのリストを格納できます。この機能は、一般にあまり使用されません。

subType

すべての持続オブジェクトは subType 属性を持つことができます。たとえば、Identity Manager は Attribute.SUBTYPE を使って利用可能な相関規則と確認規則の個々のリストを選択します。

authType

authType 属性を用いると、組織 (オブジェクトグループ) を管理しないユーザーに、限られた範囲内での、限定された認証を付与することができます。これらの主体は、Identity Manager の標準の認証付与スキームにおいては、これ以外の方法では権限を得ることができません。

属性条件を使用する

属性条件は、属性の値を評価する式です。属性条件は、特定の条件に一致するオブジェクトのサブセットを選択するためによく使われます。

個々の属性条件は、1 つの条件を表し、次の要素で構成されます。

- (クエリー可能な属性の) 属性名
- 演算子 (実行するチェックや比較の種類)
- オペランド (指定された値のセット)

属性条件の演算子

AttributeCondition には、次のような演算子が定義されています。

表 1-1 属性条件の演算子

演算子	説明
EQ、EQUALS	指定された属性に関して、(大文字小文字は無視して) 字句的にオペランドと等しい値がオブジェクトに少なくとも 1 つあります。
NE、NOT_EQUALS	指定された属性に関して、(大文字小文字は無視して) 字句的にオペランドと等しい値がオブジェクトにありません。
GT、GREATER_THAN	指定された属性に関して、(大文字小文字は無視して) 字句的にオペランドより大きい値がオブジェクトに少なくとも 1 つあります。

表 1-1 属性条件の演算子 (続き)

演算子	説明
GE	指定された属性に関して、(大文字小文字は無視して) 字句的にオペランドと等しいか、それより大きい値がオブジェクトに少なくとも 1 つあります。
GT、GREATER_THAN	指定された属性に関して、(大文字小文字は無視して) 字句的にオペランドより大きい値がオブジェクトに少なくとも 1 つあります。
GE	指定された属性に関して、(大文字小文字は無視して) 字句的にオペランドと等しいか、それより大きい値がオブジェクトに少なくとも 1 つあります。
LE	指定された属性に関して、(大文字小文字は無視して) 字句的にオペランドと等しいか、それより小さい値がオブジェクトに少なくとも 1 つあります。
LT、LESS_THAN	指定された属性に関して、(大文字小文字は無視して) 字句的にオペランドより小さい値がオブジェクトに少なくとも 1 つあります。
STARTS_WITH	指定された属性に関して、(大文字小文字は無視して) オペランドの部分文字列の先頭と一致する値がオブジェクトに少なくとも 1 つあります。
ENDS_WITH	指定された属性に関して、(大文字小文字は無視して) オペランドの部分文字列の末尾と一致する値がオブジェクトに少なくとも 1 つあります。
CONTAINS	指定された属性に関して、(大文字小文字は無視して) オペランドの部分文字列と一致する値がオブジェクトに少なくとも 1 つあります。
IS_PRESENT	指定された属性の値がオブジェクトに少なくとも 1 つあります。この演算子はオペランドを取りません。
NOT_PRESENT	指定された属性の値がオブジェクトにありません。この演算子はオペランドを取りません。
IN、IS_ONE_OF	指定された属性に関して、(大文字小文字は無視して) 字句的に (リスト) オペランド内のいずれかの値と等しい値がオブジェクトに少なくとも 1 つあります。

注 RelationalDataStore は、各属性条件を演算の WHERE 句の一部となる適切な述語に変換することにより、評価を最適化します。しかしながら、複数値の属性の処理に特別なロジックは必要ありません。RelationalDataStore は、これを処理する適切な SQL DML を自動的に生成します。

属性条件は、属性の値ごとに適用されます。特に演算子 NE は、指定された属性について、指定されたオペランドと等しい値がオブジェクトにない場合かつその場合にかぎり true になります。演算子 EQ は、指定された属性の、指定されたオペランドと一致する値がオブジェクトに少なくとも 1 つある場合に true になります。

暗黙的な AND 結合

属性条件のセットは、暗黙的に AND 結合されます。つまり、属性条件のセットは、セットに含まれるすべての属性条件が true と評価された場合かつその場合にかぎり true と評価されます。逆に、セット内のいずれかの属性条件が false と評価されると、ただちに属性条件のセットが false と評価されます。

Identity Manager の属性条件には、一般的に使用される演算子が公開されています。通常は、Identity Manager の属性条件を使って一連の選択条件を表現できます。表現できない条件もごく一部ありますが、多くの場合、クエリー可能な属性を追加するか、クエリー可能な属性の表現を変更することによって、そのような条件にも適切に対応できます。

サンプルシナリオ : 組織にユーザーメンバー規則を設定する

次の属性を使用すると、特定組織のユーザーのセットを識別できます。

- **(Identity Manager から見て) 外部のリソースアカウント属性**。この場合、複数の Identity Manager ユーザーが異なるリソース上で同じ acctid を持っている可能性があるため、一致する Identity Manager ユーザーを見つけるためには、リソースアカウント ID とリソース名の両方 (たとえば、acctid:resname) が必要です。
- **Identity Manager のユーザーアカウント属性** (たとえば、名前、場所、マネージャー)。

「OR 結合」の結果が正しく得られるように、複数の属性条件を使用しないでください。代わりに、次のようにオペランドのリストとともに「is one of」演算子を使用します。

```
<list>
  <new class='com.waveset.object.AttributeCondition'>
    <s>firstname</s>
    <s>is one of</s>
    <list>
      <s>Nicola</s>
      <s>Paolo</s>
    </list>
  </new>
</list>
```

サンプルシナリオ : 管理ロールを持たないすべてのユーザーを含める

指定された管理者ロールを持つユーザーを除くすべてのユーザーを含める規則が必要です。

属性条件は暗黙的に AND 結合されるため、次の 2 つの属性条件を使用できます。

- 少なくとも 1 つの管理者ロールを持つユーザーを選択する (つまり、管理者ではないユーザーを除外する) 条件。この条件に一致するユーザーは、adminRoles 属性の値を少なくとも 1 つ持つことになります。

```
<AttributeCondition>
   <s>adminRoles</s>
   <s>exists</s>
 </AttributeCondition>
```

- 特定の管理者ロールの任意のセットを持つユーザーを除外する条件。次に示す条件の例では、adminRoles 属性の値が ar1 か ar2 であるユーザーが除外されます。

```
<AttributeCondition>
   <s>adminRoles</s>
   <s>is not</s>
   <list>
     <s>ar1</s>
     <s>ar2</s>
   </list>
 </AttributeCondition>
```

この 2 つの条件を組み合わせることにより、指定されたリストに含まれない管理者ロールを持ったユーザーを返す規則を作成できます。

シークレット属性を使用する

Identity Manager では、編集フォームにアスタリスク付きで表示されるように属性を設定した場合でも、結果ページの属性の値は平文形式で表示されます。属性値がキャッシュに表示されないようにする場合は、属性をシークレットとして登録できます。シークレット属性の値は、ブラウザのキャッシュに平文形式で表示されませんが、これらの属性は、ほかの属性と同じように Identity Manager で処理されます。

たとえば、社会保障番号は、一般的にシークレット属性として登録される属性です。

結果テーブルを描画するときに、Identity Manager は、シークレットとして登録されている属性があるかどうかを確認し、シークレット属性の値をアスタリスクのみで表示します。

シークレット属性を登録するには、次のようにしてシークレット属性を System Configuration オブジェクトに追加します。

```
<Attribute name='secretAttributes'>
   <List>
     <String>email</String>
     <String>myAttribute</String>
   </List>
</Attribute>
```

第 2 章

認可タイプでの処理

この章では、Identity Manager 配備で使用される認可タイプ (AuthTypes) の概念的な概要について説明します。この章で説明するトピックは次のとおりです。

認可タイプとは

Identity Manager では、コード変更を必要とせずにオブジェクトに承認を行う権利を割り当てるメカニズムとして、認可タイプが提供されています。この拡張性のあるメカニズムはリポジトリ記憶領域タイプから独立しており、特に TaskDefinition および Configuration オブジェクトで役立ちます。これらのオブジェクトは、同じリポジトリタイプを共有していますが、それぞれ別個の認証を必要とするさまざまな機能を実行します。たとえば、「ユーザーメンバー規則」ドロップダウンリストに表示するには、規則に UserMembersRule の認可タイプが必要です。

Configuration:AuthorizationTypes オブジェクトには、デフォルトとカスタムの両方の認可タイプが存在します。

認可タイプはリポジトリタイプに依存しません。つまり、1 つの認可タイプを定義して、たとえば、Configuration と Rule の両方のオブジェクトに割り当てることができます。これにより、単一タイプのオブジェクトのリストをフィルタリングするため、または関連するオブジェクトセットへのアクセス権を特定の機能を持つ Identity Manager 管理者のサブセットに付与する手段として、認可タイプを使用できます。

Identity Manager での認可タイプの使用方法

Identity Manager では、呼び出し元の機能をオブジェクトの認可タイプと比較するときのアクセスチェック中に、認可タイプが使用されます。認可タイプによって既存のリポジトリタイプが拡張された場合、アクセス制御は暗黙的な「継承」の変更に従います。特に、管理者が親タイプに権限を持っている場合は、子タイプにも同じ権限を持っています。ただし、管理者が子タイプに権限を持っているが、親タイプには権限がない場合は、子タイプのオブジェクトにしかアクセスできません。

たとえば、次のような認可タイプ、管理者、およびオブジェクトを考えてみます。

認証の設定 :

```
Configuration   (repository type)
<AuthType name='Fruit' extends='Configuration'/>
<AuthType name='Vegetable' extends='Configuration'/>
```

次のように権限が割り当てられているとします。

AdminA (Configuration に対する Right.VIEW を持つ)

AdminB (Fruit に対する Right.VIEW を持つ)

AdminC (Vegetable に対する Right.VIEW を持つ)

Configuration タイプの ObjectA、authtype なし

Configuration タイプの ObjectB、authtype == Fruit

Configuration タイプの ObjectC、authtype == Vegetable

前述の認証設定によって、特定のオブジェクトに対して次のようにアクセス権限が決まります。

- AdminA は ObjectA、ObjectB、および ObjectC を参照できる
- AdminB は ObjectB のみを参照できる
- AdminC は ObjectC のみを参照できる

認可タイプを使用する理由

配備内で認可タイプを使用して、次の処理を実行できます。

- 単一のオブジェクトタイプのリストを 1 つの目的に固有のオブジェクトに制限する (SubType と非常に類似)。たとえば、**<AuthType name='foo' extends='moo'/>**
- 管理者の特定クラスで使用できるように、さまざまなタイプのオブジェクトをグループ化する。たとえば、**<AuthType name='foo' extends='red,green,blue'/>**

親タイプ (red、green、blue) に対する権限を持つ管理者は「foo」タイプへのアクセス権も持つことになるため、この 2 番目のアプローチの方がより高度な設定が必要です。

アーキテクチャー機能

認可タイプの主要なアーキテクチャー機能は、Configuration:AuthorizationTypes オブジェクトです。このオブジェクトを変更すると、認可タイプを追加したり、削除したりできます。

Configuration:AuthorizationTypes オブジェクト

Configuration:AuthorizationTypes オブジェクトでは、有効な認可タイプが定義されます。各認可タイプは、<AuthType> 要素で宣言されます。

```
<AuthType name='SPML' extends='Configuration'/>
```

AuthTypes 要素には、AuthType 要素のリストが含まれています。各 AuthType には、最低でも name 属性および一般に extends 属性があります。extends 属性の値は、別の認可タイプまたはリポジトリタイプの名前にする必要があります。

AuthType 要素

この要素には name プロパティーが必要です。次の例は、<AuthType> 要素の正しい構文を示します。次の例は、複数のユーザーを新しい組織に移動するカスタムタスクの追加方法を示します。

```
<Configuration name='AuthorizationTypes'>
   <Extension>
      <AuthTypes>
         <AuthType name='Move User'
extends='TaskDefinition,TaskInstance,TaskTemplate'/>
      </AuthTypes>
   </Extension>
</Configuration>
```

AuthType 要素では、次の属性がサポートされています。

表 2-1 AuthType 属性

AuthType オブジェクト属性	説明
name	認可タイプを識別します。
extends	このタイプのスーパータイプである認可タイプリポジトリタイプの名前を指定します。
displayName	このタイプの代替表示名 (一般にはメッセージカタログキー) を指定します。
auditKey	このタイプのオブジェクトに関連付けられた監査レコードで使用される監査ログキーを識別します。何も指定しない場合は、ベースタイプの監査キーが使用されます。
allowedRights	権限名をコンマで区切ったリストを指定します。これにより、アクセス権の定義でこの認可タイプと併用できる権限が定義されます。指定しない場合は、すべての権限が許可されます。

認証サブタイプアクセス権

Identity Manager では、認可タイプのスーパータイプを定義するために、extends 属性が使用されます。スーパータイプアクセス権はサブタイプによって継承されます。たとえば、ユーザーが TaskDefinition に対する参照権限を持っている場合は、UsageReportTask およびその他すべての TaskDefinition のサブタイプに対する参照権限も持っています。

AuthorizationTypes オブジェクトは XML でのみ編集できますが、認可タイプを参照するアクセス権は「機能」ページから定義できます (「セキュリティ」タブの「機能」サブタブで、このページにアクセスできる)。

認可タイプと機能

認可タイプは、エンドユーザー権限モデルの主要なコンポーネントです。認可タイプを使用すると、機能 (AdminGroups) を定義してから、その機能をユーザーに割り当てることができます。

AdminGroups

認可タイプを定義したあとは、AdminGroup オブジェクト内に格納された Permission オブジェクトで参照できます。次の XML の例では、ユーザーに割り当てることができる AdminGroup (機能 と呼ばれる) を定義します。

コード例 2-1

```
<AdminGroup name='EndUser'>
   <Permissions>
       <Permission type='EndUserTask' rights='View'/>
       <Permission type='EndUserRule' rights='View'/>
   </Permissions>
 <MemberObjectGroups>
 <ObjectRef type='ObjectGroup' id='#ID#All' name='All'/>
 </MemberObjectGroups>
</AdminGroup>
```

この例では、2 つの Permission 要素の両方で、リポジトリタイプではなく認可タイプのタイプ名が使用されます。EndUserTask 認可タイプが割り当てられた TaskDefinition オブジェクトのみが、この機能を持つユーザーにアクセスできます。権限のセットは、機能によって 1 つ以上の認可タイプまたはリポジトリタイプに伝達されます。基本的に、認可タイプはその他の認可タイプおよびリポジトリタイプとの階層構造であるため、「タイプ階層」の親に権限が存在すると、すべての子にも同じ権限が付与されます。

EndUser 機能

AdminGroup EndUser 機能を使用すると、通常は割り当てられた機能がなく、いずれの組織も制御しない管理側以外のユーザーにアクセス権を割り当てることができます。この機能のデフォルト定義については、「アクセス権の拡張」セクションの例で示されています。

Identity Manager では、暗黙的にすべてのユーザーに EndUser 機能が割り当てられます。この機能では、ユーザーがタスク、規則、ロール、リソースなどの複数タイプのオブジェクトを参照することが許可されます。エンドユーザーに機能を割り当てられますが、割り当てないようにも選択できます。Identity Manager では、明示的に機能が割り当てられたユーザーが管理者として定義され、管理者に関する情報がキャッシュされます。その結果、インストール環境で所有できる管理者数に効果的な上限が設定されます。

EndUserLibrary 認可タイプを使用できます。EndUser 機能 (または AdminGroup) には、authType が EndUserLibrary のライブラリへのリストアクセス権および参照アクセス権があります。

ライブラリの内容へのアクセス権をユーザーに付与するには、authType='EndUserLibrary' を設定し、ライブラリの MemberObjectGroup が All に設定されていることを確認します。

認可タイプを作成する

既存の TaskDefinition、TaskInstance、および TaskTemplate 認可タイプを拡張すると、新しい認可タイプを作成できます。次の方法のいずれかを使用すると、認可タイプを追加できます。

- <AuthType> 要素を使用して、新しい認可タイプを作成します。
- タスクに対して新しい認可タイプ要素 (AuthType) を追加して、リポジトリの Authorization Types Configuration オブジェクトを編集します。

認可タイプをリポジトリに割り当てる

リポジトリで認可タイプを設定すると、特定のオブジェクトタイプを参照、変更、または削除できるユーザーを制限できます。リポジトリタイプに対して認可タイプを定義するには、認可タイプ名をリポジトリタイプの名前に設定し、extends 属性を省略します。

例 : エンドユーザー認可タイプを設定する

Identity Manager では、「User」管理者ロールが実装され、デフォルトですべてのユーザーに割り当てられます。このロールは、さまざまなオブジェクトタイプに対して 2 つのエンドユーザー認可タイプ (AuthType) と複数のリストアクセス権を提供する EndUser AdminGroup をカプセル化します。

これらのエンドユーザー認可タイプは次のとおりです。

- **EndUserRule** - 次のように、オブジェクトで指定された EndUserRule AuthType を持つ規則オブジェクトにアクセスできます。

  ```
  <Rule authType='EndUserRule' ...>
  ```

- **EndUserTask** - 次のように、オブジェクトで指定された EndUserTask AuthType を持つ TaskDefinition オブジェクトにアクセスできます。

  ```
  <TaskDefinition authType='EndUserTask' ...>
  ```

- **EndUserLibrary** - Library オブジェクトの内容にアクセスできます。

  この AuthType を実装するには、AuthType を EndUserLibrary に設定し、ライブラリの MemberObjectGroup が All に設定されていることを確認します。EndUser 機能 (AdminGroup) には、認可タイプが EndUserLibrary のライブラリへのリストアクセス権および表示アクセス権があります。

例 : 認可タイプを使用してリソースの表示を制限する

認可タイプを使用すると、リソースレベルでリソース表示を制限できます。リソースを特別な組織に移動する代わりに、次の処理を実行できます。

- リソース (Resource-Corporate-LDAP など) ごとに認可タイプを定義する
- これらのリソースタイプに対する権限を持つ機能を構築する

ユーザーに機能を割り当てるときは、汎用のリソースタイプへの権限が含まれる機能を割り当てないでください。その代わりに、特定のリソースタイプに対する権限を持つ機能を割り当てます。

注 システムで定義されたストック認可タイプの例については、admingroups.xml ファイルを参照してください。

リソース固有の認可タイプを定義するには、次の処理を実行します。

1. Configuration:AuthorizationTypes オブジェクトにエントリを追加します。

   ```
   <AuthType name='Resource-Corporate-LDAP' extends='Resource'/>
   ```

2. 標準機能 (Resource Administrator など) のいずれかからバリアントを抽出します。この機能と標準 AdminGroup との唯一の相違点は、Permission のタイプ名であり、Resource ではなく Resource-Corporate-LDAP であることに注意してください。

コード例 2-2 リソース固有の認可タイプを定義する

```
<AdminGroup name='Corporate LDAP Resource Administrator'
           protected='true'
           displayName='UI_ADMINGROUP_RESOURCE_ADMIN'
           description='UI_ADMINGROUP_RESOURCE_ADMIN_DESCRIPTION'>
   <AdminGroups>
       <ObjectRef type='AdminGroup' id='#ID#Resource Group Administrator'/>
       <ObjectRef type='AdminGroup' id='#ID#Resource Report Administrator'/>
       <ObjectRef type='AdminGroup' id='#ID#Connect Organizations'/>
       <ObjectRef type='AdminGroup' id='#ID#Connect Policies'/>
   </AdminGroup>
   <Permissions>
       <Permission type='AttributeDefinition' rights='View'/>
       <Permission type='Resource-Corporate-LDAP'
rights='View,List,Create,Modify,Delete,Execute'/>
       <Permission type='ResourceUIConfig' rights='Create,Modify'/>
       <Permission type='Rule' rights='View'/>
       <Permission type='User' rights='View,List'/>
   </Permissions>
   <MemberObjectGroups>
       <ObjectRef type='ObjectGroup' id='#ID#All' name='All'/>
   </MemberObjectGroups>
</AdminGroup>
<ObjectRef type='AdminGroup' id='#ID#Connect Resource Groups'/>
```

例 : アクセス権を Identity Manager の特定部分に付与します

認可タイプを使用すると、Identity Manager の特定部分の詳細な管理制御をユーザーセットに付与することもできます。

AuthType を作成し、その AuthType にオブジェクトを割り当ててから、その AuthType を付与する機能を作成します。この機能をユーザーセットに割り当てた場合、認可タイプおよび機能で参照が許可された、システムの領域のみを参照できます。

次の例では、LimitedReportType 認可タイプを TaskDefinition に割り当て、Run Limited Report 機能をユーザーに割り当てます。最終的に、ユーザーは TaskDefinition が LimitedReportType 認可タイプであるレポートのみを実行できます。

```
<AuthType name='LimitedReportType' extends='TaskDefinition'/>
<AuthType name='LimitedReportType' extends='TaskInstance'/>
<AdminGroup name='Run Limited Report' ...>
...
<Permissions type='LimitedReportType' rights='View,Execute'/>
  ...
</AdminGroup>
```

第 3 章

データ読み込みと同期

この章では、アカウント情報をリソースから Identity Manager に読み込んで同期するために使用できる方法の概要を説明します。

アイデンティティーシステム ユーザーとリソースアカウントの違いを明確に区別することが重要です。次の定義は、このトピックを理解するのに役立ちます。

- **ユーザー** : Identity Manager が管理する仮想アイデンティティーです。Identity Manager ユーザーは複数のアカウントを参照する場合があります。
- **アカウント** : リソース (正確には、Identity Manager 内でリソースオブジェクトとして表現される外部システムまたはアプリケーション) で管理される実際のアイデンティティーです。たとえば、UNIX システムの `/etc/passwd` にあるエントリ、Windows システムの SAM データベースにあるエントリ、RACF のあらゆる表現アカウントにある UserProfile などです。
- **管理者** : Identity Manager システムの設定および保守の責任を負っている人です。

Identity Manager は、既知のリソースアカウントとユーザーの情報を**アカウントインデックス**に格納します。アカウントインデックスの各エントリには、最小限でもアカウント ID と Identity Manager リソース ID が含まれています。エントリには、ネイティブの GUID やアカウントのステータス (有効または無効) などの追加情報が含まれる場合もあります。エントリには、アカウントの所有者として Identity Manager ユーザーの ID も記録でき、所有者候補のリストも記録できます。

この章で説明するトピックは次のとおりです。

データ読み込みの種類

データ読み込みは、アカウント情報をリソースから Identity Manager にインポートし、それらのアカウントを Identity Manager ユーザーに割り当てるプロセスです。Identity Manager は、リソースからアカウントデータを読み込む次の機能をサポートします。

- 「検出」: リソースアカウントを最初に Identity Manager に読み込む基本機能を提供します。
- 「調整」: リソースアカウント情報を定期的に Identity Manager に読み込み、設定されたポリシーに従って各アカウントに適切な処置を行います。
- 「Active Sync」: アイデンティティー情報の源泉として「信頼性の高い」外部リソース (アプリケーションやデータベースなど) に格納されている情報を Identity Manager のユーザーデータと同期できるようにします。Active Sync 対応アダプタは、アイデンティティー情報の源泉として信頼性の高いリソースに対する変更を「待機」(ポーリング) します。

これらの概念について、それぞれ詳しく説明します。データ読み込みの種類を比較した表を「データ読み込みの種類の概要」に示します。

検出

検出プロセスは、リソースを初めて配備するときに使うことを目的に設計されています。アカウント情報を Identity Manager にすばやく読み込むための手段を提供します。そのため、調整や Active Sync に含まれる機能の一部が備えられていません。たとえば、検出プロセスではアカウントインデックスにエントリが追加されません。また、検出の前後にワークフローを実行することもできません。ただし、検出プロセスでは、相関規則が予期したとおりに機能するかどうかをより早く判定できます。

検出プロセスを開始すると、Identity Manager は入力アカウントが既存のユーザーと一致する (つまり相関する) かどうかを判定します。一致した場合、検出プロセスはアカウントをユーザーにマージします。一致しない入力アカウントがある場合は、新しい Identity Manager ユーザーが作成されます。

Identity Manager には、次の検出機能があります。

- 「ファイルから読み込み」: ファイル内に指定されたアカウントを読み取り、Identity Manager に読み込みます。
- 「リソースから読み込み」: リソースからアカウントを抽出し、Identity Manager に直接読み込みます。
- 一括処理の作成 : ファイルに一覧されているユーザー作成コマンドを実行します。

これらの検出プロセスの詳細については、次のセクションを参照してください。

ファイルから読み込み

「ファイルから読み込み」検出プロセスは、XML ファイルまたは CSV (Comma-Separated Value) ファイルに書き込まれたアカウント情報を読み取ります。

一部のリソース (Active Directory など) には、ネイティブのアカウント情報を CSV (Comma-Separated Value) 形式にエクスポートする機能があります。これらの CSV ファイルを使って、Identity Manager アカウントを作成できます。CSV 形式の詳細については、『Identity Manager 管理ガイド』を参照してください。

ファイルから読み込むときは、どのアカウント相関規則および確認規則を使用するかを指定する必要があります。詳細については、「相関と確認の規則」を参照してください。

リソースから読み込み

「リソースから読み込み」機能は、ターゲットシステムを走査し、すべてのユーザーの情報を返します。その後、Identity Manager がユーザーを作成して更新します。リソースから読み込む前に、アダプタがそのリソース用に設定されている必要があります。

リソースから読み込むときには、どのアカウント相関規則および確認規則を使用するかを指定する必要があります。詳細については、「相関と確認の規則」を参照してください。

一括処理の作成

一括処理では、複数のアカウントを同時に処理できます。一括処理を使って Identity Manager アカウントおよびリソースアカウントの作成、更新、および削除ができますが、ここでは Identity Manager アカウントの作成のみについて説明します。一括処理の詳細については、『Identity Manager 管理ガイド』を参照してください。

一括処理は、CSV (Comma-Separated Value) を使って指定します。これらの値の構造は、「ファイルから読み込み」プロセスで指定される値とは異なります。

この CSV 形式は、2 行以上の入力行で構成されます。各行は、コンマで区切られた値のリストで構成されます。1 行目にはフィールド名が入ります。残りの各行は、Identity Manager ユーザー、ユーザーのリソースアカウント、またはその両方に対して実行される処理に対応します。各行には、同じ数の値が含まれるようにしてください。値を空にすると、対応するフィールドの値は変更されません。

一括処理の CSV 入力には、常に次の 2 つのフィールドが必要です。

- user: Identity Manager ユーザーの名前を含みます。
- command: Identity Manager ユーザーに対して実行する処理を含みます。Identity Manager ユーザーを作成する場合は、この値を Create にする必要があります。

3 番目以降のフィールドは、ユーザービューのフィールドです。使用されるフィールド名は、ビュー内の属性のパス表現です。ユーザービューで利用可能な属性の詳細については、『Identity Manager ワークフロー、フォーム、およびビュー』の「ユーザービューについて」を参照してください。カスタマイズしたユーザーフォームを使用している場合は、フォーム内のフィールド名に使用可能なパス表現の一部が表示されます。

一括処理でよく使われるパス表現の一部を次に示します。

- `waveset.roles`: Identity Manager アカウントに割り当てる 1 つ以上のロール名のリストです。
- `waveset.resources`: Identity Manager アカウントに割り当てる 1 つ以上のリソース名のリストです。
- `waveset.applications`: Identity Manager アカウントに割り当てる 1 つ以上のリソースグループのリストです。
- `waveset.organization`: Identity Manager アカウントが配置される組織名です。
- `accounts`[*resource_name*].*attribute_name*: リソースアカウント属性です。属性の名前はリソースのスキーマに一覧されています。

複数の値を指定できるフィールドもあります。たとえば、`waveset.resources` フィールドを使って、1 人のユーザーに複数のリソースを割り当てることができます。
フィールド内の複数の値を区切るには、縦棒 (|) 文字 (「パイプ」文字とも呼ぶ) を使用します。複数の値の構文は、次のようにして指定できます。

```
value0 | value1 [ | value2 ... ] ]
```

一括処理の作成の例を次に示します。

```
command,user,waveset.resources,password.password,password.confir
mPassword,accounts[AD].description,accounts[Solaris].comment
Create,John Doe,AD|Solaris,changeit,changeit,John Doe,John Doe
Create,Jane Smith,AD,changeit,changeit,Jane Smith,
```

一括処理の作成は、「ファイルから読み込み」プロセスより汎用性があります。「ファイルから読み込み」では一度に 1 つのリソースから情報を読み込みますが、一括処理では複数のリソースを処理できます。

調整

調整では、アカウントインデックスの内容と各リソースに現在含まれている内容を比較します。調整では、次の機能を実行できます。

- 新規アカウントと削除されたアカウントの検出
- アカウント属性値の変更の検出
- アカウントと Identity Manager ユーザーの関連付け
- Identity Manager ユーザーと関連のないアカウントの検出
- 検出した個々のアカウント状況に応じたワークフローの実行
- リソース上の特定のコンテナからリソース上の別のコンテナにユーザーを移動した時刻の検出

注 調整を行う前に、アダプタがそのリソース用に設定されている必要があります。アダプタの詳細については、『Identity Manager リソースリファレンス』を参照してください。

調整には、完全調整と差分調整の 2 つの種類があります。

完全調整

完全調整は、アダプタがリストする各アカウント ID に対して、存在、所有権、状況を再計算します。これは所有権を再計算するために、リソースを要求する各 Identity Manager ユーザーをチェックします。

Identity Manager ユーザーは、次の方法でリソースを要求できます。

- リソースを間接的に指定するロールを使用する方法
- 直接的なリソース割り当てを使用する方法
- このリソースのアカウントを参照する方法
- リソースグループを使用する方法

調整プロセスは、アカウントインデックスに記録された Identity Manager 所有者がまだ存在し、アカウントを要求していることを、アカウントごとに確認します。所有者のいないすべてのアカウントは、このリソースの調整ポリシーで相関規則が指定されているかぎり、Identity Manager ユーザーと関連付けられます。相関規則で 1 人以上の所有者候補がいることがわかると、それぞれの候補が確認規則で二重にチェックされます (指定されている場合)。規則の詳細については、「相関と確認の規則」を参照してください。

アカウントに対して状況が設定されると、調整はこのリソースの調整ポリシーで設定された応答を実行します。調整ポリシーがアカウント単位で実行するワークフローを指定している場合、完全調整は、状況処理を実行したあとに、調整するアカウントごとにこれを実行します。ワークフローの詳細については、「調整ワークフロー」を参照してください。

差分調整

差分調整は、差分バックアップと同質のものです。完全調整よりも高速であり、必要な処理のほとんどを実行しますが、完全調整ほどには完璧ではありません。

差分調整では、アカウントインデックスで保守している情報が正確であると信頼しています。既知のアカウント ID のリストが正確であると信頼し、Identity Manager 所有者によるアカウントの所有は正確に記録されていると信頼しているので、差分調整はいくつかの処理段階を省略または短縮することができます。

差分調整は、リソースを要求している Identity Manager ユーザーを検査する手順を省略しています。また差分調整は前回のリソース調整のあとに追加または削除されたアカウントだけを対象にして状況を算出します。リソースのアカウントインデックスにあるアカウント ID のリストとリソースアダプタが返したアカウント ID のリストを比較して、この計算を実行します。新規アカウントは、既存のアカウントとして記録され、削除されたアカウントはもはや存在しないアカウントとして記録されます。これ以降、これらの 2 種類のアカウントだけが処理されます。

差分調整は完全調整よりもはるかに高速であり、使用する処理サイクルも少ないので、差分調整の実行頻度を高くし、完全調整の頻度を低くすると快適な環境にできます。

Active Sync

Active Sync では、リソースに対する変更を「待機」(ポーリング) し、差分変更をリアルタイムで検出します。Active Sync は変更を検出するように設計されているため、アカウント情報をはじめて Identity Manager に読み込むときには使用しないでください。代わりに、調整または検出プロセスを使用します。

一般に、次のような場合は Active Sync リソースに対して調整を実行します。

- リソースの最初の読み込みを実行する場合
- 属性を無視または除外するように Active Sync が設定されているために Identity Manager で更新されていない属性を検出する場合

Active Sync は、次の点で調整と異なります。

- Active Sync では、複数のアカウントにわたる属性の同期が取れているかどうかを確認するためのユーザーフォームを管理者が指定できます。

- すべての Active Sync 処理を完全に制御する処理規則を実装できます。これは、一般にリソース上のアカウントが変更されたときに特別な処理 (リポジトリ内の複数のオブジェクトを編集するなど) を実行する必要がある場合に使用します。

Active Sync では、適切に設定された Active Sync 対応アダプタを使用する必要があります。リソースを設定して Active Sync を実装する方法の詳細については、『Identity Manager 管理ガイド』を参照してください。

データ読み込みの種類の概要

次の表は、検出と調整の機能を比較したものです。

表 3-1 データ読み込みの種類の概要

機能	検出	調整	Active Sync
新規アカウントの検出	あり	あり	あり
削除されたアカウントの検出	なし	あり	あり
アカウント属性値の変更の検出	なし	あり	あり
Identity Manager ユーザーと関連のないアカウントの検出	あり	あり	あり
リソース上の特定のコンテナからリソース上の別のコンテナにユーザーを移動した時刻の検出	なし	あり	あり
アカウントと Identity Manager ユーザーの関連付け	あり	あり	あり
検出した個々のアカウント状況に応じたワークフローの実行	なし	あり	あり
スケジュール可能	なし	あり	あり
差分モード	なし	あり	該当しない
アカウントインデックスへのエントリの追加	なし	あり	あり
複数リソース上の属性の同期	なし	なし	あり

読み込み処理のコンテキスト

次の表には、読み込み処理カテゴリに関連する一般的な Identity Manager プロセスまたはタスクに関する情報を示します。

表 3-2 読み込み処理プロセスまたはタスク

プロセスまたはタスク	使用方法	名前空間
ファイルから読み込み	CVS または XML ファイル (管理者インタフェースから呼び出される) からアカウント情報を取得します。 Identity Manager は、ファイルから `WSUser` オブジェクトを読み込み、それをユーザービューに変換し、フォームを適用します。属性は Identity Manager ユーザーの拡張属性であるかのように処理されます。属性は `accounts[Lighthouse]` に配置され、フォームで属性ごとにグローバルフィールドが定義されている場合は、`global` 属性の下のみに配置されます。	ファイルの各行のすべての属性値は、グローバルな名前空間に配置されます。 `global.`<*attr name*> **注** : `create` 処理のみに適用されます。
リソースから読み込み	特定のリソース (管理者インタフェースから起動され、アダプタを使用してアカウントを一覧表示およびフェッチする) からアカウント情報を取得します。	リソースの各アカウントのすべての属性値は、グローバルな名前空間に配置されます。 `global.`<*LHS Attr Name*> **注** : `create` 処理のみに適用されます。
一括処理	CVS ファイル (管理者インタフェースから起動される) からコマンドおよびユーザービューデータを取得します。 ユーザービュー名前空間で、任意の属性を指定できます。属性名は、ビューパス構文を使用して指定されます。ユーザービュー名前空間およびビューパス構文の詳細については、『Identity Manager ワークフロー、フォーム、およびビュー』の「ユーザービューについて」を参照してください。	ファイルからの属性値は、グローバルな名前空間に配置されます。 • `accounts[*].*` • `waveset.*` • `accountInfo.*` • `global.*` **注** : 許可された利用可能セッションはありません。

調整を管理する

調整プロセスは、主に管理者インタフェースを介して管理されます。ただし、このインタフェースから実行できない調整の機能もあります。たとえば、新しい相関規則や確認規則の作成、調整ワークフローの作成、または調整設定オブジェクトの編集が必要な場合があります。以降の項では、これらの機能およびその他について説明します。調整ポリシーの定義の概要については、『Identity Manager 管理ガイド』を参照してください。

調整ポリシー

調整ポリシーでは、調整タスクごとに (リソースによる) 一連の応答を設定できます。ポリシー内では、調整を実行するサーバーを選択し、調整の実行回数と実行時間を決定し、調整中に発生する個々の状況に対する応答を設定します。アカウント属性に対してネイティブに (Identity Manager を介さずに) 行われた変更を検出するように調整を設定することもできます。

これらの各ポリシー設定は、異なる範囲で定義できます。

- グローバルに (全リソースタイプに対して)
- 特定のリソースタイプに対して
- 個々のリソースインスタンスに対して

それぞれの範囲の値は、下位の各範囲のデフォルトになります。したがって、調整ポリシーは、継承ツリーを定義します。

- グローバルの値は、全リソースタイプのデフォルトになります。
- 各リソースタイプは、グローバル値を継承するか、特定の値を指定します。
- リソースタイプ値は、このタイプの全リソースインスタンスのデフォルトです。
- 各リソースインスタンスは、親リソースタイプの値を継承できます。また特定の値を指定できます。

継承は、リソースが多い場合に、ポリシーの管理を容易にします (特に多くのリソースが同じ設定である場合)。

たとえば、すべてのリソースを同じ方法で処理したい場合、1 セットのポリシー設定をグローバルレベルで管理するだけで済みます。すべての Windows リソースを 1 つの方法で処理し、すべての Solaris リソースを別の方法で処理したい場合、2 つのリソースタイプにそれぞれ 1 つずつポリシーを設定し、それぞれの範囲内でポリシー設定を管理する必要があります。一部のリソースインスタンスのリソースタイプレベル

で定義するポリシーに例外がある場合、これらの個々のリソースに対応するために、必要なポリシー設定を優先する (つまり、指定する) ことができます。各ポリシー設定は個々に継承されるので、指定する必要があるのは異なる設定だけです。その他のポリシー設定は、以前と変わらずに、上位から値を継承することができます。

相関と確認の規則

Identity Manager は、ユーザーにリンクされていないリソースアカウントと Identity Manager ユーザーを次の 2 段階で照合します。

- **相関** - 所有者の候補を検出します。
- **確認** - それぞれの所有者候補をテストします。

相関規則は、アカウントを所有している可能性がある Identity Manager ユーザーを検索します。確認規則は、ユーザーが実際にアカウントを所有しているかどうかを確認するために、アカウントに対して Identity Manager ユーザーをテストします。所有者候補をすばやく検索したり (名前または属性に基づいて)、負担の大きなチェックの実行を所有者候補だけに制限したりするので、この 2 段階のアプローチによって、Identity Manager は相関を最適化することができます。

調整ポリシーでは、各リソースに対して、相関規則と確認規則を選択できます。また、「確認規則なし」も指定できます。デフォルトの相関規則は、入力したアカウントのアカウント ID と厳密に一致する名前を持つユーザーを検索します。デフォルトでは、「確認規則なし」が使用されています。

注	相関規則と確認規則は、検出や Active Sync でも使用できます。

Identity Manager は、いくつかの相関規則と確認規則を `sample/reconRules.xml` 内で事前に定義しています。また独自の相関規則や確認規則を記述することもできます。`SUBTYPE_ACCOUNT_CORRELATION_RULE` または `SUBTYPE_ACCOUNT_CONFIRMATION_RULE` というサブタイプを持つすべての規則オブジェクトは、適切な調整ポリシー選択リストに自動的に表示されます。

相関規則

相関規則は、リソースアカウントの属性値に基づいて、ユーザー名のリストを生成できます。また相関規則は、ユーザーの選択に使用する属性条件のリストを生成できます (ユーザーオブジェクトのクエリー可能な属性を参照)。

要求されていない各アカウントに対して、相関規則は 1 回ずつ実行されます。

注	相関規則は、比較的「負担の大きくない処理」ですが、使用する場合は状況を選ぶべきです。可能であれば、負担の大きな処理は確認規則に割り当ててください。

Identity Manager は、sample/reconRules.xml 内でいくつかの相関規則を事前に定義しています。

- **アカウント ID と一致するユーザー名**：accountId 属性の値を返します。これはリソースアカウント ID と一致する名前を持つあらゆる Identity Manager ユーザーを所有者候補として選択します。これがデフォルトの相関規則です。
- **一致するアカウント ID を所有するユーザー**：属性条件のリストを返します。これは同じ accountId 値に一致するリソースアカウントを持っているすべての Identity Manager ユーザーを所有者候補として選択します。
- **ユーザーの電子メールがアカウントの電子メールに一致**：アカウントの email 属性に基づいて Identity Manager ユーザーを選択する属性条件のリストを返します。

相関規則の入力は、アカウント属性のマップです。出力は次のいずれかでなければなりません。

- 文字列 (ユーザー名または ID を含む)
- 文字列要素のリスト (各ユーザー名または ID)
- WSAttribute 要素のリスト
- AttributeCondition 要素のリスト

より複雑な規則で、アカウント属性値を組み合わせたり、処理したりして、名前のリストや属性条件のリストを生成できます。

注　属性条件は、UserUIConfig オブジェクトの QueryableAttrNames として設定されているクエリー可能な属性を参照する必要があります。

たとえば、reconRules.xml には、相関規則の 4 番目のサンプル、「アカウントフルネームと一致するユーザーフルネーム」があります。追加の設定を実行しなければ、このサンプルは正常に動作しないので、この規則は XML コメントで無効にしてあります。この規則は、fullname に基づいて Identity Manager ユーザーを検索しますが、この属性はデフォルトではクエリー可能になっていません。

拡張属性の相関には、特別な設定が必要です。

- 拡張属性は、UserUIConfig でクエリー可能と指定する必要があります (QueryableAttrNames のリストに追加)。
- UserUIConfig の変更を有効にするには、Identity Manager アプリケーション (またはアプリケーションサーバー) を再起動する必要があります。

確認規則

確認規則は、相関規則が返す一致ユーザーごとに 1 回ずつ実行されます。

一般的な確認規則は、ユーザービューの内部の値とアカウント属性の値を比較します。相関処理のオプションの第 2 段階として、確認規則は相関規則で表現できないチェックを実行します (または、相関規則では負担が大きいので評価できないチェック)。一般的に、確認規則が必要なのは、次の場合だけです。

- 相関規則が一致するユーザーを複数返す可能性がある場合
- 比較する必要があるユーザー値をクエリーできない場合

Identity Manager は、`sample/reconRules.xml` 内で 2 つの確認規則を事前に定義しています。

- **ユーザーの電子メールがアカウントの電子メールに一致** : ユーザーの電子メールがアカウントの電子メールに一致すると、true が値として返されます。ここでは、所有に関する多くの判断を、相関規則または確認規則で表現できるという事実を説明します。ただし、Identity Manager ユーザーの `email` 属性は、自動的にクエリー可能になるので、ほとんどの場合、これを相関規則で表現する方法が、より効率的だと思われます。
- **ユーザーの名前および姓がアカウントと一致** : XPRESS 言語を使用して、ユーザーの名前と姓を、アカウントの対応する値と比較します。

確認規則には、次のものを入力します。

- **ユーザービュー** : Identity Manager ユーザーの完全なビューです。
- **アカウント** : リソースアカウント属性のマップです。

ユーザーがアカウントを所有している場合、確認規則は文字列形式でブール値の true を返します。それ以外の場合、値として false を返します。

デフォルトの確認規則は、「確認規則なし」です。これは、相関規則がアカウントごとにほぼ 1 人のユーザーを検出するという状況を前提としています。相関規則が複数のユーザーを選択する場合、アカウントの状況は「DISPUTED」になります。

調整ワークフロー

ユーザー定義のワークフローに対応するいくつかの接続ポイントを提供することにより、典型的な調整処理を拡張できます。

負荷の大きい (つまり長時間実行する) アカウント単位ワークフローを使用する場合は、『Identity Manager ワークフロー、フォーム、およびビュー』の「ワークフロープロパティーの設定」セクションで説明されているように、デフォルトワークフロー設定を変更することを検討してください。

プレリソースワークフロー

プレリソースワークフローは、ほかの調整処理の前に起動できます。「Notify Reconcile Start」ワークフローは、プレリソースワークフローの例です。

「Notify Reconcile Start」ワークフローは、あるリソースに対する調整が開始したことを通知する電子メールを管理者に送信します。このワークフローを実行する前に、「Notify Reconcile Start」電子メールテンプレートを設定する必要があります。

プレリソースワークフローには、次のパラメータが渡されます。

- **resourceName** - これから調整するリソースの名前です。
- **resourceId** - これから調整するリソースのオブジェクト ID です。

アカウント単位ワークフロー

アカウント単位ワークフローは、応答 (使用する場合) が完了したあと、調整によって処理されるアカウントごとに起動されます。応答のタイプと応答の結果は、このワークフローに影響を与えません。

「Notify Reconcile Response」ワークフローは、アカウント単位ワークフローの例です。調整プロセスが、検出された状況に対して自動的に応答しようとしたときに管理者に電子メールを送信します。このワークフローを実行する前に、「Notify Reconcile Response」電子メールテンプレートを設定する必要があります。

アカウント単位ワークフローには、次のパラメータが渡されます。

- **accountId** - 調整されたリソースアカウントの名前です。
- **resourceId** - 調整しているリソースのオブジェクト ID です。
- **resourceName** - 調整しているリソースの名前です。
- **userID** - アカウントの所有者として識別される Identity Manager ユーザーのオブジェクト ID です (状況に応じて、要求または相関によって識別)。アカウントに関連付けられたユーザーがいない場合、この値は NULL になります。
- **userName** - アカウントの所有者として識別される Identity Manager ユーザーの名前です (状況に応じて、要求または相関によって識別)。アカウントに関連付けられたユーザーがいない場合、この値は NULL になります。
- **initialSituation** - アカウントに関して最初に検出され、応答を開始した状況です。値は有効なメッセージキーです。
- **responseSuccess** - 応答が正常に完了したかどうかを示すブール値です。実行された応答がない場合、値は NULL になります。
- **finalSituation** - 応答を適用したあとのアカウントの調整状況です。アカウントがすでに存在せず、Identity Manager にこのアカウントへの参照がない場合、この値は NULL になります。

ポストリソースワークフロー

ポストリソースワークフローは、ほかの調整処理がすべて完了したあとに起動できます。「Notify Reconcile Finish」ワークフローは、ポストリソースワークフローの例です。

「Notify Reconcile Finish」ワークフローは、あるリソースに対する調整が完了したことを通知する電子メールを管理者に送信します。このワークフローを実行する前に、「Notify Reconcile Finish」電子メールテンプレートを設定する必要があります。

ポストリソースワークフローには、次のパラメータが渡されます。

- **`resourceName`** - 調整したリソースの名前です。
- **`resourceId`** - 調整したリソースのオブジェクト ID です。

ネイティブ変更を監査する

「Audit Native Change To Account Attributes」ワークフローが起動されるのは、調整またはプロビジョナが、Identity Manager による変更ではなく、リソースアカウント属性のネイティブ変更を検出したときです。変更の監視が実行されるのは、ユーザーが指定した属性だけです。デフォルトでは、属性の監視は実行されません。

このワークフローには、次のパラメータが渡されます。

- **`resource`** - アカウントがネイティブに変更されたリソースオブジェクトです。
- **`accountID`** - ネイティブに変更されたリソースアカウントの名前です。
- **`prevAttributes`** - Identity Manager によって記録される監視対象のリソースアカウント属性を含むマップです。
- **`newAttributes`** - リソース上で現在設定されている監視対象のリソースアカウント属性を含むマップです。
- **`attributeChanges`** - どの属性が変更されたかを示す汎用オブジェクトのリストを含むマップです。各オブジェクトには、前の値と新しい値が格納されます。
- **`formattedChanges`** - 監査レコードに適した簡潔な形式で属性の変更を表す文字列です。

ネイティブ変更を監査するには、次を実行する必要があります。

- 「**調整ポリシーの編集**」ページで、「**属性レベル調整**」ドロップダウンメニューから「**アカウント属性へのネイティブ変更を検出する**」オプションを選択します。属性のリストを表示するには、「**リソースタイプポリシーを継承**」チェックボックスの選択を解除する必要がある場合があります。監査する属性を選択します。
- 監査イベントのリストに「**Identity Manager の外部での変更**」を追加します。そのためには、「設定」タブを選択し、左側の「監査イベント」を選択します。

リソースのスケジューリング

調整は、それぞれのリソースのために、差分調整と完全調整という 2 つの別々のスケジュールを保守します。

各リソースは、個々の「要求者」タスクによってスケジュールが管理されます。調整ポリシー GUI でリソースに対して調整スケジュールを設定すると、要求者タスク用の TaskSchedule が設定されます。必要があれば、これで外部のタスクスケジューラによって調整を管理できるようになります。またこれで調整タスクの負担が最小限になります。何も処理を実行していない調整デーモンは、ほとんどリソースを消費しません。このデーモンは、メモリー内のキューを定期的にポーリングしているからです (調整の準備が整ったリソースを探して、データベースのクエリーを実行することはない)。

調整は、リソースアダプタを通じて、各リソースにアクセスします。調整はアダプタを直接的に呼び出して、アカウントのリスト、アカウントの反復、または個人用リソースアカウントのフェッチを実行します。また調整は、プロビジョナを通じて間接的にリソースにアクセスし、プロビジョナを使用してリソースアカウントを作成したり、リソースアカウントから Identity Manager ユーザーを作成したりします。

調整とプロビジョナは、両方ともアカウントインデックスの保守を実行します。また、リソースの調整を行うたびに、アカウントインデックスから不要な情報が除去されます。調整タスクは、Identity Manager ユーザーがアカウントを所有していないかぎり、リソースにもはや存在しないアカウントのエントリを自動的に削除します。したがって、アカウントインデックスにあるリソースの情報を手動でクリアする必要はありません。

各 Identity Manager サーバーは、デーモンタスクとして調整を実行します。つまり、Identity Manager スケジューラは、調整タスクをすぐに開始し、これが停止した場合は、タスクを自動的に再開します。

注　リソース調整は自動的に再開されません。新しい要求に応答できるように、`ReconTask` デーモン自体は、自動的に再起動されますが、ホストサーバーが停止したとき (またはアプリケーションサーバーがシャットダウンされたとき) に実行していた要求は失われます。デーモンはリソース調整を再開しません。要求がないときに、リソースを調整するのは不適切だからです。

調整設定オブジェクト

ReconcileConfiguration オブジェクトには、「調整ポリシーの編集」ページで編集できない複数の属性が含まれます。

次の表で、調整の属性について説明します。ReconcileConfiguration オブジェクト (#ID#Configuration:ReconcileConfiguration) の属性値を編集するには、デバッグページを使用します。

表 3-3　ReconcileConfiguration オブジェクトの主要な属性

属性	説明
fetchTimeout	調整プロセスがアカウントをフェッチするときにリソースからの応答を待機するミリ秒数です。デフォルト値は、1 分 (60,000 ミリ秒) です。
listTimeout	調整プロセスがアカウントをリストするときにリソースからの応答を待機するミリ秒数です。デフォルト値は、10 分 (600,000 ミリ秒) です。
maxConcurrentResources	サーバーが同時に調整するリソースの最大数です。デフォルト値は、3 です。
maxQueueSize	調整サーバーの作業キューに含まれるエントリの最大数です。デフォルト値は、1000 です。

次の例では、ReconcileConfiguration オブジェクトのデフォルト値を示します。

コード例 3-1　ReconcileConfiguration オブジェクトのデフォルト値

```
<Configuration id='#ID#Configuration:ReconcileConfiguration'
name='Reconcile Configuration'>
 <Extension>
  <Object>
    <Attribute name='listTimeout' value='600000' />
    <Attribute name='fetchTimeout' value='60000' />
    <Attribute name='maxConcurrentResources' value='3' />
    <Attribute name='maxQueueSize' value='1000' />
  </Object>
 </Extension>
 <MemberObjectGroups>
  <ObjectRef type='ObjectGroup' id='#ID#All' name='All'/>
 </MemberObjectGroups>
</Configuration>
```

Active Sync を管理する

Active Sync 対応アダプタは、管理者インタフェースで管理できます。このインタフェースには、1 つのアダプタに対する Active Sync のほとんどの設定項目を管理者が完全に設定できるウィザードが含まれます。このウィザードでは、管理者が Identity Manager IDE を使わずにリソース、入力、またはフォームを構築することもできます。Active Sync ウィザードの詳細については、『Identity Manager 管理ガイド』を参照してください。

Active Sync 対応アダプタが機能する仕組み

この項では、次の内容を説明します。

- アダプタ処理の基本的な手順の概要
- Active Sync 変数のコンテキスト
- 規則の使用
- フォームの使用
- ワークフロープロセスの起動

アダプタ処理の基本的な手順

すべての Active Sync 対応アダプタは、Identity Manager で定義されているリソースの変更を知るために、次の基本的な手順でリスニングまたはポーリングを実行します。変更されたリソースを検出すると、Active Sync 対応アダプタは次の手順を実行します。

1. リソースから変更された情報を抽出します。
2. どの Identity Manager オブジェクトに関係があるか判断します。
3. システムに渡すユーザー属性のマップを、アダプタの参照および追加オプションのマップとともに生成します。これにより、Identity Application Programming Interface (`IAPI`) オブジェクトが作成されます。
4. `IAPI` オブジェクトを `ActiveSync` マネージャーに送信します。
5. `ActiveSync` マネージャーは、このオブジェクトを処理し、処理の成功を Active Sync 対応アダプタに通知する `WavesetResult` オブジェクトをアダプタに返します。このオブジェクトには、ID の更新のために Identity Manager システムが使用するさまざまな手順の結果を多く含めることができます。一般に、ワークフローは Identity Manager 内のエラーにも対応し、多くの場合、担当管理者の承認にあとを任せます。

Active Sync 名前空間

次の表には、Active Sync 処理カテゴリに関連する一般的な Identity Manager プロセスまたはタスクに関する情報を示します。

表 3-4　Active Sync プロセスまたはタスク

実行中のプロセスまたはタスク	使用方法	名前空間
ActiveSync `IAPIUser`	• 特定のリソースでユーザー関連の変更を処理します。 • 指定されたワークフロープロセスを起動する前に、完全なユーザービューで直接アクションを実行します。	ActiveSync イベントからの属性をユーザービューにマージします。 入力フォームの一般的な属性は次のとおりです。 • `accounts[*].*` • `waveset.*` • `accountInfo.*` • `activeSync.`<*LHS Attr Name*> • `activeSync.`*resourceName* • `activeSync.resourceId` • `activeSync.resource` • `display.session` (プロキシ管理者用のセッション) • `global.`<*LHS Attr Name*> (リソースに `set globals` フラグが設定されている場合)
ActiveSync `IAPIProcess`	• プロセスビューを作成して、リソース上の一般イベントを処理します。 • プロセスビューの最上位フィールドは、タスクへの任意の入力です。 • `global` 属性でのタスクの起動に関連する属性を収集します。 • 最上位プロパティーとしてではなく、`global` 属性から入力を取得するワークフローを作成します。	最上位のワークフロー `global` 属性にダンプされる ActiveSync ポーリング属性を使用して、指定したタスクを起動します。 ワークフロー属性では次の形式が想定されます。 `global.`<*LHS Attr Name*>

規則を使用する

Active Sync 対応アダプタは、リソース上でのアカウントの変更を検出すると、受け取った属性を Identity Manager ユーザーにマップします。あるいは、一致するアカウントがない場合や、アカウントを作成するように Active Sync リソースが設定されている場合は、Identity Manager ユーザーアカウントを作成します。

Active Sync ウィザードでは、さまざまな条件が発生したときの処理を制御するための規則を指定できます。次の表は、規則の種類についてそれぞれ説明しています。

表 3-5 規則の種類

パラメータ	説明
処理規則	`TaskDefinition` の名前、またはフィード内のすべてのレコードに対して実行される `TaskDefinition` の名前を返す規則のいずれかです。この処理規則は、activeSync 名前空間内のリソースアカウント属性を、リソース ID およびリソース名とともに取得します。 処理規則は、システムがリソース上の変更を検出したときに実行されるすべての機能を制御します。アカウント処理を完全に制御する必要がある場合に使用します。この結果、処理規則はほかのすべての規則より優先されます。 処理規則が指定されると、このアダプタ上にほかのどんな設定があっても、すべての行に対してその処理が実行されます。 処理規則は、少なくとも次の機能を実行する必要があります。 • 一致するユーザービューに対するクエリー。 • ユーザーが存在する場合は、ビューのチェックアウト。ユーザーが存在しない場合は、ユーザーの作成。 • ビューの更新またはビューへの設定。 • ユーザービューのチェックイン。 ユーザー以外のオブジェクト (LDAP ロールなど) を同期することもできます。

表 3-5　規則の種類 (続き)

パラメータ	説明
相関規則	リソースアカウントを所有する Identity Manager ユーザーのリソース情報が特定されない場合、Identity Manager は相関規則を呼び出し、(アカウントの名前空間内の) リソースアカウント属性に基づいて、ユーザーの照合に使用する、一致する可能性のあるユーザーまたはアカウント ID の候補のリスト、あるいは属性条件を特定します。 規則は、エントリを既存の Identity Manager アカウントに関連付けるために使用できる次のいずれかの情報を返します。 • Identity Manager ユーザー名 • WSAttribute オブジェクト (属性ベースの検索に使用) • AttributeCondition 型または WSAttribute 型の項目のリスト (AND 結合による属性ベースの検索) • String 型の項目のリスト (各項目は Identity Manager アカウントの Identity Manager ID またはユーザー名) 相関規則によって複数の Identity Manager アカウントが識別された場合は、複数の一致を処理するために確認規則またはプロセス解決規則が必要です。 データベーステーブル、フラットファイル、および PeopleSoft コンポーネントの Active Sync アダプタの場合は、デフォルトの相関規則はリソース上の調整ポリシーから継承されます。 調整と Active Sync で同じ相関規則を使用できます。詳細については、「相関と確認の規則」を参照してください。
確認規則	相関規則によって返されるすべてのユーザーを対象にして評価される規則です。ユーザーごとに、Identity Manager の ID と (「account.」名前空間にある) リソースアカウント情報の相関を示す完全なユーザービューが確認規則に渡されます。確認規則は、ブール値で表すことができる値を返すことが期待されます。たとえば、「true」、「1」、「yes」、および「false」、「0」、「null」です。 データベーステーブル、フラットファイル、および PeopleSoft コンポーネントの Active Sync アダプタの場合は、デフォルトの確認規則はリソース上の調整ポリシーから継承されます。 調整と Active Sync で同じ確認規則を使用できます。詳細については、「相関と確認の規則」を参照してください。

表 3-5　規則の種類 (続き)

パラメータ	説明
削除規則	activeSync. または account. という形式のキーを持つ値すべてのマップを期待できる規則です。プロキシ管理者のセッションに基づく LighthouseContext オブジェクト (display.session) は、この規則のコンテキストで利用できます。この規則は、ブール値で表すことができる値を返すことが期待されます。たとえば、「true」、「1」、「yes」、および「false」、「0」、「null」です。 あるエントリに関してこの規則によって true が返された場合、アダプタの設定方法に応じて、フォームとワークフローを介してアカウント削除要求が処理されます。
プロセス解決規則	TaskDefinition の名前、またはフィード内のあるレコードに対して複数の一致がある場合に実行される TaskDefinition の名前を返す規則のいずれかです。プロセス解決規則は、リソースアカウント属性をリソース ID およびリソース名とともに取得します。 この規則は、一致がなく、「**一致しないアカウントの作成**」が選択されていない場合にも必要です。 このワークフローは、管理者による手動処理を求める処理にすることもできます。
一致しないアカウントの作成	true に設定すると、一致する Identity Manager ユーザーが見つからない場合に、リソース上にアカウントが作成されます。false に設定すると、処理規則が設定され、その規則が識別するワークフローによって新しいアカウントが保証されていることが確認されないかぎり、Identity Manager はアカウントを作成しません。デフォルトは true です。
グローバルで利用	true に設定すると、activeSync 名前空間に加えてグローバル名前空間にも値が入力されます。デフォルト値は、false です。

アダプタがユーザーを検出しなかった場合

Identity Manager が既存の Identity Manager ユーザーと一致するものを見つけられない場合は、「一致しないアカウントの作成」が true に設定されているか、プロセス解決ワークフローで feedOp of create が指示されていれば、更新処理が作成処理に変更されます。

feedOp フィールドを利用できるのは、ユーザーの作成、削除、または更新を実行するロジックを設定したフォームです。このフィールドを使用すると、特定イベントの固有なフィールドを無効化、あるいは有効化することができます (たとえば、feedOp フィールドが作成に設定されると、パスワードを生成するなど)。

この例の feedOp フィールドがパスワードを作成するのは、Active Sync 対応アダプタが Identity Manager ユーザーと一致しないユーザーをリソース内で検出し、Identity Manager 内にこのユーザーを作成するときだけです。

コード例 3-2 feedOp フィールドの例

```
<Field name='waveset.password'>
   <Disable>
      <neq>
         <ref>feedOp</ref>
         <s>create</s>
      </neq>
   </Disable>
   <expression>
      <cond>
         <notnull>
            <ref>activeSync.password</ref>
         </notnull>
         <ref>activeSync.password</ref>
         <s>change12345</s>
      </cond>
   </expression>
</Field>
```

フォームを使用する

一般的に、Active Sync 対応アダプタは、処理中にリソースフォームとユーザーフォームの 2 つのフォームを使用します。

フォーム処理は、次の 3 つの手順で実行されます。

1. Active Sync フィールドには、属性とリソース情報が入れられます。activeSync 名前空間を使用してリソース上の属性を取得および設定します。
2. リソースフォームが拡張され、取得されます。この拡張処理の間、すべてのユーザービュー属性が利用できます。
3. ユーザーフォームが拡張され、取得されます。

$WSHOME/sample/forms ディレクトリには、末尾に ActiveSyncForm.xml が付けられたサンプルフォームがあります。これらのフォームには、新規ユーザーと既存ユーザーの対応を分けるロジックのほかに、リソース上で削除を検出したときに Identity Manager ユーザーの無効化または削除を行うロジックも含まれています。

注 リソースフォームにはリソース固有のロジックだけを設定し、ユーザーフォームに共通のロジックを設定します。リソース固有のロジックは、feedop フィールドが NULL 以外の値になったときに有効になります。リソースフォームを「なし」に設定すると、すべての Active Sync 属性 (accountId を除く) がグローバル属性と見なされ、自動的にその設定がほかの属性に反映されます。

リソースフォーム

リソースフォームは、リソースの作成または編集時に、管理者がプルダウンメニューから選択するフォームです。選択したフォームの参照は、リソースオブジェクトに保存されます。

リソースフォームは、次のように Active Sync 対応アダプタで使用されます。

- スキーママップから受け取った属性を変換します。
- パスワード、ロール、および組織などのフィールドを生成します。
- 新規ユーザーと既存ユーザーの対応を分けるロジック、削除を検出したときに Identity Manager ユーザーの無効化または削除を行うロジックなど、カスタマイズした処理のために、簡単な制御ロジックを提供します。
- ユーザーフォームが入力として使用するフィールドに、ActiveSync から取得した属性をコピーし、オプションでこの値を変換します。作成処理に必要なフィールドは、waveset.accountId と waveset.password です。ほかのフィールドも設定できます (たとえば、accounts[AD].email または waveset.resources)。
- IAPI.cancel に true を設定して、ユーザーの処理をキャンセルします。多くの場合、これは特定のユーザーの更新を無視するために使用します。

  次の例では、「Doe」という姓を持つ全ユーザーを無視する簡単なフィールドを説明します。

コード例 3-3　「Doe」という姓を持つ全ユーザーを無視するフィールド

```
<Field name='IAPI.cancel'>
   <Disable>
      <neq>
         <ref>activeSync.lastName</ref>
         <s>Doe</s>
      </neq>
   </Disable>
   <expression>
      <s>true</s>
   </expression>
</Field>
```

リソースフォームには、新規ユーザーと既存ユーザーの対応を分けるロジック、削除を検出したときに Identity Manager ユーザーの無効化または削除を行うロジックがあります。

ユーザーフォーム

ユーザーフォームは、Identity Manager インタフェース上での編集作業に使用します。プロキシ管理者をアダプタに割り当てることで、ユーザーフォームを割り当てます。プロキシ管理者が自分に関連付けられたユーザーフォームを持っている場合、処理の実行時に、このフォームがユーザービューに適用されます。

プロキシ管理者とユーザーフォーム

`ProxyAdministrator` 属性を使用して、アダプタに対してプロキシ管理者を設定します。この属性は、あらゆる Identity Manager 管理者に設定できます。すべての Active Sync 対応アダプタの処理は、プロキシ管理者が実行したかのように実行されます。プロキシ管理者が割り当てられていない場合、デフォルトユーザーフォームが指定されます。

属性を処理する代替フォーム

ベストプラクティスとしては、名と姓からフルネームを導出するなど、共通の変更はユーザーフォームに設定することをお勧めします。リソースフォームには、HR ステータスの変更時にユーザーを無効化するなど、リソース固有の変更だけを設定します。ただし、目的の属性が incoming など共通のパスに入れられたあとは、組み込みフォームなどに代替的に共通の変更を設定できます。

```
<Form>
   <Field name='incoming.lastname'>
      <ref>activeSync.lastname</ref>
   </Field>
   <Field name='incoming.firstname'>
      <ref>activeSync.firstname</ref>
   </Field>
</Form>
```

これ以降、共通のフォームでは、共通のロジックのために incoming.*xxx* を参照します。

```
<Form>
   <Field name='fullname'>
      <concat>
          <ref>incoming.firstname</ref>
          <s> </s>
          <ref>incoming.lastname</ref>
      </concat>
   </Field>
</Form>
```

キャンセル処理の実行

ユーザーの処理をキャンセルするには、リソースフォームで IAPI.cancel に true を設定します。これを使用すると、特定のユーザーへの更新を無視できます。

注 Active Sync フォームで IAPI.cancel に true の値が設定された場合、IAPIUser イベントまたは IAPIProcess イベントに関連付けられたプロセスは起動されません。

次の例では、「*Doe*」という姓を持つ全ユーザーを無視する、リソースフォーム内の簡単なフィールドを説明します。

```
<Field name='IAPI.cancel'>
   <Disable>
      <eq><ref>activeSync.lastName</ref><s>Doe</s></eq>
   </Disable>
   <Expansion>
      <s>true</s>
   </Expansion>
</Field>
```

ワークフロープロセスを起動する

管理者は、Active Sync ウィザードを使ってポーリング前またはポーリング後のワークフローを指定できます。これらのワークフローは、「調整ワークフロー」で説明したワークフローと概念上ほぼ同じです。

一部の Active Sync 対応アダプタは、取得したユーザービューの変更をチェックする代わりに、指定されたワークフローを実行するリソース属性をサポートします。このワークフローは、Active Sync データだけで構成される入力変数で実行されます。個々のプロセスをサポートしないアダプタの場合、または標準的なユーザーフォームを使用してプロセスを起動したいアダプタの場合、オプションを設定してプロセスよりも優先させることができます。

```
<Form>
   <Field name='sourceOptions.Process'>
      <Expansion>
         <s>My workflow process name</s>
      </Expansion>
   </Field>
</Form>
```

フォームで指定したワークフローは、標準的なプロビジョニングワークフローと同じように呼び出されます。カスタムワークフローは、標準的な作成ワークフローや更新ワークフローに基づいて作成することを強くお勧めします。`workflow.xml` 内のユーザー作成ワークフローやユーザー更新ワークフローを参考にしてください。

例 : Active Sync 対応アダプタからアカウントを無効化する

この例では、企業における従業員の現在のステータスでリソース (HR データベース) を更新できます。Active Sync 対応アダプタは、この HR データベースからの入力に基づいて、Identity Manager リポジトリを更新し、企業全体を通じてユーザーアカウントの無効化、削除、作成、あるいはその他の処理を実行できます。

次のコード例では、Status という属性を受け取り、これがアクティブ (「A」) でない場合、従業員の全アカウントを無効化します。次の表では、この属性の 4 つの状態を示します。

表 3-6 属性の状態

状態	説明
A	アクティブ
T	終了
L	中断
S	変更の保留

Status 属性の値に基づいて、アカウントの有効化または無効化を設定できます。

コード例 3-4 受け取ったアクティブでない Status 属性に基づくアカウントの無効化

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE Configuration PUBLIC 'waveset.dtd' 'waveset.dtd'>
<Configuration wstype='UserForm' name='PeopleSoft ActiveSync Form'>
   <Extension>
   <Form>
<!-- this is a sample of how to map the accountID to a different
field than the one from the schema map
Commented out because we want to use the default account
ID mapped from the resource Schema Map.
      <Field name='waveset.accountId'>
         <Disable><neq><ref>feedOp</ref><s>create</s></neq></Disable>
         <Expansion>
            <concat><s>ps</s><ref>waveset.accountId</ref></concat>
         </Expansion>
      </Field>
    -->

    <!-- this is the real one, limited to create -->
```

コード例 3-4 受け取ったアクティブでない Status 属性に基づくアカウントの無効化 (続き)

```
      <Field name='waveset.accountId'>
         <Disable><neq><ref>feedOp</ref><s>create</s></neq></Disable>
         <Expansion>
            <ref>activeSync.EMPLID</ref>
         </Expansion>
      </Field>

    <!-- we need to make up a password for accounts that are being
            created.This picks the last six digits of the SSN.
    -->
      <Field name='waveset.password'>
         <Disable><neq><ref>feedOp</ref><s>create</s></neq></Disable>
         <expression>
            <s>change123456</s>
         </expression>
      </Field>

      <Field name='waveset.resources'>
<!--       <Disable><neq><ref>feedOp</ref><s>create</s></neq></Disable> -->
<!-- Don't change the resources list if it already contains peoplesoft -->
         <Disable>
            <member>
               <ref>activeSync.resourceName</ref>
               <ref>waveset.resources</ref>
            </member>
         </Disable>

         <expression>
            <appendAll>
               <ref>waveset.resources</ref>
               <ref>activeSync.resourceName</ref>
            </appendAll>
         </expression>
      </Field>

    <!-- Status is mapped by the schema map to PS_JOB.EMPL_STATUS which
has at least four states - A for active, T terminated,
L laid off, and S which is a pending change.
The audit data tells us what the state was, and the global
data tells us what it is. Based on the change we can disable
or enable the account
Note that this can happen on a create also!
    -->
      <Field>
         <Disable><eq><ref>activeSync.Status</ref><s>A</s></eq></Disable>
      <Field name='waveset.disabled'>
```

コード例 3-4 受け取ったアクティブでない Status 属性に基づくアカウントの無効化 (続き)

```
         <Expansion>
            <s>true</s>
         </Expansion>
      </Field>
      <FieldLoop for='name' in='waveset.accounts[*].name'>
      <Field name='accounts[$(name)].disable'>
         <expression>
            <s>true</s>
         </expression>
      </Field>
      </FieldLoop>

      </Field>

<!-- Status is mapped by the schema map to PS_JOB.EMPL_STATUS which has at
least four states - A for active, T terminated, L laid off, and S which is a
pending change.

This is the enable logic. It is disabled if the account status is <> A or is
already enabled
-->
      <Field>
         <Disable>
            <neq>
               <ref>activeSync.Status</ref>
               <s>A</s>
            </neq>
         </Disable>

      <Field name='waveset.disabled'>
         <Disable><eq><ref>waveset.disabled</ref><s>false</s></eq></Disable>
            <Expansion>
               <s>false</s>
            </Expansion>
      </Field>

      <FieldLoop for='name' in='waveset.accounts[*].name'>
         <Field name='accounts[$(name)].disable'>
            <Expansion>
               <s>false</s>
            </Expansion>
         </Field>
      </FieldLoop>
      </Field>
```

コード例 3-4 受け取ったアクティブでない Status 属性に基づくアカウントの無効化 (続き)

```
   </Form>
   </Extension>
   <MemberObjectGroups>
      <ObjectRef type='ObjectGroup' id='#ID#Top' name='Top'/>
   </MemberObjectGroups>
</Configuration>
```

第 4 章

データ読み込みのシナリオ

この章では、アカウント情報を Identity Manager に読み込むための準備をするときに考慮すべきヒントを紹介します。遭遇する可能性のあるいくつかの問題について解説するためにシナリオ例も示します。

環境を評価する

Identity Manager へのユーザーアカウント情報の読み込みを開始する前に、次の質問の中で自分の環境に該当するものがあるかどうかを判断してください。

- すべてのユーザー ID に対して、アイデンティティー情報の源泉として信頼性の高いリソースが存在しますか。

  答えが「はい」の場合、Identity Manager へのユーザーアカウントの読み込みは容易です。そのリソースを最初のリソースとして使用し、ほかのリソースからアカウントを読み込み、相関規則を使用してアカウントどうしをリンクします。

- 重複しているが相関キーが存在するリソースから、ユーザーの完全なリストを取得できますか。

  答えが「はい」の場合、ユーザーアカウントを読み込むプロセスはほぼ同じになります。ほかのリソースからアカウントを読み込む前に、必ず重複のあるリソースからユーザーアカウントが Identity Manager に読み込まれるようにしてください。

- 重複していて相関キーが存在しないリソースからユーザーのリストを取得できますか。

  答えが「はい」の場合、それらのリソースからユーザーの一意の集合を最も簡単に検出できる方法を判断します。多くの場合、リソースごとにユーザーを手動で相関させたり削除したりする必要があります。

これらすべての質問に対して答えが「いいえ」の場合、アカウントを読み込むプロセスは難しくなります。可能な範囲でユーザーアカウントを読み込み、ほかの Identity Manager ユーザーの作成は部署の管理者またはエンドユーザーに委任することを検討してください。

最初のリソースを選択する

Identity Manager にアカウントを読み込むために使用する最初のリソースは、次の特徴を備えていることが理想的です。

- **広範囲のユーザーの集合を参照していること。**最初の読み込みの目的は、できるかぎり多くのアカウントを Identity Manager に取り込むことです。したがって、次のアプリケーションが選択肢として適しています。
  - PeopleSoft や SAP などの人事アプリケーション (アプリケーションに契約社員やその他の臨時職員のデータが含まれていない場合は、それらの従業員のアカウントを別途読み込むようにしてください)。
  - LDAP や Active Directory などのディレクトリベースアプリケーション。多くの場合、大半のユーザーが中央の組織または組織単位に定義されます。
- **Identity Manager のアカウント ID を構築するための十分な情報を保持していること。**個々の Identity Manager アカウント ID は一意である必要があります。一意性が保証されており、Identity Manager のアカウント ID として使用できる属性がリソースに存在していることが理想的です。そのような属性の例としては、従業員 ID や Active Directory の sAMAccountName 属性などがあります。姓と名を組み合わせてアカウント ID を作成することもできますが、この方法では一意の Identity Manager アカウントが作成されることを保証できません。

注　現在、Lighthouse アカウントは *Identity Manager* アカウントと呼ばれています。カスタムカタログを使用することによって、この名前の変更を上書きできます。

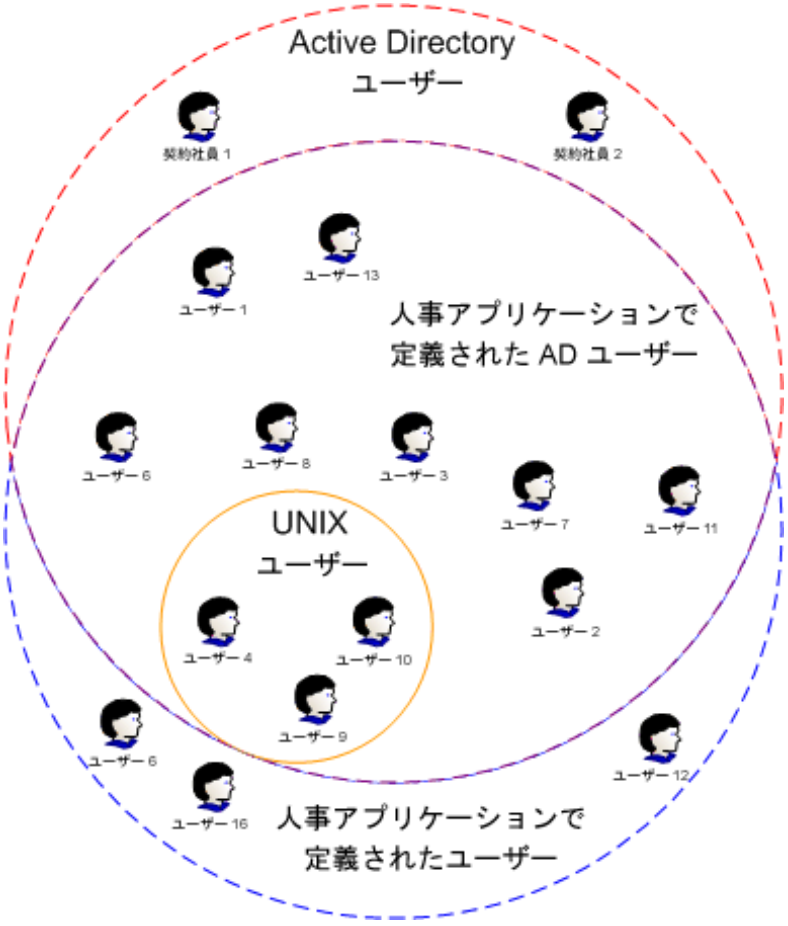
カスタムカタログの詳細については、付録 B「国際化サポートを有効にする」を参照してください。

- **相関キーとして使用できるユーザー属性を格納していること。**Identity Manager でリソースアカウントをリンクするには、複数のリソースにまたがって同じ値を持つ属性が必要です。二次リソースでの値が常に、最初のリソースでの値と完全に一致するのが理想的です。さらに、二次リソースでの値がそのリソース内で一意であればなお理想的です。最も適した属性としては従業員 ID やフルネームがありますが、それら以外でも、すでにあるもので、複数のリソース間で値が一貫していればどのような属性でも使用できます。

- **アイデンティティー情報の源泉として信頼性が高いとみなせるデータを保持していること。**ユーザーが自分自身のアカウントデータを編集できる場合、システム間でデータの一貫性が失われる可能性があります。

次の図は、3 種類のリソースが存在するある会社のシナリオを例示したものです。この会社のほとんどの従業員は、PeopleSoft や SAP などの人事アプリケーションで定義されています。ただし、この会社では契約社員の情報を人事アプリケーションに入力しないため、このアプリケーションを使用して Identity Manager に契約社員のデータを読み込むことはできません。Active Directory でもほとんどのユーザーを定義していますが、すべてを網羅しているわけではありません (コンピュータへのアクセスを必要としない工場従業員などがこれに該当する場合がある)。このように、大部分のユーザーは両方のリソースで定義されていますが、どちらのリソースもすべてのユーザーを網羅しているわけではありません。一部の従業員は UNIX アカウントも持っています。

図 4-1　小規模なデータ読み込みのシナリオ



どのリソースを最初のリソースとして選択すべきでしょうか。UNIX リソースはごく少数のユーザーしか網羅していないため、問題なく検討から外すことができます。Active Directory と人事アプリケーションにはほとんど同数のユーザーが登録されているため、どちらにも明確な利点はありません。

Active Directory と人事アプリケーションのどちらのリソースを最初に読み込むべきかの判断材料となる要素として、次の点が挙げられます。

- Identity Manager 内でアカウントを管理する緊急性。人事アプリケーションでのみ定義され Active Directory では定義されていない従業員に、UNIX やその他のシステムのアカウントなど、追加のリソースアカウントが一切付与されていない場合、人事アプリケーションは Active Directory リソースほどには重要でない可能性があります。

- 相関キー。UNIX アカウントにも存在する複数の属性が一方のリソースに存在する場合、その属性がより適切な選択肢である場合があります。
- Identity Manager ログイン名。一方のリソースが、Identity Manager のログイン名としてより適切な名前を生成する場合、これが決定的な要因となる可能性があります。

最初のデータ読み込みプロセスを選択する

ユーザーデータを Identity Manager に読み込むために最初に使用するリソースを選択したあとは、使用するプロセスを決定する必要があります。次の表は、各種のデータ読み込みプロセスの長所と短所をまとめたものです。それぞれのデータ読み込みプロセスについては、次節以降で詳しく検討します。

表 4-1 データ読み込みプロセスの概要

データ読み込みプロセス	長所	短所
ファイルから読み込み	• 最も高速な読み込みプロセス。 • どの属性を読み込むかの制御が容易。 • 調整よりも設定が容易で高速。	• 顧客側にリソースから CSV ファイルを生成する時間が必要。 • 本稼働環境への投入前に完全調整が必要。 • アカウントの更新には使用できない。
リソースから読み込み	• すべてのリソースに対して有効 • 調整よりも設定が容易	• 読み込むリソースアカウントを自由に選択できない。 • 本稼働環境への投入前に完全調整が必要。
一括処理の作成	1 名の Identity Manager ユーザーに複数のアカウントを同時に追加できます。	• リソースからの読み込みや調整と比べて低速。 • リソースから CSV ファイルを容易に生成できない。 • この機能を十分に活用するには Identity Manager の詳細な知識が必要。 • 本稼働環境への投入前に完全調整が必要。
調整	• 調整ポリシーのすべての側面を実装可能 • 事前の調整を使用することで最終段階での不測の事態を回避できる	• 読み込むリソースアカウントを自由に選択できない。 • 大規模環境 (従業員が 50,000 名を超える) では全アカウントの読み込みに膨大な時間がかかる可能性がある

表 4-1　データ読み込みプロセスの概要 (続き)

データ読み込みプロセス	長所	短所
ActiveSync	アカウント情報を読み込む手段として ActiveSync を選ぶことは可能なかぎり避けてください。ActiveSync は変更を検出するように設計されており、結果として初回の読み込みは低速です。	

ファイルから読み込み

「ファイルから読み込み」プロセスは、Identity Manager アカウントにアカウント ID、姓、名、電子メールアドレスなどの基本的な値を割り当てます。アカウント ID は唯一の必須属性です。

「ファイルから読み込み」プロセスでは、CSV (Comma-Separated Value) ファイルの内容を Identity Manager にインポートします。このファイルの先頭行には、属性名のリストをコンマ区切りで記述します。それ以降の行には、対応する一連の属性値を記述します。すべての属性はコンマで区切る必要があります。

注　「ファイルから読み込み」では XML ファイルからの読み込みも可能ですが、XML ファイルの構文が、「ファイルへ抽出」機能によって生成されるファイルの構文と一致している必要があります。この形式について、ここでは詳しく説明しません。

CSV ファイル内のデータは多くの場合、リソースからエクスポートされたものです。たとえば、Microsoft 管理コンソール (MMC) の「Active Directory ユーザーとコンピュータ」では、組織単位の内容を CSV ファイルに直接エクスポートすることができます。組織単位に定義されているすべてのユーザーが、表示されている属性とともにエクスポートされます。したがって、「Active Directory ユーザーとコンピュータ」MMC でエクスポートを実行するときは、Identity Manager によって管理される属性だけが表示されていることを確認してください。不必要な属性が含まれていると、読み込みにかかる時間が長くなります。

リソースの中には、ユーザーアカウント情報を CSV 形式に直接エクスポートできないものもあります。そのようなリソースからデータを読み込む場合、別途プログラムを使用して情報を抽出したり、手動でデータを追加したりするなどの手順が必要な場合があります。

たとえば、CSV ファイルの最初の 3 行が次のようになっているとします。

```
accountId,firstname,lastname,EmployeeID

Josie.Smith,Josie,Smith,1436

AJ.Harris,Anthony,Harris,c310
```

CSV ファイルに記述されている属性が、ユーザービューの属性としてあらかじめ定義されている必要があります。accountId、email、password、confirmpassword などの基本属性はあらかじめ定義されています。その他の属性は、拡張ユーザー属性設定オブジェクトで定義されます。このオブジェクトはデフォルトで、firstname、lastname、fullname を利用可能な属性のリストに追加します。

Identity Manager の定義済みでない従業員 ID などの属性値を保持したい場合、拡張ユーザー属性設定オブジェクトにそれらの属性を追加する必要があります。

「ファイルから読み込み」プロセスの設定ページでは、相関規則と確認規則の指定を求められます。これが最初のデータ読み込みであるため、「**アカウント ID と一致するユーザー名**」相関規則を選択します。確認規則は不要です。

重要な点として、CSV ファイルに含まれているデータは Identity Manager アカウントを構成する目的のみに使用されます。たとえば、データが Active Directory から直接エクスポートされた場合でも、そのデータはどの Active Directory アカウントまたは Active Directory リソースにもリンクされません (リンクを行うことを目的としたカスタムユーザーフォームを作成した場合を除く)。カスタムユーザーフォームがない場合、リソースのアカウントデータを Identity Manager のユーザーとリンクするには、別のデータ読み込み機構を使用する必要があります。ユーザーフォームについては「ユーザーフォームを割り当てる」で簡単に説明します。

注 「ファイルから読み込み」プロセスでは、Identity Manager アカウントインデックスにエントリが追加されません。したがって、Identity Manager の配備が完了する前に、完全調整またはユーザーの更新を実行する必要があります。また、「ファイルから読み込み」では、Identity Manager でのユーザー作成時にワークフローは一切実行されません。

リソースから読み込み

「リソースから読み込み」プロセスと「ファイルから読み込み」プロセスの設定ページはほとんど同じ内容ですが、「リソースから読み込み」は機能的には調整に似ています。「リソースから読み込み」プロセスと調整プロセスでは、データをリソースから抽出したあとで、見つかったアカウントを Identity Manager に追加します。したがって、これらの処理を実行する前にアダプタを設定する必要があります。

「リソースから読み込み」は調整と比べて初回の実行が高速ですが、アカウントインデックスに値が設定されません。差分モードでの 2 回目の実行時の調整プロセスは、「リソースから読み込み」よりも高速です。調整が長期的なソリューションとして望ましいツールである場合は、ユーザーの初回読み込みに調整を使用してください。

ユーザーフォームは、アカウント ID の設定、組織へのユーザーの配置、ユーザー作成に関連するその他のタスクの実行に使用できます。詳細については、「ユーザーフォームを割り当てる」を参照してください。

最初に読み込むリソースに対して、「リソースから読み込み」を使用して Identity Manager アカウントに属性値を割り当てるとき、相関規則と確認規則はそれほど重要な意味を持ちません。「**アカウント ID と一致するユーザー名**」相関規則を選択します。確認規則は不要です。

注	「リソースから読み込み」プロセスでは、Identity Manager アカウントインデックスにエントリが追加されません。したがって、Identity Manager の配備が完了する前に、完全調整またはユーザーの更新を実行する必要があります。また、「ファイルから読み込み」では、Identity Manager でのユーザー作成時にワークフローは一切実行されません。

一括処理の作成

一括処理の作成では、CSV ファイルからデータを読み込みます。一括処理の作成では「ファイルから読み込み」プロセスと異なり、Identity Manager に固有の属性、グローバル属性、リソースアカウント属性などを含む、任意の書き込み可能な属性をユーザービューで定義することができます。この柔軟性の代償として、一括処理用の CSV ファイルの作成は多くの場合、多少手順が複雑になります。複数のリソースを一括処理で処理する場合、リソースデータを単一の CSV ファイルにマージする方法を探す必要があります。ただし、一括処理用の比較的単純なファイルを定義し、あとで更新処理を使用してデータを Identity Manager ユーザーアカウントに読み込むという方法もあります。

一括処理では、作成、更新、削除の各処理に対してデフォルトのワークフローを実行します。これによりユーザーアカウントの読み込み処理の速度は低下しますが、柔軟性が大きく向上します。

次の例では、一括処理の作成のみを使用します。command 属性と user 属性は必須です。

```
command,user,waveset.resources,password.password,password.confir
mPassword,accounts[MyAD].description,accounts[MySolaris].comment

Create,John Doe,MyAD|MySolaris,changeit,changeit,John Doe,John
Doe
Create,Jane Smith,MyAD,changeit,changeit,Jane Smith
```

次の例は、2 つの個別のファイルで一括処理の作成と一括更新処理を使用する方法を示しています。

```
command,user,waveset.resources,password.password,password.confir
mPassword,accounts[MyAD].description,
Create,John Doe,MyAD,changeit,changeit,John Doe,John Doe
Create,Jane Smith,MyAD,changeit,changeit,
Jane Smith

command,user,waveset.resources,password.password,password.confir
mPassword,accounts[MySolaris].comment
Update,John Doe,MySolaris,changeit,changeit,John Doe
Update,Jane Smith,MySolaris,changeit,changeit,Jane Smith
```

注 一括処理を使用してアカウントを作成しても、Identity Manager のアカウントインデックスにエントリは追加されません。したがって、Identity Manager の配備が完了する前に完全調整を実行する必要があります。

調整

多くの場合、リソースの初回の調整には 2 回目以降の調整よりも長い時間がかかります。一般的に、リソースの初回の調整では多数のアカウントインデックスエントリが追加されます。

データ読み込みの準備を行う

Identity Manager へのアカウント情報の読み込みプロセスを開始する前に、次の各項の内容を確認してください。

- アダプタを設定する
- アカウント ID ポリシーとパスワードポリシーを設定する
- データ読み込み用のアカウントを作成する
- フォームを割り当てる

アダプタを設定する

リソース上のアカウントを管理するには、アカウント情報のソースごとにアダプタを設定する必要があります。「ファイルから読み込み」プロセスまたは一括処理を使用する場合、アダプタの設定は調整の準備ができた時点で行えば問題ありません。それ以外の場合は、Identity Manager にデータを読み込む前にアダプタを設定する必要があります。

アダプタの設定についての全般的な情報は、『Identity Manager 管理ガイド』を参照してください。個別のアダプタに関する詳しい情報は、『Identity Manager リソースリファレンス』またはオンラインヘルプを参照してください。

アカウント ID ポリシーとパスワードポリシーを設定する

「リソースから読み込み」、調整、または Active Sync によってリソースからアカウントデータを読み込むとき、Identity Manager はリソースからパスワードを取得しません。リソースからパスワードが渡された場合は、リソース上のセキュリティーが侵害される可能性があります。したがって、Identity Manager アカウントのパスワードは、リソース上のパスワードと同じにはなりません。デフォルトでは、Identity Manager によってランダムのパスワードが生成され、このパスワードを再設定する必要があります。ただし、ユーザーフォームのパスワードビューを使用して、全ユーザーに共通の、または Identity Manager アカウント ID と同一のリテラル文字列などを一時パスワードとして指定することもできます。詳細については、「ユーザーフォームを割り当てる」と、「Identity Manager のビュー」の章を参照してください。

一括処理および「ファイルから読み込み」では、パスワード値を CSV ファイルで指定できます。これらのパスワードは、ユーザーが変更しなければならない一時パスワードとして考えるべきです。

ポリシーは Identity Manager アカウントに適用される制限を設定し、次のように分類されます。

- Identity Manager アカウントポリシー -- ユーザー、パスワード、および認証のポリシーオプションを設定するために使用します。Identity Manager のアカウントポリシーは組織またはユーザーに割り当てられます。
- リソースパスワードおよびアカウント ID ポリシー -- 長さ規則、文字タイプ規則、使用可能な単語および属性値を設定または選択するために使用します。

デフォルトポリシーを更新する場合は必ず、Identity Manager へのアカウント情報の読み込みを開始する前に行ってください。

次の表は、Identity Manager に用意されているポリシーとそのデフォルト設定の一覧です。

表 4-2 デフォルトの Identity Manager ポリシー

ポリシー名	デフォルト特性
アカウント ID ポリシー	アカウント ID の長さは 4 文字以上 16 文字以下である必要があります。
デフォルトの Lighthouse アカウントポリシー	アカウント ID ポリシーを「**アカウント ID ポリシー**」に、パスワードポリシーを「**パスワードポリシー**」に設定します。パスワードはユーザーではなく Identity Manager によって生成されます。
パスワードポリシー	パスワードの長さは 4 文字以上 16 文字以下である必要があります。パスワードにはユーザーの電子メールアドレス、姓、名、またはフルネームを含めることはできません。
Windows 2000 パスワードポリシー	パスワードの長さは 6 文字以上である必要があります。パスワードには次の文字タイプのうち 3 種類以上が含まれている必要があります。 • 数字 (1) • 大文字のアルファベット (1) • 小文字のアルファベット (1) • 特殊文字 (1) また、パスワードにはアカウント ID を含めることはできません。

アカウントポリシーとパスワードポリシーの詳細については、『Identity Manager 管理ガイド』の「Identity Manager ユーザー」の章を参照してください。

データ読み込み用のアカウントを作成する

次の理由により、データ読み込みを実行するための管理者アカウントを別途作成することをお勧めします。

- 監査を有効にしたときにデータ読み込み処理の追跡が容易になります。
- すべての Identity Manager 管理者に、Identity Manager ユーザーを作成および編集するユーザーフォームが割り当てられます。データ読み込み用のアカウントを作成すると、デフォルトのフォームよりも実行が高速な簡易形式のフォームを指定できます。詳細については、次の項を参照してください。

アカウントの作成の詳細については、『Identity Manager 管理ガイド』を参照してください。

ユーザーフォームを割り当てる

データ読み込みのコンテキストでは、ユーザーフォームはバックグラウンド処理を実行する目的で使用されます。たとえば、フォームとリソースアダプタを一緒に使用して、外部リソースから得た情報を処理し、Identity Manager リポジトリにこれを保存できます。ユーザーフォームは、入力ユーザーデータに基づいて正しい組織にユーザーを配置する目的にも使用できます。

ユーザービューは、Identity Manager ユーザーの利用可能な全情報が含まれるデータ構造です。このビューには、以下の項目が含まれます。

- Identity Manager リポジトリに格納される属性
- リソースアカウントからフェッチされる属性
- リソース、ロール、および組織など、ほかのソースから取得される情報

ビューには多くの属性があり、ビュー属性はビュー内にある名前を付けられた値です (たとえば、`waveset.accountId` は、Identity Manager アカウント名を値とする、ユーザービュー中の属性)。

ほとんどのフォームフィールドの名前は、ビュー属性と関連付けられています。フォームフィールドの名前として、ビュー属性の名前を指定することで、フィールドとビュー属性を関連付けます。ユーザービューの全属性の参照情報など、ユーザービューの詳細については、「Identity Manager のビュー」の章を参照してください。

ユーザーを読み込むユーザーフォームには、多くの場合次のフィールドが含まれます。

- firstname (名)
- lastname (姓)
- fullname (フルネーム)
- email (電子メール)
- waveset.accountId
- waveset.organization
- EmployeeId (従業員 ID)

`waveset.accountId` および `waveset.organization` は、Identity Manager に固有の値です。`EmployeeId` 属性はカスタマイズした属性です。その用途については、「カスタム相関キーを定義する」で例示しています。

Identity Manager には、システムにインストール済みの多数のフォームが用意されています。また、追加のフォームが `$WSHOME/sample/forms` ディレクトリに収められています。このディレクトリ内のフォームの多くは、リソース固有の内容になっています。Identity Manager IDE を使用してこれらのフォームを確認し、本稼働環境で使用するかどうかを判断することができます。

一括処理中のパフォーマンスを向上させるために、管理者に割り当てられるユーザーフォームはできるだけ簡易なものにしてください。データ読み込み用のフォームを作成する場合、データ表示のために記述されたコードは削除できます。フォーム簡易化の別の例は、一括追加処理を使用する場合です。CSV ファイルでは、`firstname` や `lastname` などの基本属性を定義できます。これらの属性は、あとで管理者のユーザーフォームから削除できます。フォームの作成と編集の詳細については、「Identity Manager のフォーム」の章を参照してください。

注 Identity Manager に用意されているフォームを直接修正しないでください。代わりに、これらのフォームのコピーを作成して一意の名前を付け、このファイル名を変更したコピーフォームを編集するようにしてください。こうすることにより、カスタマイズしたコピーがアップグレードやサービスパックの更新の際に上書きされるのを防止できます。

ほかのリソースへアカウントをリンクする

Identity Manager では、相関規則と確認規則を使用してアカウントどうしをリンクします。相関規則は、アカウントを所有している可能性がある Identity Manager ユーザーを検索します。相関規則で定義された条件に一致するユーザーのリストが返されます。確認規則は、ユーザーが実際にアカウントを所有しているかどうかを確認するために、アカウントに対して Identity Manager ユーザーをテストします。true または false の値が返されます。所有者候補をすばやく検索したり (名前または属性に基づいて)、負担の大きなチェックの実行を所有者候補だけに制限したりするので、この 2 段階のアプローチによって、Identity Manager は相関を最適化することができます。

相関規則と確認規則の使用を開始する前に、最初のデータ読み込みの時点から含まれるデータを確認しておく必要があります。Identity Manager の `accountId` 属性は常に含まれます。「ファイルから読み込み」または一括処理の作成を実行した場合、CSV ファイルの先頭行に記述された属性値も含まれます。「リソースから読み込み」または調整を実行した場合、リソース上の一部のキー属性は含まれますが、ほかの属性は明示的に保存された場合にのみ含まれます。

また、二次リソース上に格納されたアカウントデータについても確認しておく必要があります。二次リソースには、Identity Manager にすでに存在するデータと重複するデータが含まれているのが理想的です。

これは意外に難しい条件である場合があります。多くの場合、異なるリソース間ではユーザーアカウントについての要件も異なります。例として、次の表では Windows のアカウント名と Solaris のアカウント名について要件および制限を比較しています。

表 4-3 Windows アカウント名と Solaris アカウント名の要件の比較

特性	Windows	Solaris
最大長	20 文字	8 文字
使用できる特殊文字	" / ¥ [ ] : ; | = , + * ? < > を除くすべて	ピリオド (.)、下線 (_)、ハイフン (-) のみ

表 4-3　Windows アカウント名と Solaris アカウント名の要件の比較 (続き)

特性	Windows	Solaris
フルネームを指定可能	あり	コメントフィールドが 1 つあり。このコメントフィールドには伝統的にユーザーのフルネームを記述することになっていますが、その他の情報が含まれている可能性があります。
従業員 ID などの追加コメントを指定できるか	あり	

Windows と Solaris のアカウント名の違いから、アカウントのリンクに伴ういくつかの問題が明らかになります。

- 最大長の違いが理由で、これら 2 つのシステムではアカウント名が異なるのが一般的です。Windows では、ユーザーが自分の姓および名と一致するアカウント名を持つことがよくあります。Solaris では、名の先頭の文字と、名の先頭から 7 文字を組み合わせたものをアカウント名として使用する場合があります。したがって、アカウント ID を比較する相関規則は、Solaris アカウントを Windows アカウントとリンクする手段として確実ではありません。
- Windows でユーザーアカウントを作成するとき、管理者は表示名を指定する必要があります。Solaris のユーザーアカウントには表示名は不要ですが、オプションの GECOS フィールドを使用してユーザーの名前を指定できます。このフィールドの内容または形式についてのガイドラインはありません。ユーザーの GECOS フィールドは空にすることも、Windows の表示名とは関係のないテキストを入れることもできます。したがって、フルネームを比較する相関規則は、一般的に考えられているほど多くは利用されません。

アカウントをリンクするための準備をするときは、次の質問の答えを検討してください。

- Mary A. Jones と Mary B. Jones、または John Doe Jr. と John Doe III などの、よく似た名前を持つユーザーがいますか。答えが「はい」の場合、それらのユーザーをどのようにして区別しますか。
- 各リソース上でアカウント名を定義する方法は共通化されていましたか。答えが「はい」の場合、Identity Manager に格納される値を使用してリソース上のアカウント ID を比較する規則を容易に定義できます。
- リソースのアカウントデータを編集する機能がユーザーに許可されていましたか。答えが「はい」の場合、そのデータは、ユーザーが変更できないシステム上に格納される値と一致しない可能性があります。

カスタム相関キーを定義する

規則では、値がシステムに格納されないかぎり、リソース上のアカウントの値を Identity Manager の値と比較することはできません。accounts[Lighthouse] 属性にはこれらの値の多くが格納されますが、それら以外の値は、拡張ユーザー属性設定オブジェクトを使用して追加する必要があります。設定オブジェクトに登録されていない属性はシステムに保存されません。

デフォルトでは、次の属性が拡張ユーザー属性として含まれています。

- firstname（名）
- lastname（姓）
- fullname（フルネーム）

注 拡張ユーザー属性 fullname は、QueryableAttrNames のリストに追加する必要があります。

これまでに述べた以外の、従業員 ID などの属性を相関規則の一部として使用したい場合、その属性を拡張ユーザー属性設定オブジェクトに追加する必要があります。追加は次の手順で行います。

1. Identity Manager のデバッグページ (http://PathToIDM/debug) にアクセスします。「System Settings」ページが表示されます。
2. 「**List Objects**」プルダウンメニューから「Configuration」を選択します。「**List Objects of type: Configuration**」ページが表示されます。
3. 「**User Extended Attributes**」の「**edit**」リンクを選択します。
4. 次の例のように、List 要素に新しい属性を追加します。

   ```
   <String>EmployeeId</String>
   ```

5. リソースの「アカウント属性 (スキーママップ)」ページで、属性を Identity Manager 属性として定義する必要があります。
6. 変更を保存します。Identity Manager の「System Settings」デバッグページに戻ります。

カスタム属性は、UserUIConfig 設定オブジェクトの QueryableAttrNames 要素にも追加する必要があります。

1. 「**List Objects**」プルダウンメニューから「Configuration」を選択します。「**List Objects of type: Configuration**」ページが表示されます。
2. 「**UserUIConfig**」の「**edit**」リンクを選択します。
3. 次の例のように、<QueryableAttrNames><List> 要素に新しい属性を追加します。

```
<String>EmployeeId</String>
```

4. 変更を保存します。Identity Manager の「System Settings」デバッグページに戻ります。
5. アプリケーションサーバーを再起動します。

カスタム規則を作成する

Identity Manager は、いくつかの相関規則と確認規則を sample/reconRules.xml 内で事前に定義しています。これらは、独自の規則を作成するためのベースとして使用できます。規則には SUBTYPE_ACCOUNT_CORRELATION_RULE または SUBTYPE_ACCOUNT_CONFIRMATION_RULE のサブタイプを割り当てる必要があります。

次の規則は、二次リソースで定義された属性である account.EmployeeId を、以前に Identity Manager に読み込まれた属性である EmployeeId と比較します。二次リソースに account.EmployeeId の値が存在する場合、EmployeeId と一致するユーザーのリストが相関規則によって返されます。

```
<Rule subtype='SUBTYPE_ACCOUNT_CORRELATION_RULE' name='Correlate
Employee IDs'
   <cond>
      <ref>account.EmployeeId</ref>
      <list>
         <new class='com.waveset.object.AttributeCondition'>
            <s>EmployeeId</s>
            <s>equals</s>
            <ref>account.EmployeeId</ref>
         </new>
      </list>
   </cond>
</Rule>
```

この例では、EmployeeId 属性はあらかじめユーザー拡張属性設定オブジェクトおよび UserUIConfig 設定オブジェクトに追加されています。リソースの側でデフォルトの Identity Manager 属性名にこの属性が含まれていなかった場合、リソースのスキーママップでこの属性を追加または編集する必要もあります。

相関規則は見つかった一致のリストを返します。従業員 ID のように、結果で 1 つの一致しか返されないことが予想される場合、確認規則は必要ありません。一方で、複数の一致が考えられる場合 (たとえば、姓名が一致する複数のアカウントが相関規則で検出された場合など) は、一致をさらに検証するための確認規則が必要です。

規則を Identity Manager に追加する方法には、Identity Manager IDE を使用する方法、XML ファイルをインポートする方法、デバッグページで既存の規則を編集してその名前を変更する方法があります。

手動でアカウントをリンクする

Identity Manager には、相関規則および確認規則で一致が見つからないときにアカウントを割り当てるために使用できる複数の機構が用意されています。

アカウントインデックスを使用する

アカウントインデックスには、Identity Manager が認識している各リソースアカウントの最新の状態が記録されています。このインデックスは主に調整によって管理されますが、Identity Manager のほかの機能でも必要に応じてアカウントインデックスの更新が行われます。

注 「リソースから読み込み」、「ファイルから読み込み」、および一括処理では、アカウントインデックスの更新は行われません。

アカウントインデックスを確認するには、「リソース」タブをクリックし、左側の「アカウントインデックス」リンクをクリックします。リソースへ移動すると、対象のリソース上のすべてのアカウントの状態が表示されます。

相関のないアカウント (「アカウントインデックス」テーブルで、状況が「UNMATCHED」、所有者が「_UNKNOWN_」として表示される) を右クリックすると、Identity Manager のメニューが表示されます。このメニューから、新しい Identity Manager ユーザーアカウントの作成、リソースに対して有効な調整ポリシーを使用した単一アカウントに対する調整の実行、所有者の指定、リソースアカウントの削除または無効化などの処理を実行できます。「所有者の指定」オプションを選択すると、Identity Manager は、指定された条件に一致する所有者を検索するための画面を表示します。詳細については、『Identity Manager 管理ガイド』を参照してください。

自己検索を有効にする

Identity Manager のユーザーが自分自身のリソースアカウントの検索を行えるように、Identity Manager のユーザーインタフェースを設定することができます。つまり、Identity Manager の ID を持つユーザーはこの機能によって、既存のまだ関連付けられていないリソースアカウントに自分の ID を関連付けることができます。自己検索を有効にすることができるリソースは、パススルー認証をサポートするリソースに限られます。

自己検索を有効にするには、エンドユーザーリソース設定オブジェクトを編集して、ユーザーによるアカウント検索を許可するリソースの名前をそれぞれこのオブジェクトに追加する必要があります。次の手順で実行します。

1. Identity Manager のデバッグページ (`http://PathToIDM/debug`) にアクセスします。「System Settings」ページが表示されます。
2. 「**List Objects**」プルダウンメニューから「Configuration」を選択します。「**List Objects of type: Configuration**」ページが表示されます。
3. 「**End User Resources**」の「**edit**」リンクを選択します。
4. `<List>` 要素のあとに、`<String>Resource</String>` を追加します。ここで、`Resource` はリポジトリ内のリソースオブジェクトの名前です。たとえば、ユーザーがリソース `AD` および `Solaris` 上のアカウントを自己検索できるようにするには、`<List>` 要素を次のように編集します。

```
<List>
   <String>AD</String>
   <String>Solaris</String>
</List>
```

5. 変更を保存します。Identity Manager の「System Settings」デバッグページに戻ります。

自己検索を有効にすると、Identity Manager のユーザーインタフェースに新しいメニュー項目 (「**他のアカウントについてアイデンティティーシステムに知らせる**」) が追加されます。ユーザーはこの領域で、利用可能なリストからリソースを選択し、リソースアカウント ID とパスワードを入力して、アカウントを自分の Identity Manager ID とリンクすることができます。

シナリオの例

この項では、1 つまたは複数のリソースからアカウントを読み込むプロセスを例示するシナリオを提供します。次のシナリオでは、さまざまな環境で発生する可能性がある問題について検討します。

- Active Directory、SecurID、および Solaris
- LDAP、PeopleSoft、および Remedy
- 高速一括追加

Active Directory、SecurID、および Solaris

ある会社で、Identity Manager を使用して Active Directory、SecurID、および Solaris のアカウントを管理することを検討しています。すべての従業員が Active Directory アカウントを、ほとんどの従業員が SecurID アカウントを持っています。ごく一部の従業員が Solaris アカウントを持っています。各リソース上のアカウントデータを検証したあとで、Identity Manager 管理者は、相関キーとして次の属性を使用できると判断しました。

表 4-4 相関キーの候補

相関キーの候補	Active Directory	SecurID	Solaris
アカウント ID が AD と一致	N/A	あり	なし
従業員 ID	あり	なし	なし
フルネーム	あり	あり	使用可 (Description 属性)

すべての従業員が Active Directory アカウントを持っているため、最初のデータ読み込みリソースには Active Directory を使用します。SecurID リソース上のアカウント ID は Active Directory 上のアカウント ID と一致するため、SecurID リソースを 2 番目に読み込みます。アカウント ID は常に一意であることから、フルネームよりも相関キーとして適しています。Active Directory アカウントおよび SecurID アカウントの間には、問題なく相関が成立すると考えられます。

Solaris アカウントについては、相関の成立は難しくなります。Solaris アカウントに存在する属性のうち、唯一の相関属性はユーザーのフルネームです。Solaris アカウントには、姓と名を定義するための独立した属性はありません。結果として、相関規則では、Solaris の `useradd -c` コマンドで定義される文字列と、Active Directory の `fullname` の値を比較することになります。この比較は、ニックネームの使用や不必要なスペースまたは句読文字が原因で、よく失敗します。

ユーザーの例

このシナリオでは、次のユーザーが存在する環境で、アカウントの読み込み時に発生する可能性があるいくつかの問題の例を示します。

表 4-5 データ読み込みのシナリオ : アカウントの読み込み時に発生する可能性のある問題

従業員名	AD および SecurID ログオン名	AD フルネーム	Solaris アカウント名	Solaris の Description
Anthony Harris	AJ Harris	Anthony J Harris	ajharris	A.J. Harris
Isabelle Moreno	Isabelle Moreno	Isabelle Moreno	imoreno	Isabelle Moreno
John Thomas (Sr.)	John Thomas	John Thomas	jthomas	John Thomas
John Thomas (Jr.)	John P. Thomas	John P. Thomas	jthomas2	John Thomas
Robert Blinn	Robert Blinn	Bob Blinn	rblinn	Bob Blinn
Theodore Benjamin	Theodore Benjamin	Theodore Benjamin	tbenjami	Ted Benjamin

Active Directory アカウントを読み込む

調整を使用して Active Directory アカウントを Identity Manager に読み込むためのガイドラインとして、次の手順を使用します。

1. 管理者インタフェースの「リソース」ページで、「新規リソース」プルダウンメニューから「Windows 2000 / Active Directory」リソースを選択します。次に、アダプタを設定します。

   Identity Manager のユーザー属性 `accountId` または `fullname` をスキーママップから削除しないように注意してください。また、アイデンティティーテンプレートが正しいことを確認してください。アダプタの設定の詳細については、オンラインヘルプおよび『Identity Manager リソースリファレンス』を参照してください。

2. (オプション) アカウントポリシーとパスワードポリシーを必要に応じて変更します。詳細については、「アカウント ID ポリシーとパスワードポリシーを設定する」を参照してください。

3. (オプション) 調整に使用するユーザーフォームを作成します。詳細については、「ユーザーフォームを割り当てる」を参照してください。

4. (オプション) データ読み込みを実行するための Identity Manager ユーザーを作成します。前の手順で作成したユーザーフォームをユーザーに割り当てます。

5. リソースの調整ポリシーを設定します。最初のリソースについては、相関規則は重要ではなく、確認規則は Identity Manager ユーザーの作成時には使用されません。これは最初のリソースであるため、UNMATCHED 状況に対する応答オプションには「リソースアカウントに基づく新規ユーザーの作成」を割り当てるのが適切と考えられます。
6. データ読み込みを実行するためのユーザーを作成したら、そのユーザーとしてログインします。調整の場合はこの手順は不要ですが、「ファイルから読み込み」、「リソースから読み込み」、または一括処理の場合は必要です。
7. Active Directory リソースを調整します。

結果

デフォルトの Identity Manager アカウントポリシーとデフォルトの Active Directory アイデンティティーテンプレートを使用した場合、Theodore Benjamin の名前が 16 文字を超えているため、Identity Manager はこのユーザーの Active Directory アカウントにリンクする Identity Manager ユーザーを作成しません。これを回避するために、この例では、アカウント ID ポリシーを 25 文字に設定しました。

Identity Manager は、状況が CONFIRMED 状態であるすべてのリソースアカウントに対してユーザーアカウントを作成します。パスワードポリシーとアカウント ID ポリシーをパスしたすべてのユーザーがこれに該当します。ユーザーフォームでほかの設定を指定した場合を除き、Identity Manager のアカウント名は Active Directory のログイン名と同じになります。

SecurID アカウントを読み込む

SecurID を実装するとき、SecurID のユーザーレコードは通常、Microsoft セキュリティーアカウントマネージャー (SAM) データベースまたは LDAP サーバーからインポートされます。その結果、SecurID のアカウント ID はインポートソースのアカウント ID と一致します。SecurID アカウントと Active Directory アカウントの間には 1 対 1 の相関が成立するため、ユーザー間の相関確立は比較的容易になります。これらのアカウントを迅速にリンクする目的で、「アカウント ID と一致するユーザー名」の相関規則を使用できます。

SecurID アカウントを読み込むには、「Active Directory アカウントを読み込む」で説明した手順を、次のように変更して実行します。

- SecurID アダプタを設定しているときは、Identity Manager のユーザー属性である `accountId` を削除しないように注意してください。
- 調整ポリシーを次のように設定します。
    - 相関規則を「アカウント ID と一致するユーザー名」に設定します。

- Active Directory はアイデンティティー情報の源泉として信頼性の高いリソースと考えられ、SecurID は Active Directory のアカウント情報に依存するため、UNMATCHED 状況時の動作オプションを「リソースアカウントの削除」または「リソースアカウントの無効化」に設定することをお勧めします。UNASSIGNED 状況は「Identity Manager ユーザーへリソースアカウントをリンク」に設定します。

結果

すべての SecurID アカウントについて、Active Directory アカウントとの相関が確立されます。必要に応じて、UNMATCHED または DISPUTED 状況を解決するための手順を追加します。

Solaris アカウントを読み込む

このシナリオでは、`fullname` 属性が唯一の相関キーです。スペースや句読文字などの違いにより確実に一致が失敗するため、これは弱い相関キーです。また、ユーザーは自分の表示名を Solaris の `chfn` コマンドで変更できます。フルネームが当初一致していたとしても、ユーザーが `chfn` コマンドを実行したことが原因で相関を確立できない可能性があります。

デフォルトでは、`fullname` 属性はクエリー可能ではありません。クエリー可能にするには、`UserUIConfig` 設定オブジェクトを編集し、`fullname` 属性を `<QueryableAttrNames><List>` 要素に追加する必要があります。詳細については、「カスタム相関キーを定義する」を参照してください。

`fullname` 属性どうしを相関させるためのカスタム規則を作成する必要もあります。この例では、「Correlate Full Names」という名前の規則を作成して相関処理を実行させます。この規則では、Solaris リソースの `account.Description` 属性の値を、Active Directory から読み込まれたシステム属性である `fullname` 属性と比較します。

```
<Rule subtype='SUBTYPE_ACCOUNT_CORRELATION_RULE' name='Correlate
Full Names'
   <cond>
      <ref>account.Description</ref>
      <list>
         <new class='com.waveset.object.AttributeCondition'>
            <s>fullname</s>
            <s>equals</s>
            <ref>account.Description</ref>
         </new>
      </list>
   </cond>
</Rule>
```

この規則では、Solaris リソースからの Description 属性を、Identity Manager の `fullname` 属性と比較します。2 つの属性が一致する場合、アカウント間に相関が成立し、状況が CONFIRMED に変化します。

Solaris アカウントを読み込むには、「Active Directory アカウントを読み込む」で説明した手順を、次のように変更して実行します。

- Solaris アダプタを設定しているときは、Identity Manager のユーザー属性である `accountId` または Description を削除しないように注意してください。
- 調整ポリシーを次のように設定します。
  - 相関規則を「Correlate Full Names」(サンプル規則) に設定します。
  - すでに Identity Manager に読み込まれているアカウントと相関のない Solaris アカウントが数多く存在する可能性があります。UNASSIGNED 状況時の動作オプションを「リソースアカウントをユーザーにリンク」設定します。UNMATCHED 状況はほとんどの場合、「何もしない」に設定するのが適切です。一致しないユーザーを削除または無効化すると、データの消失や生産性の低下を招く可能性があります。

結果

表 4-6 データ読み込みのシナリオのユーザー例

従業員名	AD フルネーム	Solaris アカウント名	Solaris の Description
Anthony Harris	Anthony J Harris	ajharris	A.J. Harris
Isabelle Moreno	Isabelle Moreno	imoreno	Isabelle Moreno
John Thomas (Sr.)	John Thomas	jthoma	John Thomas
John Thomas (Jr.)	John P. Thomas	jthomas2	John Thomas
Robert Blinn	Bob Blinn	rblinn	Bob Blinn
Theodore Benjamin	Theodore Benjamin	tbenjami	Ted Benjamin

この例では、Isabelle Moreno のアカウントのみに相関が成立することが予測されます。

- Active Directory の `fullname` 属性は Solaris の Description 属性と完全には一致しないため、Anthony Harris、John Thomas (Jr.)、Robert Blinn、および Theodore Benjamin のアカウントについては相関が成立しません。これらのアカウントの状況は UNMATCHED となります。このシナリオでは、Solaris のアカウント名は名のイニシャルと姓の組み合わせで構成されます。John Thomas のアカウントの例外を除き、これら不一致の Solaris アカウントの割り当ては簡単です。

- Solaris アカウント jthomas および jthomas2 は、DISPUTED の状況となります。これらのアカウントは Description の値がともに John Thomas です。Solaris アカウント jthomas および jthomas2 がどのユーザーに属するのかを決定する別の手段を見つける必要があります。確認規則を使用して 2 つのアカウントを区別できれば理想的です。しかしながら、Solaris アカウントと Active Directory アカウントには、確認規則を作成するための十分な共通属性が含まれていません。

LDAP、PeopleSoft、および Remedy

このシナリオでは、理論的に LDAP リソースまたは PeopleSoft リソースが一次リソースとなりえます。

- すべての従業員および契約社員が PeopleSoft で追跡される場合、このアプリケーションをアイデンティティー情報の源泉として信頼性の高いリソースとして考えることができます。
- すべての従業員が LDAP アカウントを持つ場合、LDAP をアイデンティティー情報の源泉として信頼性の高いリソースとして考えることができます。

Remedy アカウントを持つ従業員の割合はごく限られているため、Remedy は一次リソースの候補とはなりません。

複数のアイデンティティー情報の源泉として信頼性の高いリソースが存在する場合、それらのうちどのリソースを最初に読み込んでもかまわないケースがほとんどです。ただし、PeopleSoft コンポーネントアダプタは ActiveSync 機能のみを実行します。別の PeopleSoft アダプタも利用できますが、範囲は限定されています。PeopleSoft コンポーネントアダプタは調整を実行しないため、結果的に、リソースに対して調整ポリシーを設定できません。利用可能な相関規則がないため、PeopleSoft アカウントを最初に読み込む必要があります。Identity Manager アカウント名は、PeopleSoft の EMPLID (従業員 ID) の値と一致します。

PeopleSoft の従業員 ID は、システムで定義されるすべてのユーザーについて一意であるため、相関キーとして理想的です。LDAP の inetOrgPerson オブジェクトには employeeNumber 属性が含まれます。従業員 ID は、Description などのラベルが付いた属性、またはカスタム属性に格納することもできます。このシナリオでは、LDAP の employeeNumber 属性が使用されていると想定します。

Remedy アダプタにはデフォルト属性が用意されていません。環境に合わせてアダプタをカスタマイズする必要があります。Remedy アプリケーションは、システムが要求を受け取ったときに電子メールを送信するように設定されることが多いため、電子メール属性が利用可能であると想定されます。LDAP の inetOrgPerson オブジェクトにもメール属性が含まれます。したがって、電子メールアドレスが相関キーとなります。

次の表は、このシナリオにおける各リソースの相関キーの一覧です。

表 4-7　データ読み込みのシナリオ : 各リソースの相関キー

相関キーの候補	PeopleSoft	LDAP	Remedy
従業員 ID	あり	あり	なし
電子メールアドレス	なし	あり	あり

ユーザーの例

このシナリオでは、次のユーザーが存在する環境で、アカウントの読み込み時に発生する可能性があるいくつかの問題の例を示します。

表 4-8　配備シナリオ : アカウントの読み込み時に発生する可能性のある問題

従業員名	PeopleSoft EMPLI	LDAP 従業員番号	LDAP 電子メール (@example.com)	Remedy 電子メール (@example.com)
Robert Blinn	945	945	Bob.Blinn	bblinn
William Cady	なし	なし	William.Cady	William.Cady
Eric D'Angelo	1096	1096	Eric.D'Angelo	Eric.D'Angelo
Renée LeBec	891	なし	なし	なし
Josie Smith	1436	1463	Josie.Smith	なし
John Thomas	509	509	John.Thomas	なし
John P. Thomas	なし	なし	John.P.Thomas	John.P.Thomas

PeopleSoft ユーザーを読み込む

ActiveSync を使用する PeopleSoft アカウントを、調整を使用して Identity Manager に読み込むためのガイドラインとして、次の手順を使用します。

1. Identity Manager 管理者インタフェースの「リソース」ページで、「新規リソース」プルダウンメニューから「PeopleSoft コンポーネント」リソースを選択します。このリソースが表示されない場合、「**管理するリソースの設定**」ボタンをクリックし、カスタムリソースとして `com.waveset.adapter.PeopleSoftComponentActiveSyncAdapter` を追加します。このアダプタは、PeopleSoft が提供する JAR ファイルのインストールを必要とします。詳細については、『Identity Manager リソースリファレンス』を参照してください。

2. アダプタを設定します。Identity Manager のユーザー属性 accountId または fullname をスキーママップから削除しないように注意してください。また、アイデンティティーテンプレートが正しいことを確認してください。
3. (オプション) アカウントポリシーとパスワードポリシーを必要に応じて変更します。詳細については、「アカウント ID ポリシーとパスワードポリシーを設定する」を参照してください。
4. (オプション) データの読み込みに使用するユーザーフォームを作成します。$WSHOME/sample/forms/PeopleSoftForm.xml ファイルをベースとして使用できます。詳細については、「ユーザーフォームを割り当てる」を参照してください。
5. PeopleSoft アダプタ上で ActiveSync を開始します。

結果

ユーザーフォームでアカウントを読み込まないよう指示された場合を除き、Identity Manager はすべてのユーザーを読み込みます。このシナリオで、Renée LeBec は LDAP アカウントと Remedy アカウントを持っていません。この人物はすでに会社を退職していると考えられます。デフォルトの PeopleSoft フォームを使用した場合、Identity Manager によって、退職した従業員の PeopleSoft アカウントは無効にされます。

William Cady および John P. Thomas は PeopleSoft システムで定義されていないため、これらのユーザーのアカウントは作成されません。

LDAP ユーザーを読み込む

このシナリオでは、LDAP の inetOrgPerson オブジェクトの employeeNumber 属性が相関キーです。この属性は LDAP アダプタのスキーママップにデフォルトで入っていないため、手動で追加する必要があります。この例の場合、属性 EmployeeId をスキーママップの「Identity Manager ユーザー属性」の側に、employeeNumber を「リソースユーザー属性」の側に追加します。

注 PeopleSoft アダプタは、Identity Manager の属性名 EmployeeId をデフォルトで使用します。LDAP と PeopleSoft の間で一貫性を保つためにこの値を選択しましたが、これは必須ではありません。

電子メールアドレスは Remedy リソースの相関キーとなりますが、LDAP アカウントを読み込む前に設定しておく必要があります。inetOrgPerson オブジェクトには、Remedy アカウントを読み込むための相関キーとなる mail 属性が含まれます。mail 属性もスキーママップに追加する必要があります。email 属性をスキーママップの

「Identity Manager ユーザー属性」の側に、mail 属性を「リソースユーザー属性」の側に追加します。email は事前に定義された Identity Manager 属性であるため、この属性を使用するほうが、ユーザー拡張属性設定オブジェクトまたは UserUIConfig 設定オブジェクトを編集して mail 属性を追加するよりも簡単です。

Identity Manager では、アカウント ID は User オブジェクトの resourceAccountIds 属性に格納されます。これは複数の値をとる属性であり、各値の形式は accountId@objectId です。LDAP の EmployeeId の値を PeopleSoft の accountId と比較する、次のような規則を作成できます。

コード例 4-1 LDAP の EmployeeId 値と PeopleSoft の accountId の比較

```
<Rule subtype='SUBTYPE_ACCOUNT_CORRELATION_RULE' name='Correlate EmployeeId with
      accountId'>
   <cond>
      <ref>account.EmployeeId</ref>
      <list>
         <new class='com.waveset.object.AttributeCondition'>
            <s>resourceAccountIds</s>
            <s>startsWith</s>
            <concat>
               <ref>account.EmployeeId</ref>
               <s>@</s>
            </concat>
         </new>
      </list>
   </cond>
</Rule>
```

注 このシナリオでは、accountId および email 属性はシステム上で常に利用可能であるため、ユーザー拡張属性設定オブジェクトまたは UserUIConfig 設定オブジェクトに属性を追加する必要はありません。

LDAP アカウントを読み込むには、次の手順を実行します。

1. 管理者インタフェースの「リソース」ページで、「新規リソース」プルダウンメニューから「LDAP」リソースを選択します。その後、アダプタを次のように設定します。
   a. Identity Manager のユーザー属性 EmployeeId および email を追加します。

b. Identity Manager のユーザー属性である accountId をスキーママップから削除しないように注意してください。

c. アイデンティティーテンプレートが正しいことを確認します。

アダプタの設定の詳細については、オンラインヘルプおよび『Identity Manager リソースリファレンス』を参照してください。

2. リソースの調整ポリシーを次のように設定します。

a. 相関規則を「Correlate EmployeeId with accountId」に設定します。

b. 次の状況の値を設定します。

UNASSIGNED 状況時の動作オプションを「Identity Manager ユーザーへリソースアカウントをリンク」に設定します。

UNMATCHED 状況のオプションを適切な処理に設定します。ほかのリソース上に検出されたユーザーを追加できるかどうかについては、PeopleSoft 管理者への確認が必要な場合があります。「リソースアカウントに基づく新規ユーザーの作成」オプションを選択する場合、Identity Manager ユーザーにはデフォルトで、LDAP の cn 属性に基づくアカウント名が付与されます。

3. LDAP リソースを調整します。

結果

このシナリオでは、William Cady、Josie Smith、および John P. Thomas のアカウントは UNMATCHED 状態になります。Josie Smith については、従業員 ID の値が PeopleSoft リソース上と LDAP リソース上で一致しません。従業員 ID は PeopleSoft によって生成されるため、LDAP リソース側の値が正しくありません。誤りを訂正し、もう一度調整を行ってください。

William Cady および John P. Thomas は PeopleSoft で定義されていません。LDAP アカウントの読み込み手順の手順 2 の記述に従って、アカウントを PeopleSoft に追加する必要があるかどうかを検討してください。

Remedy ユーザーを読み込む

Remedy アダプタには事前に定義されたアカウント属性はありません。これらの属性をスキーママップに追加する必要があります。Remedy では、追跡する各属性を一意に識別するために整数を使用します。たとえば、Remedy のアカウント ID には 1002000100 のような値が割り当てられる場合があります。これらの Remedy 属性番号を、スキーママップのリソースユーザー属性に追加する必要があります。最低限、次の Identity Manager ユーザー属性を追加する必要があります。

- accountId
- email (相関キー)

USER_EMAIL_MATCHES_ACCOUNT_EMAIL_CORR 相関規則は、Remedy アカウントを Identity Manager アカウントにリンクします。

Remedy アカウントを読み込むには、次の手順を実行します。

1. Identity Manager 管理者インタフェースの「リソース」ページで、「新規リソース」プルダウンメニューから「Remedy」リソースを選択します。その後、アダプタを次のように設定します。

   最低限、Identity Manager のユーザー属性 `accountId` および `email` を追加します。ほかの属性を追加することもできます。

   アダプタの設定の詳細については、オンラインヘルプおよび『Identity Manager リソースリファレンス』を参照してください。

2. リソースの調整ポリシーを次のように設定します。

   a. 相関規則を `USER_EMAIL_MATCHES_ACCOUNT_EMAIL_CORR` に設定します。

   b. 次の状況の値を設定します。

      I. `UNASSIGNED` 状況時の動作オプションを「リソースアカウントをユーザーにリンク」に設定します。

      II. `UNMATCHED` 状況を適切な処理に設定します。

3. Remedy リソースを調整します。

結果

William Cady、Eric D'Angelo、および John P. Thomas の Remedy アカウントについては、LDAP と Remedy で定義されている電子メールアドレスが一致したため、正常に相関が確立されました。Robert Blinn の Remedy アカウントについては、相関を確立できませんでした。Remedy 上の電子メールアドレスが別名で設定されていたことが原因です。このシナリオのほかのユーザーは Remedy アカウントを持っていません。

高速一括追加のシナリオ

次の手順は、「作成」処理を使用して、数名のユーザーを Identity Manager に迅速に追加する方法を示します。CSV ファイルをシステムに読み込む前に、CSV ファイルで参照されるすべてのリソースが Identity Manager 内で定義されている必要があります。

1. 一括処理の作成を実行するために必要な情報を格納した CSV ファイルを生成します。CSV ファイルの形式の詳細については、「一括処理の作成」を参照してください。

2. CSV ファイルにパスワードが含まれていない場合、デフォルトのパスワードポリシーオプションを設定し、パスワードがシステムによって生成されるようにします。これを行うには、「**設定**」タブをクリックし、左側の「**ポリシー**」をクリックします。「**Default Lighthouse Account Policy**」リンクをクリックします。次に、「**パスワード提供者**」ドロップダウンメニューから「生成」オプションを選択します。

3. デフォルトの「ユーザー作成」プロビジョニングタスクに基づいて、新しいプロビジョニングタスクを作成します。

   XML 画面で、`TaskDefinition` タグの `resultLimit` パラメータの値を 0 に編集します。この値は、タスクの結果を保持する秒数を指定します。

   次に、タスクから次のセクションを削除します。

   ```
   <Activity id='1' name='Approve'>

   ...

   </Activity>


   <Activity id='3' name='Notify'>

   ...

   </Activity>
   ```

注　　プロビジョナを呼び出すアクティビティーは残す必要があります。

   残っていないと、ユーザーが正しく作成されません。タスクの名前を「Fast Create User」のような値に忘れずに変更してください。

4. デフォルトのユーザーフォームに基づく新しいフォームを作成します。

   XML 画面で、`<Form>... </Form>` 構造全体を次のように置き換えます。

```
<Form help='account/modify-help.xml'>
   <Field name="viewOptions.Process">
      <Expansion>
         <s>Fast Create User</s>
      </Expansion>
   </Field>
</Form>
```

「Fast User Form」のようにユーザーフォームの名前を忘れずに変更してください。

5. アカウントを Identity Manager に読み込むために使用する Identity Manager アカウントを作成します。手順 4 で作成したユーザーフォームをこのユーザーに割り当てます。
6. 前の手順で作成したアカウントを使用して、Identity Manager にログインします。
7. 一括処理の作成を実行します。

第 5 章

データエクスポータ

この章では、データエクスポータの機能について説明し、それを配備するために必要な情報を提供します。

データエクスポータとは

Identity Manager では、広範囲のシステムおよびアプリケーションでユーザーアカウント情報が処理され、変更が企業ポリシーに準拠した内容となるための、制御され、監査された環境を提供します。Identity Manager は、「データライト」アーキテクチャーです。管理対象のシステムおよびアプリションに最小限のアカウント情報がローカルに格納され、必要に応じて、実際のシステムまたはアプリケーションからデータがフェッチされます。

このアーキテクチャーは、プロビジョニング処理中にデータの重複を減らしたり、無効なデータが転送される危険性を最小限にしたりするのに役立ちますが、ローカルに格納されたアカウントデータを持つことが望ましい場合もあります。たとえば、基本システムやアプリケーションにアクセスしないでアカウント情報のクエリーを実行できると、特定の属性値を持つすべてのアカウントを識別するなどの一部の処理で、大幅にパフォーマンスが改善されることがあります。一般に、システムまたはアプリケーションのアカウントデータの使用は、プロビジョニング処理よりもレポート処理に関連しますが、そのデータが組織にとって価値がある場合もあります。

Identity Manager では、「データライト」アーキテクチャーに加えて、「現在のデータのみ」のデータモデルが使用されます。つまり、履歴レコード (監査ログとシステムログ以外) は保持されません。このモデルの利点は、オペレーショナルリポジトリのサイズが管理対象のアカウント、システム、およびアプリケーションの数に比例する傾向があることです。その結果、プロビジョニングシステム自体で必要なメンテナンスが少なくなります。ただし、Identity Manager で処理されるデータは、履歴処理で役立つ場合があります。

たとえば、次のような質問では履歴データが必要です。

- A から B の期間にシステム X にアクセスしたユーザー、およびそのアクセスを承認したユーザーはだれですか。
- 最近 48 時間内に処理されたプロビジョニング要求の数、および各要求の処理にかかった時間はどれくらいですか。

データエクスポータを使用すると、上記のような質問に回答するために必要なアカウントおよびワークフローデータなどの Identity Manager で処理される数多くの情報を選択的に取得できます。Identity Manager では、商用データベースの変換、レポート、および分析ツールを使用してクエリーおよび変換の基盤として追加処理または使用するために、このデータはデータウェアハウスに転送できる形式で生成されます。

Identity Manager からデータをエクスポートする必要はありません。このタイプの履歴データを追跡する必要がない場合は、このデータを保持する必要はありません。このデータが必要な場合は、Identity Manager に影響を与えることなく、独自のデータ有効期限および保持期間のポリシーを自由に確立できます。

エクスポート可能なデータ型

データエクスポータは、持続的なデータと一時的なデータの両方をエクスポートできます。持続的なデータとは、Identity Manager によってリポジトリに格納されたデータを指します。一時的なデータとは、デフォルトで Identity Manager リポジトリに格納されないデータ、または変更済みのレコードの定期的なフェッチを除外するライフサイクルを持つデータのいずれかです。一部のデータ型は、一時的と持続的の両方の場合があります (タスクインスタンスや作業項目など)。これらのデータ型は、外部から予測不可能な時点で Identity Manager によって削除されるため、一時的であると考えられます。

Identity Manager では、次のデータ型がエクスポートされます。

表 5-1 サポートされるデータ型

データ型	永続	説明
Account	持続的	User と ResourceAccount 間のリンケージを含むレコード
Entitlement	持続的	特定ユーザーのアテステーションのリストを含むレコード
LogRecord	持続的	単一の監査レコードを含むレコード
ObjectGroup	持続的	組織として設計されたセキュリティーコンテナ
Resource	持続的	アカウントがプロビジョニングされるシステムまたはアプリケーション
ResourceAccount	一時的	特定リソース上にアカウントを構成する属性セット

表 5-1　サポートされるデータ型 (続き)

データ型	永続	説明
Role	持続的	アクセス用の論理コンテナ
Rule	持続的	Identity Manager で実行可能なロジックのブロック
TaskInstance	一時的および持続的	実行中または完了済みのプロセスを示すレコード
User	持続的	0 (ゼロ) 個以上のアカウントを含む論理ユーザー
WorkflowActivity	一時的	Identity Manager ワークフローの単一アクティビティー
WorkItem	一時的および持続的	Identity Manager ワークフローからの手動アクション

データエクスポータを使用すると、ウェアハウスの正確な要求に応じて、各データ型のエクスポート方針を定義できます。たとえば、一部のデータ型ではオブジェクトへのすべての変更をエクスポートする必要がある一方で、ほかのデータ型では一定の間隔でエクスポートすることで十分であり、データへの中間の変更がスキップされる可能性があります。

エクスポートする型を選択できます。型を選択すると、その型の新規および変更済みインスタンスがすべてエクスポートされます。削除済みのオブジェクトをエクスポートするように、持続的なデータ型を設定することもできます。

データエクスポータアーキテクチャー

データエクスポータを有効にすると、Identity Manager では、リポジトリ内のテーブルにレコードとして指定されたオブジェクト (データ型) への検出済みの各変更が格納されます。データ型ごとに設定可能な間隔で、エクスポートするレコードを選択するための 2 つのクエリーが実行されます。

- 1 番目のクエリーでは、最後のエクスポート以降に変更された、特定の型の持続的なオブジェクトがリポジトリ内で検索されます。ウェアハウスタスクでは、これらのレコードがエクスポートされ、最近変更されたレコードのタイムスタンプが特定され、次回クエリーを実行する開始点としてこの値が使用されます。
- 2 番目のクエリーでは、キューテーブルが検索されます。特定のデータ型のすべてのレコードが検出され、それらのレコードがエクスポートされたあとに、キューから削除されます。クエリーの完了後に追加されたレコードは、次のサイクルでエクスポートされます。

エクスポートされたレコードは順序付けられません。ただし、エクスポート済みのデータには、ウェアハウスの後続クエリーで時系列順にデータを配置できるフィールドがあります。

一般的な配備では、データエクスポータによってデータがステージングテーブルのセットに書き込まれます。Identity Manager では、サポートされるデータベースタイプごとに、これらのテーブルを定義する SQL スクリプトが用意されています。Identity Manager の配備にエクスポートが必要な拡張属性が含まれていない限り、これらのテーブルを変更する必要がありません。ただし、エクスポートする拡張属性がある場合は、エクスポートスキーマをカスタマイズし、これらの属性を処理するための独自のファクトリクラスをコンパイルする必要があります。詳細については、96 ページの「データエクスポータをカスタマイズする」を参照してください。

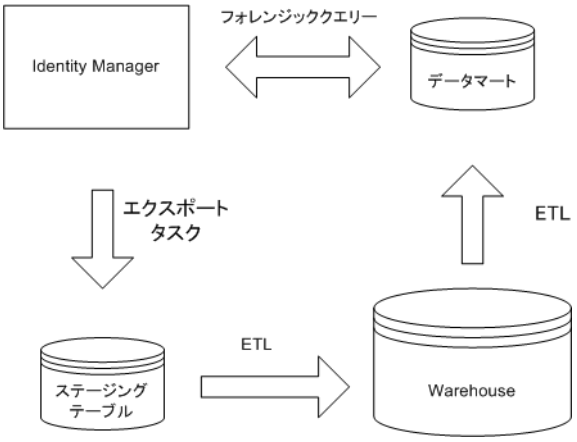
ステージングテーブルにデータをエクスポートすると、データウェアハウスおよび最終的にデータマートへの格納用にデータを処理できるように、独自の ETL (Extract, Transform, and Load) インフラストラクチャーを作成できます。タイムスタンプ処理は、一般に実装される変換です。システムでは、`YYYY-MM-DD hh:mm:ss` の形式の `java.sql.Timestamp` が使用されます。タイムスタンプには曜日が明示的に指定されていませんが、変換を使用すると曜日を抽出できます。

ウェアハウスおよびデータマートに情報を転送する必要がない場合は、そのステージングテーブルが最終宛先であると見なすことができます。この場合、読み取り処理と書き込み処理で同一の接続情報を必ず使用してください。データエクスポータの設定の詳細については、『Identity Manager 管理ガイド』を参照してください。

フォレンジッククエリーを使用すると、Identity Manager で、データウェアハウス (または単一環境のステージングテーブル) に格納されたデータを読み取ることができます。ユーザー、ロール、または関連するデータ型の現在値または履歴値に基づいて、ユーザーまたはロールを特定できます。フォレンジッククエリーは「ユーザーの検索」または「ロールの検索」レポートと似ていますが、履歴データと比較してマッチング条件を評価できる点と、クエリー対象のユーザーまたはロール以外のデータ型の属性を検索できる点で異なります。フォレンジッククエリーの定義の詳細については、『Identity Manager 管理ガイド』を参照してください。

次の図は、データエクスポータを有効にしたときのデータフローを示します。

図 5-1



データエクスポータを計画する

データエクスポータを配備する前に、次の点について計画する必要があります。

- エクスポートするデータ量はどのくらいですか。エクスポートされたデータ型の数によって、必要となるデータベーステーブルの数が決まります。データ型へのすべての変更をキューに入れたり、削除したオブジェクトまでエクスポートしたりすることを選択するとデータベース要件が大きくなります。詳細については、90 ページの「データベースの検討事項」を参照してください。
- 専用のエクスポートサーバーが必要ですか。複雑なワークフローを実行するサーバー上でデータをエクスポートすると、パフォーマンスが低下する可能性があります。詳細については、92 ページの「エクスポートサーバーの検討事項」を参照してください。
- エクスポートする必要があるカスタム拡張属性がありますか。該当する場合は、エクスポートスキーマを更新し、Warehouse Interface Code (WIC) を再コンパイルしたあとに、エクスポートスキーマを更新する必要があります。詳細については、96 ページの「データエクスポータをカスタマイズする」を参照してください。

データベースの検討事項

データエクスポータは、Identity Manager リポジトリとしてサポートされている任意のデータベースにエクスポートできます。さらに、データエクスポータは、Hibernate 3.2 でサポートされている任意の RDBMS で処理するようにしてください。

Hibernate のサポート

データエクスポータでは、Identity Manager Java オブジェクトと RDBMS テーブル間の双方向のマッピングで Hibernate 3.2 が使用されます。Identity Manager では、ウェアハウス Beans と RDBMS テーブル間のマッピングを制御するファイルセット (データ型ごとに 1 つ) が用意されています。これらのファイルは、`$WSHOME/exporter/hbm` ディレクトリに配置されています。

詳細については、96 ページの「データエクスポータをカスタマイズする」を参照してください。

Hibernate では、接続プールとして C3P0 が使用されます。C3P0 ではログエントリが JRE ログシステムに送信されます。デフォルトでは、INFO レベルのログが有効化されています。ログの対象を制限するには、`$JRE/lib/logging.properties` ファイルの最終行に次の行を追加します。

```
com.mchange.v2.c3p0.impl.level=SEVERE
com.mchange.v2.c3p0.level=SEVERE
com.mchange.v2.log.level=SEVERE
```

オブジェクト関係マッピング

Identity Manager では作業の実行に (Java) オブジェクトが使用されますが、これらのオブジェクトをリレーショナルデータベーステーブルのセットにエクスポートする場合は、オブジェクト関係マッピングと一般に呼ばれている変換をオブジェクトで実行する必要があります。この変換が必要であるのは、RDBMS 関係で表現できるデータ型と任意の Java オブジェクトで表現できるデータ型には相違点があるためです。たとえば、次の Java クラスを検討します。

```
class Widget {
  private String _id;
  private Map<String,Widget> _subWidgets;
  ...
}
```

_subWidgets フィールドは入れ子構造であるため、関係用語で表現されると、このクラスで問題が発生します。subWidgets が共有された Widget オブジェクトの 2 つの階層を RDBMS テーブルのセットに分解し、階層の 1 つを削除しようとすると、参照関連の問題が発生します。

表現上の相違点に対処するために、Identity Manager では、エクスポートできるデータ型にいくつかの制約が置かれます。特に、この制限を使用すると、最上位の Java オブジェクトにスカラー属性、スカラー属性のリスト、およびスカラー属性のマップを含めることができます。場合によっては、Identity Manager により柔軟な表現方法が必要になることがあります。このような問題を解決するために、Identity Manager には PseudoModel が導入されています。概念上は、PseudoModel はスカラー属性のみを含むデータ構造です。最上位の Java オブジェクトには、PseudoModel または PseudoModel のリストである属性を含めることができます。PseudoModel は、拡張不可能な Identity Manager 構造です。PseudoModel の例は次のとおりです。

```
class TopLevelModel
{
    private String _name;
    private List<PseudoModelPoint> _points;
}
class PseudoModelPoint
{
    private String _name;
    private String _color;
    private int _x;
    private int _y;
    private int _z;
}
```

PseudoModelPoint にはスカラー属性のみが含まれるため、Identity Manager では、TopLevelModel のオブジェクト関係変換を適切に実行できます。クエリーの注釈では、PseudoModel の色属性は次のようにアドレス可能です。

```
TopLevelModel.points[].color
```

Identity Manager データエクスポートスキーマを調査すると、PseudoModel 型がいくつか見つかります。これらの型では、最上位のエクスポートモデルのより複雑なデータの一部が表現されます。PseudoModel は直接エクスポートされないため、PseudoModel のクエリーを直接実行できません。PseudoModel は単に、最上位モデルの属性で保持される構造化データです。

データベーステーブル

ウェアハウス DDL で定義された RDBMS テーブルの数は、エクスポートされるモデルの種類の数、および各モデルでエクスポートする属性の種類によって異なります。一般に、各モデルには 3 ～ 5 つのテーブルが必要です。それぞれのテーブルには、リスト / マップ値の属性が格納されます。デフォルト DDL には、約 50 個のテーブルが含まれています。エクスポートスキーマについて調べてから、一部の属性テーブルが除外されるように Hibernate マッピングファイルを変更することもできます。

領域の要件

エクスポータウェアハウスで必要な領域の量は、次の条件によって異なります。

- エクスポートするオブジェクト
- レコードがエクスポートウェアハウスに保持される期間
- Identity Manager サーバーの稼動状況

通常、WorkflowActivity および ResourceAccount はもっともボリュームが大きいエクスポートモデルです。たとえば、単一のワークフローに複数のアクティビティーが含まれる場合があり、各ワークフローが実行されると、Identity Manager によって、ウェアハウスに書き込まれる数多くの新規レコードが作成される場合があります。ユーザーオブジェクトを編集すると、アカウントごとに 1 つの ResourceAccount レコードがユーザーにリンクされる場合があります。TaskInstance、WorkItem、および LogRecord も、ボリュームの大きいモデルです。単一の Identity Manager サーバーでは、1 時間で 50,000 個以上のオブジェクト変更が発生し、エクスポートされる可能性があります。

エクスポートサーバーの検討事項

特に大量のデータをエクスポートする予定の場合、専用サーバーでエクスポートタスクを実行することを検討するようにしてください。エクスポートタスクは、Identity Manager からウェアハウスに効率的にデータを転送し、エクスポート処理中に CPU を最大限消費します。専用サーバーを使用しない場合は、大量データのエクスポート中にレスポンス時間が大幅に低下するため、対話型トラフィックのサーバー処理を制限するようにしてください。

エクスポートタスクでは、主に Identity Manager リポジトリとステージングテーブルとの入出力処理が実行されます。キューに入れられたレコード数が増加するにつれてメモリーを増設する必要がありますが、エクスポートタスクのメモリー要件は控えめです。一般に、エクスポートタスクは入出力の速度によって制約され、スループットを向上させるために複数の同時実行スレッドが使用されます。

適切なサーバーを選択するには、検証が必要となります。入力 (Identity Manager リポジトリ) または出力 (ステージングテーブル) の転送速度が遅い場合、エクスポートタスクで最新の CPU は飽和状態になりません。エクスポート処理ではエクスポートサイクルの開始時にのみクエリーが発行されるため、入力パスのクエリー速度は問題になりません。大部分の時間は、レコードの読み書きに費やされます。

Identity Manager には、入出力データ速度を決定する JMX MBeans が用意されています。これらの MBeans の詳細については、『Identity Manager 管理ガイド』を参照してください。

デフォルト DDL を読み込む

このセクションでは、データベースの作成およびデフォルト DDL (Data Definition Language) の読み込みに必要なコマンドを列挙します。エクスポート DDL は、現在のエクスポートスキーマに一致するように、Identity Manager で提供されたツールによって生成されます。

`create_warehouse` スクリプトは、`$WSHOME/exporter` ディレクトリに配置されています。Identity Manager では、対応する `drop_warehouse` スクリプトも同じディレクトリに含まれています。

DB2

システム DBA として次のようなスクリプトを実行します。スクリプトを実行する前に、必ず `idm_warehouse` データベースおよび `idm_warehouse/idm_warehouse user` を作成してください。

```
CONNECT TO idm_warehouse USER idm_warehouse using 'idm_warehouse'
CREATE SCHEMA idm_warehouse AUTHORIZATION idm_warehouse
GRANT CONNECT ON DATABASE TO USER idm_warehouse
```

DDL を読み込むには、`%WSHOME%¥exporter¥create_warehouse.db2` ファイルに次の行を追加します。

```
CONNECT TO idm_warehouse USER idm_warehouse using 'idm_warehouse'
```

次に、次のコマンドを実行します (Windows DB2 サーバーを想定)。

```
db2cmd db2setcp.bat db2 -f create_warehouse.db2
```

MySQL

システム DBA として次のようなスクリプトを実行します。

```
# Create the database (Schema in MySQL terms)

CREATE DATABASE IF NOT EXISTS idm_warehouse CHARACTER SET utf8
COLLATE utf8_bin;

# Give permissions to the "idm_warehouse" userid logging in from any
host.

GRANT ALL PRIVILEGES on idm_warehouse.* TO idm_warehouse IDENTIFIED
BY 'idm_warehouse';

# Give permissions to the "idm_warehouse" userid logging in from any
host.

GRANT ALL PRIVILEGES on idm_warehouse.* TO idm_warehouse@'%'
IDENTIFIED BY 'idm_warehouse';

# Give permissions to the "idm_warehouse" user when it logs in from
the localhost.

GRANT ALL PRIVILEGES on idm_warehouse.* TO idm_warehouse@localhost
IDENTIFIED BY 'idm_warehouse';
```

DDL を読み込むには、次のコマンドを実行します。

```
# mysql -uidm_warehouse -pidm_warehouse -Didm_warehouse <
create_warehouse.mysql
```

Oracle

システム DBA として次のようなスクリプトを実行します。

```
-- Create tablespace and a user for warehouse

CREATE TABLESPACE idm_warehouse_ts
   DATAFILE 'D:/Oracle/warehouse/idm_warehouse.dbf' SIZE 10M
   AUTOEXTEND ON NEXT 10M
   DEFAULT STORAGE (INITIAL 10M NEXT 10M);

CREATE USER idm_warehouse IDENTIFIED BY idm_warehouse
   DEFAULT TABLESPACE idm_warehouse_ts
   QUOTA UNLIMITED ON idm_warehouse_ts;

GRANT CREATE SESSION to idm_warehouse;
```

DDL を読み込むには、次のコマンドを実行します。

```
sqlplus idm_warehouse/idm_warehouse@idm_warehouse <
create_warehouse.oracle
```

SQL Server

システム DBA として次のようなスクリプトを実行します。必要に応じて、行のコメントを解除します。

```
CREATE DATABASE idm_warehouse
GO

--For SQL Server authentication:
-- sp_addlogin user, password, defaultdb
--For Windows authentication:
-- sp_grantlogin <domain\user>

--For SQL Server 2005:
--CREATE LOGIN idm_warehouse WITH PASSWORD = 'idm_warehouse',
DEFAULT_DATABASE = idm_warehouse sp_addlogin 'idm_warehouse',
'idm_warehouse', 'idm_warehouse'

USE idm_warehouse
GO

--For SQL Server 2005 SP2 create a schema - not needed in other
versions:
--CREATE SCHEMA idm_warehouse
--GO

--For SQL Server 2005 SP2 use CREATE user instead of
sp_grantdbaccess
--CREATE USER idm_warehouse FOR LOGIN idm_warehouse with
DEFAULT_SCHEMA = idm_warehouse

sp_grantdbaccess 'idm_warehouse'
GO
```

DDL を読み込むには、次のコマンドを実行します。

```
osql -d idm_warehouse -U idm_warehouse -P idm_warehouse <
create_warehouse.sqlserver
```

データエクスポータをカスタマイズする

データエクスポートには、事実上、内部 (ObjectClass) スキーマと外部 (Export) スキーマの 2 レベルのスキーマがあります。これらのスキーマでは、Identity Manager の複数のリリースに準拠したデータ「インタフェース」が提供されます。準拠とは、属性名、データ型、およびデータの意味が変更されないことを意味します。属性は削除できますが、別の意味を表すためにその属性名を再使用することはできません。属性はいつでも追加できます。準拠するスキーマを使用すると、スキーマのバージョンに関係なくレポートを書き込むことができ、今後のバージョンでも変更なしで実行できます。

ObjectClass スキーマではデータの外観について Identity Manager サーバーのプログラムに通知します。一方、外部スキーマではデータの外観についてウェアハウスに通知します。内部スキーマはリリースごとに異なりますが、外部スキーマではリリース間で準拠が保持されます。

Identity Manager ObjectClass スキーマ

User および Role 型の ObjectClass スキーマは拡張できますが、その他のスキーマは変更できません。ObjectClass スキーマは、データオブジェクト自体にアクセスするために、Identity Manager サーバーで実行されるプログラムで使用されます。このスキーマは Identity Manager 内にコンパイルされ、Identity Manager 内で格納および処理されるデータを表現します。

このスキーマは Identity Manager の複数のバージョン間で変更される場合がありますが、エクスポートスキーマが原因で、データウェアハウスにとっては抽象的です。ObjectClass スキーマでは、Identity Manager の持続的なオブジェクトレイヤーの最上部にスキーマが抽象的に表示されます。これは、Identity Manager リポジトリに格納されるデータオブジェクトです。

カスタムユーザーおよびロール属性 (拡張属性とも呼ばれる) は、`IDMSchemaConfiguration` オブジェクトで定義されます。ObjectClass スキーマへの拡張属性の追加の詳細については、139 ページの「設定オブジェクトを編集する」を参照してください。

エクスポートスキーマ

エクスポートスキーマには、ウェアハウスに書き込むことができるデータが定義されます。2 つのスキーマ間の相違点は非常に少ないですが、デフォルトでは ObjectClass スキーマのサブセットに制限されています。ObjectClass スキーマは Java オブジェクトで表現されますが、エクスポートスキーマには Java オブジェクトと RDBMS テーブル間の双方向のマッピングが必要です。

拡張属性を IDMSchemaConfiguration オブジェクトに追加したあとに、エクスポートスキーマでも同じ属性を定義する必要があります。この属性は、$WSHOME/model-export.xml ファイルで定義します。このファイルで Role または User モデルを検索し、属性を定義する field 要素を追加します。field 要素には、次のパラメータを含めることができます。

表 5-2 エクスポート属性パラメータ

パラメータ	説明
name	属性の名前。この値は、IDMSchemaConfiguration オブジェクトに割り当てられた名前と一致する必要があります。
type	属性のデータ型。完全 Java クラス名 (java.lang.String や java.util.List など) を指定する必要があります。
introduced	オプション。属性がスキーマに追加されたリリースを指定します。
friendlyName	データエクスポータの設定ページに表示されるラベル。
elementType	type パラメータが java.util.List である場合は、このパラメータにはリスト内の項目のデータ型が指定されます。共通の値には、java.lang.String と com.sun.idm.object.ReferenceBean が含まれます。
referenceType	elementType パラメータが com.sun.idm.object.ReferenceBean である場合は、このパラメータは別の Identity Manager オブジェクトまたは擬似オブジェクトを参照します。
forensic	関係の決定に属性が使用されることを示します。指定可能な値は、User および Role です。
exported	false に設定すると、属性はエクスポートされません。データエクスポータのデータ型設定ページで、デフォルト属性を非表示にする場合は、属性定義に exported=false を追加します。 エクスポートが無効にされたデフォルトスキーマに属性をエクスポートできるようにするには、カスタム WIC ライブラリを作成する必要があります。
クエリー可能	false に設定すると、このフィールドをフォレンジッククエリーで使用できません。
max-length	値の最大長。

次の例では、User モデルの一部としてエクスポートスキーマに telno という拡張属性を追加します。

```
<field name='telno'
   type='java.lang.String'
   introduced='8.0'
   max-length='20'
   friendlyName='Telephone Number'>
   <description>The phone number assigned to the user.</description>
</field>
```

ウェアハウスインタフェースコードを変更する

Identity Manager では、ウェアハウスインタフェースコード (WIC) はバイナリおよびソースの形式で提供されます。多くの配備ではバイナリ形式 (変更なし) で WIC コードを使用できますが、一部の配備ではその他の変更を行う場合もあります。WIC コードでは、エクスポートで使用される 2 つのインタフェース、およびフォレンジッククエリーインタフェースで使用される 3 番目のインタフェースを実装する必要があります。

デフォルトの WIC 実装では、RDBMS テーブルのセットに書き込まれます。多くのアプリケーションではこれで十分ですが、JMS キューまたはその他の一部のコンシューマに日付を書き込むために、カスタム WIC コードを作成することもできます。

com.sun.idm.exporter.Factory および com.sun.idm.exporter.Exporter クラスは、データのエクスポートに使用されます。エクスポートコードは、格納に適した形式へのモデル (Java データオブジェクト) の変換に関与します。一般に、これはリレーショナルデータベースへの書き込みを意味します。その結果、WIC コードはオブジェクトから関係への変換に関与します。

デフォルトの WIC 実装では、オブジェクト関係マッピングを提供するために Hibernate が使用されます。このマッピングは、Hibernate の .hbm.xml マッピングファイルで制御されます。同様にこのファイルは、エクスポートスキーマに基づいて生成されます。Hibernate では、その作業に Java bean スタイルのデータオブジェクトの使用が優先されます。これを実現するために、さまざまな get および set メソッドがあります。WIC コードでは、エクスポートスキーマと一致する Bean および Hibernate ファイルが生成されます。Hibernate で必要なマッピング機能が提供される場合は、任意の WIC コードを手動で変更する必要がなくなる場合もあります。

WIC ファイルは、*InstallationDirectory*/REF/exporter ディレクトリに配置されています。

新規ファクトリクラスを生成する

Identity Manager を使用すると、カスタム User および Role 属性を ObjectClass スキーマに追加できます。これらの属性 (拡張属性と呼ばれる) は、エクスポートスキーマに追加し、ウェアハウスインタフェースコード (WIC) を再生成し、コードを配備しない限り、エクスポートできません。

拡張属性が追加されると、エクスポートスキーマ制御ファイルを編集し、属性を追加する必要があります。属性をエクスポータから除外する場合は、単にスキーマフィールドに `exported='false'` を指定し、WIC コードを再生成します。

WIC コードを変更するには、次のものをシステムにインストールする必要があります。

- Identity Manager のインストール
- Java 1.5 SDK
- ant 1.7 (以上)
- テキストエディタ

拡張属性のエクスポートに必要なステップは次のとおりです。

1. WIC ソースコードを REF キットから入手します。
2. `WSHOME` 環境変数を Identity Manager のインストールディレクトリに設定します。
3. エクスポートスキーマ制御ファイル `$WSHOME/model-export.xml` をバックアップしてから、編集します。
4. WIC ソースの最上位ディレクトリに移動します。このディレクトリには、`build.xml`、`BeanGenerator.java`、および `HbmGenerator.java` というファイルが含まれているはずです。
5. アプリケーションサーバーを停止します。
6. 環境から `CLASSPATH` を削除します。

注	次のステップで `ant rebuild` を実行する前に、環境から `CLASSPATH` を削除する必要があります。

7. `ant rebuild` コマンドを使用して、WIC コードを再ビルドします。
8. `ant deploy` コマンドを使用して、変更済みの WIC コードをアプリケーションサーバーに配備します。
9. アプリケーションサーバーを再起動します。

注	model-export.xml を変更し、前述のステップで示したように WIC を再ビルドすると、新規ウェアハウス DDL が生成されます。古いテーブルを削除し、新規 DDL をロードする必要があります。これにより、テーブル上にあるデータがすべて削除されます。

トラブルシューティング

エクスポータから転送されるデータの量と種類が増えると、データエクスポート中に問題が発生する可能性が高くなります。

Beans およびその他のツール

データエクスポータのパフォーマンスとスループットは、Identity Manager で提供される JMX 管理 Bean から監視できます。エクスポートデータのパフォーマンス影響を最小限にするために、Identity Manager では、揮発性のメモリーベースのキューが一部使用されています。予期せずにサーバーが終了した場合は、これらのキューのデータは失われます。これらのキューのサイズを一定期間監視することにより、このような危険にさらされているかどうかを判断できます。

Model Serialization の制限

適切な時間にエクスポートできることを保証するには、データエクスポータではいくつかのオブジェクトをキューに入れる必要があります。これらのオブジェクトは、Java 直列化によってキューに入れられます。ただし、直列化できないエクスポート済みオブジェクトにデータが含まれる可能性があります。この場合、エクスポータコードでは直列化不可能なデータが検出され、問題を示すトークンで置換されます。これにより、残りのオブジェクトをエクスポートできるようになります。

レジストリポーリングの設定

各タイプには、独立したエクスポートサイクルを指定できます。管理者インタフェースでは、ほどんどの用途に十分に対応できる単純なサイクルを定義するための簡単な方法が提供されます。ただし、エクスポートサイクルは、さらに優れた柔軟性をサポートするネイティブ cron のスタイルで指定することもできます。

トレースとログ

デフォルト WIC コードでは、エクスポート済みデータオブジェクトのオブジェクト/RDBMS マッピングを提供するために Hibernate が使用されます。ただし、Hibernate ライブラリを使用することは、トレースおよびログが完全には統合されないことを意味します。com.sun.idm.warehouse.* package を使用すると、実際の WIC コードのトレースを実行できます。ただし、Hibernate のログを有効化するには別の技術が必要です。

Hibernate セッションを開始するコードに Hibernate プロパティーを渡すには、`DatabaseConnection` 設定オブジェクトに属性を追加します。属性名に「X」という接頭辞を付ける必要があります。たとえば、ネイティブプロパティー名が `hibernate.show_sql` である場合は、設定オブジェクトに `Xhibernate.show_sql` と定義する必要があります。次の例では、Hibernate によって生成済みの SQL がアプリケーションサーバーの標準出力に表示されます。

```
<Attribute name='Xhibernate.show_sql' value='true'>
```

デフォルトでは、接続プールに C3P0 が Hibernate で使用されます。C3P0 でログを取得するには、`java.logging` 機能が使用されます。この機能は、`$JRE/lib/logging.properties` ファイルで制御されます。

第 6 章

ユーザーアクションを設定する

この章では、Identity Manager の管理者インタフェースにカスタムタスクを追加する方法と、次の 2 つのインタフェース領域から実行できるユーザーアクションの設定方法について説明します。

- 「ユーザーアカウントの検索結果」ページ
- 「アカウント」ページ上の「ユーザー」アプレット

注 カスタムタスクを追加するには、既存の `TaskDefinition` を編集する必要があります。Identity Manager IDE を使用すると、タスク定義を参照および編集できます。Identity Manager IDE のインストールおよび設定の指示書は、https://identitymanageride.dev.java.net から入手できます。

カスタムタスクを追加する

カスタムタスクを追加するには、次の一般的な手順に従います。

- タスクへの認証の設定
- タスクのリポジトリへの追加

カスタムタスクへの認証を設定する

通常は、タスクへのアクセスを特定の管理者の集合に制限するように、カスタムタスクへの認証を設定します。認証を設定するには、次の手順を実行します。

1. 新しい `AuthType` をタスクのリポジトリに追加します。
2. タスクの新しい `AdminGroup` (機能) を作成します。
3. 新しい機能を 1 名以上の管理者に許可します。

手順 1 : AuthType を作成する

新しい AuthType は、既存の TaskDefinition、TaskInstance、および TaskTemplate の各 AuthType を拡張して作成してください。AuthType を追加するには、リポジトリの AuthType 設定オブジェクトを編集して、タスクの新しい AuthType 要素を追加します。

<AuthType> 要素を使用して、新しい AuthType を作成します。この要素には、1 つの必須プロパティー name があります。次の例は、<AuthType> 要素の正しい構文を示します。

AuthType の作成後、リポジトリの AuthType 設定オブジェクトを編集して、新しい <AuthType> 要素を追加する必要があります。

次の例は、複数のユーザーを新しい組織に移動するカスタムタスクの追加方法を示します。

コード例 6-1 新しい組織への複数ユーザーの移動

```
<Configuration name='AuthorizationTypes'>
      <Extension>
        <AuthTypes>
          <AuthType name='Move User'
extends='TaskDefinition,TaskInstance,TaskTemplate'/>
        </AuthTypes>
      </Extension>
</Configuration>
```

手順 2 : AdminGroup を作成する

次に、新しく作成した AuthType に Right.VIEW を許可する AdminGroup を作成します。これを行うには、新しい管理者グループを定義した XML ファイルを作成し、このファイルを Identity Manager リポジトリにインポートする必要があります。

コード例 6-2 AdminGroup の作成

```
<?xml version='1.0' encoding='UTF-8'?>
    <!DOCTYPE Waveset PUBLIC 'waveset.dtd' 'waveset.dtd'>
    <Waveset>
    <AdminGroup name='Move User'
                protected='true'
                displayName='UI_ADMINGROUP_MOVE_USER'
                description='UI_ADMINGROUP_MOVE_USER_DESCRIPTION'>
      <Permissions>
        <Permission type='Move User' rights='View'/>
      </Permissions>
      <MemberObjectGroups>
        <ObjectRef type='ObjectGroup' id='#ID#All' name='All'/>
      </MemberObjectGroups>
    </AdminGroup>
    </Waveset>
```

注 displayName 属性と description 属性はメッセージカタログキーです。メッセージカタログに見つからない場合、これらの属性は属性に見つかった時点で表示されます。メッセージカタログキーを使用する場合、WPMessages.properties またはカスタムメッセージカタログのどちらかにメッセージを追加する必要があります。

手順 3 : 管理者に機能を許可する

最後に、新しく定義したタスクを実行するためのアクセス権限を管理者に許可する必要があります。これは、次の 2 つの方法のいずれかで行うことができます。

- 新しい機能を直接割り当てる
- 新しい機能を管理者ロールに (直接、または機能規則を使用して) 追加し、そのロールを割り当てる

タスクをリポジトリに追加する

タスクへの認証を設定したあとで、タスクをリポジトリに追加することができます。タスクは、Identity Manager IDE を使用しての定義または XML 形式でのインポートが可能な標準の `TaskDefinition` です。たとえば、複数のユーザーの組織を変更するタスクは次の例のようになります (この例は `samples` ディレクトリにある)。

コード例 6-3 複数ユーザーの組織の変更

```
<?xml version='1.0' encoding='UTF-8'?>
  <!DOCTYPE TaskDefinition PUBLIC 'waveset.dtd' 'waveset.dtd'>
<!-- MemberObjectGroups="#ID#Top" authType="Move User" name="Change
Organizations" taskType="Workflow" visibility="runschedule"-->
       <TaskDefinition authType='MoveUser'
           name='Change Organizations' taskType='Workflow'
           executor='com.waveset.workflow.WorkflowExecutor'
           suspendable='true'
             syncControlAllowed='true' execMode='sync'
             execLimit='0' resultLimit='0'
                  resultOption='delete' visibility='runschedule'
               progressInterval='0'>
       <Form name='Change Organization Form'
         title='Change Organization Form'>
           <Display class='EditForm'/>
             <Include>
                 <ObjectRef type='UserForm' name='User Library'/>
                 <ObjectRef type='UserForm' name='Organization Library'/>
             </Include>
        FieldRef name='namesList'/>
         <FieldRef name='orgsList'/>
         <FieldRef name='waveset.organization'/>
       </Form>
       <Extension>
      <WFProcess name='Change Organizations' title='Change Organizations'>
         <Variable name='waveset.organization'/>
         <Variable name='userObjectIds' input='true'>
          <Comments>The names of the accounts to change the organization
on.</Comments>
         </Variable>
```

コード例 6-3　　複数ユーザーの組織の変更 (続き)

```
<?xml version='1.0' encoding='UTF-8'?>
              Activity id='0' name='start'>
                 <ReportTitle>
                    <s>start</s>
                 </ReportTitle>
                   <Transition to='Process Org Moves'/>
                 </Activity>
                 <Activity id='1' name='Process Org Moves'>
                    <Action id='0' process='Move User'>
                     <Iterate for='currentAccount' in='userObjectIds'/>
                     <Argument name='userId' value='$(currentAccount)'/>
                     <Argument name='organizationId'
                          value='$(waveset.organization)'/>
                     </Action>
                    <Transition to='end'/>
                  </Activity>

        <Activity id='2' name='end'/>
        </WFProcess>
       </Extension>
   <MemberObjectGroups>
     <ObjectRef type='ObjectGroup' id='#ID#Top' name='Top'/>
   </MemberObjectGroups>
       </TaskDefinition>
```

例について

前掲の例については、次の点に注意してください。

- タスクの `authType` 属性は「Move User」に設定されます。これにより、このタスクにアクセスできるのは、この AuthType を実行する機能を割り当てられたユーザーに限定されます。
- フォームには `namesList` および `orgsList` への FieldRef (フィールド参照) が含まれます。これらのフィールドは、それぞれユーザーライブラリと組織ライブラリで定義されます。これらのフィールドを含めると、すべての選択対象ユーザーおよび選択対象組織の名前のリストが表示されます。危険を伴う可能性のあるタスクについては、タスクを実行することの潜在的な影響をユーザーが認識できるように、これらのフィールドのどちらかまたは両方を含めるようにしてください。

- タスクには userObjectIds という名前の入力変数があります。この変数の内容は、「ユーザーアカウントの検索結果」ページ、または「アカウント」ページのユーザーアプレットで選択されたユーザーの名前または ID のリストです。すべての選択したユーザーに対して目的のアクションを実行するには、この変数を反復処理します。

次の表は、タスクへの入力として使用できる変数の一覧です。

表 6-1 タスク変数

変数	説明
userObjectIds	選択されたユーザーの ID のリスト。「ユーザーアカウントの検索結果」ページおよび「アカウント」ページから利用できます。「ユーザーアカウントの検索結果」ページから呼び出された場合、このリストの内容は選択されたユーザーの名前になります。
userNames	選択されたユーザーの名前のリスト。「ユーザーアカウントの検索結果」ページおよび「アカウント」ページから利用できます。
orgObjectIds	選択された組織の ID のリスト。「アカウント」ページからのみ利用できます。
orgNames	選択された組織の名前のリスト。「アカウント」ページからのみ利用できます。

このワークフローを有効にするには、次の例に示すように、ユーザーの組織を変更するためのサブプロセスをリポジトリに追加する必要もあります。

コード例 6-4 ユーザーの組織の変更

```
<?xml version='1.0' encoding='UTF-8'?>
  <!DOCTYPE Configuration PUBLIC 'waveset.dtd' 'waveset.dtd'>
  <!--  MemberObjectGroups="#ID#Top" configType="WFProcess"
name="Move User"-->
  <Configuration name='Move User' createDate='1083353996807'>
    <Extension>
      <WFProcess name='Move User' title='Move User'>
        <Variable name='userId' input='true'>
          <Comments> 移動するユーザーのアカウント ID。< /Comments>
        </Variable>
        <Variable name='organizationId' input='true'>
          <Comments> ユーザーの移動先組織の ID。
into.</Comments>
        </Variable>
```

コード例 6-4 ユーザーの組織の変更 (続き)

```
        <Activity id='0' name='Start'>
          <Transition to='Update Organization'/>
        </Activity>
        <Activity id='1' name='Update Organization'>
          <Action id='0' process='Update User View'>
            <Argument name='accountId' value='$(userId)'/>
            <Argument name='updates'>
              <map>
                <s>waveset.organization</s>
                <ref>organizationId</ref>
              </map>
            </Argument>
          </Action>
          <Transition to='End'/>
        </Activity>

        <Activity id='2' name='End'/>
      </WFProcess>
    </Extension>
    <MemberObjectGroups>
      <ObjectRef type='ObjectGroup' id='#ID#Top' name='Top'/>
    </MemberObjectGroups>
  </Configuration>
```

ユーザーアクションを設定する

カスタムアクションを開始するボタンおよびアクションメニュー選択の定義を設定する必要があります。「ユーザーアカウントの検索結果」ページと「アカウント」ページに表示されるボタンおよびアクションメニュー項目の定義は、User Actions Configuration 設定オブジェクトに格納されます。

User Actions Configuration オブジェクトを直接編集しないでください。その代わりに、ユーザーアクションを設定するために推奨される方法は次のとおりです。

- User Actions Configuration 設定オブジェクトを新しい設定オブジェクトにコピーします。
- 新しい設定オブジェクトを指すように System Configuration オブジェクトを修正します。

一般に、ユーザーアクションを設定する手順は次のとおりです。

1. User Actions Configuration 設定オブジェクトを新しい XML ファイルにコピーします。

2. 新しいオブジェクトの名前を「My User Actions Configuration」に変更します。
3. My User Actions Configuration オブジェクトに必要な変更を行います。
4. 「交換ファイルのインポート」ページから、XML ファイルを Identity Manager にインポートします。
5. SystemConfiguration の `userActionsConfigMapping` 属性の値を「My User Actions Configuration」に変更します。

設定オブジェクトは、次の設定セクションで構成されます。

表 6-2 User Actions Configuration 属性

属性	説明
`findUsersButtons`	管理者インタフェースの「ユーザーアカウントの検索結果」ページのボタン定義のリスト。
`userApplet.userMenu`	ユーザーに対する処理メニューのメニュー項目定義のリスト。このメニューは、管理者インタフェースの「アカウント」ページのアプレット内でユーザーを右クリックしたときに表示されます。
`userApplet.organizationMenu`	組織に対する処理メニューのメニュー項目定義のリスト。このメニューは、「アカウント」ページのアプレット内で組織を右クリックしたときに表示されます。

各セクションでは、インタフェースで表示するユーザーアクションのリストが定義されます。ボタンおよびメニューの設定項目には共通の基本プロパティーがあります。どちらの項目にも、インタフェースに特有のいくつかの拡張が含まれます。

次の抜粋部分は、各リストに Change Organization (組織変更) タスクを追加するためのカスタマイズされたユーザーアクション設定の例です。

コード例 6-5 各リストへの組織変更タスクの追加

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE Waveset PUBLIC 'waveset.dtd' 'waveset.dtd'>
<Waveset>

<Configuration name='My User Actions Configuration'>
  <Extension>
    <Object>
      <!-- ユーザー検索の結果ページのボタン -- -->
      <Attribute name='findUsersButtons'>
        <List>
          <Object>
            <Attribute name='textKey'        value='UI_NEW_LABEL' />
```

コード例 6-5 各リストへの組織変更タスクの追加 (続き)

```
            <Attribute name='commandName'    value='New' />
            <Attribute name='requiredPermission'>
              <Object>
                <Attribute name='objectType' value='User' />
                <Attribute name='rights'     value='Create' />
              </Object>
            </Attribute>
            <Attribute name='alwaysDisplay' value='true' />
          </Object>
          ...
          <Object>
            <Attribute name='textKey'
value='UI_CHANGE_ORGANIZATIONS_LABEL' />
            <Attribute name='commandName'
value='Change Organizations' />
          </Object>
        </List>
</Attribute>
      <Attribute name='userApplet'>
        <Object>
          <!-- ユーザーが選択されたときに表示するメニュー -- -->
          <Attribute name='userMenu'>
            <List>
              <Object>
                <Attribute name='textKey'
value='UI_ACCT_JAVA_MENU_NEW_ORG' />
                <Attribute name='commandName'
value='New Organization' />
                <Attribute name='requiredPermission'>
                  <Object>
                    <Attribute name='objectType' value='ObjectGroup' />
                    <Attribute name='rights'     value='Create' />
                  </Object>
                </Attribute>
              </Object>
              ...
              <Object>
                <Attribute name='separator' value='separator' />
              </Object>
              <Object>
                <Attribute name='textKey'
value='UI_CHANGE_ORGANIZATIONS_MENU_LABEL' />
                <Attribute name='commandName'
value='Change Organizations' />
              </Object>
            </List>
</Attribute>
          <!-- 組織が選択されたときに表示するメニュー -- -->
          <Attribute name='organizationMenu'>
            <List>
              <Object>
                <Attribute name='textKey'
value='UI_ACCT_JAVA_MENU_NEW_JUNCTION' />
                <Attribute name='commandName'
```

コード例 6-5 各リストへの組織変更タスクの追加 (続き)

```
value='New Directory Junction' />
                <Attribute name='requiredPermission'>
                  <Object>
                    <Attribute name='objectType' value='ObjectGroup' />
                    <Attribute name='rights'     value='Create' />
                  </Object>
                </Attribute>
                <Attribute name='orgTypes' value='normal,dynamic' />
              </Object>
              ...
              <Object>
                <Attribute name='separator' value='separator' />
              </Object>
              <Object>
                <Attribute name='textKey'
value='UI_CHANGE_ORGANIZATIONS_MENU_LABEL' />
                <Attribute name='commandName'
value='Change Organizations' />
              </Object>
            </List>
          </Attribute>
        </Object>
      </Attribute>
    </Object>
  </Extension>
  <MemberObjectGroups>
    <ObjectRef type='ObjectGroup' name='All'/>
</MemberObjectGroups>
</Configuration>
</Waveset>
```

ユーザーアクション定義は、次のコア属性をサポートします。

表 6-3 サポートされるコア属性

属性	説明
textKey	ボタンまたはメニュー項目のテキストのメッセージカタログキー。
commandName	実行するコマンドの名前。この属性は、(「新規」や「ユーザーの削除」などの) ネイティブにサポートされているコマンド、または実行する TaskDefinition の名前に設定できます。
requiredPermission.objectType	この項目を表示するために権限が必要なオブジェクトの種類。これは、ネイティブにサポートされているコマンドのみに適用されます。タスク定義については、アクセスの制御に AuthType を使用するようにしてください。

表 6-3　サポートされるコア属性 (続き)

属性	説明
requiredPermission.rights	指定された objectType がこの項目を表示するために必要な権限名のコンマ区切りのリスト。これは、ネイティブにサポートされているコマンドのみに適用されます。タスク定義については、アクセスの制御に AuthType を使用するようにしてください。
alwaysDisplay	オプション。このボタンを常に表示するかどうかを指定します。この値を true に設定した場合、ユーザー検索で結果が何も返されない場合でもボタンが表示されます。この属性のデフォルト値は false です。 findUsersButtons セクションのみに適用されます。

userApplet セクションのユーザーアクション定義は、次の表に示す属性もサポートします。

表 6-4　userApplet ユーザーアクションでサポートされる属性

属性	説明
orgTypes	組織メニューに項目を表示する組織タイプのコンマ区切りリスト。指定できる値は、通常の組織を表す normal、動的組織を表す dynamic、仮想組織を表す virtual です。 この属性を指定しない場合、すべての組織タイプに対してメニュー項目が表示されます。
separator	<Object><Attribute name='separator' value='separator'/></Object> の形式の特殊項目。セパレータは管理者インタフェースに水平バーとして表示され、選択できません。

第 7 章

Identity Manager のプライベートラベリング

この章では、会社のイントラネット環境または企業のスタイルガイドラインに合わせて Identity Manager インタフェースを再定義するために必要な基本コンポーネントについて説明します。プライベートラベリングとは、このような企業のガイドラインに合わせてインタフェースをカスタマイズすることを言います。

プライベートラベリングのタスク

プライベートラベリングのタスクは、3 つのカテゴリに大別されます。

- **デフォルトのヘッダーコンテンツの変更**。ユーザーインタフェースと管理者インタフェースの両方で、企業ロゴの取り込み、デフォルトテキストの変更、色の変更を行います。
- **表示フォントとコンポーネントの色の変更**。`styles¥customStyle.css` のスタイルシートを使用して、アプリケーション全体にわたる変更を行います。
- **よく使用するページでの Identity Manager の動作の変更**。System Configuration オブジェクトを編集することによって行います。「**パスワードをお忘れですか ?**」ボタンの無効化を含むこれらのタスクは、製品インタフェースの再定義の間にユーザーによってたびたび実行されます。

アーキテクチャー機能

プライベートラベリングの作業には、次の表に示すコンポーネントの編集が含まれます。

表 7-1 カスタマイズ可能なコンポーネント

コンポーネント	インタフェース
`$WSHOME/styles/customStyle.css`	管理者 / ユーザー
`$WSHOME/WEB-INF/lib/idmcommon.jar`	管理者 / ユーザー
`$WSHOME/user/`**userFooter_beforeFirstTableRowTag.jsp**	ユーザーインタフェース
`$WSHOME/user/`**userFooter_beforeEndBodyTag.jsp**	ユーザーインタフェース
`$WSHOME/`**user/userFooter_beforeLastEndTableRowTag.jsp**	ユーザーインタフェース
`$WSHOME/`**includes/bodyEnd_beforeFirstTableRowTag.jsp**	管理者インタフェース
`$WSHOME/includes/`**bodyEnd_beforeEndBodyTag.jsp**	管理者インタフェース
`$WSHOME/`**includes/bodyEnd_beforeLastEndTableRowTag.jsp**	管理者インタフェース
`$WSHOME/`**index_quickLinks.jsp**	管理者 / ユーザー

スタイルシート

製品インタフェースでのテキストの表示特性に影響する、4 つのスタイルシートがあります。

- `lockhart.css` - Web アプリケーション用の Sun コーポレートインタフェーススタイルが含まれます。
- `style.css` - 両方のインタフェースにわたる、ページの表示属性を定義します。このファイルは、ヘッダーに含まれる画像も制御します。
- `customStyle.css` - `style.css` および `lockhart.css` に含まれるデフォルト設定への変更が含まれます。このファイルの設定は、`style.css` および `lockhart.css` の設定より優先されます。これら 2 つのファイルを編集する代わりに、`customStyle.css` でカスタマイズを行うことをお勧めします。
- `Styles-Help.css` - オンラインヘルプおよびポップアップヘルプ (i-Help) で使用されるスタイルクラスを定義します。

デフォルトテキスト

デフォルトテキストは、製品インタフェース全体を通して次の場所に出現します。

- フォームのタイトル、サブタイトル、ボタン、奇数行、偶数行、セクションヘッダー
- 一般テキスト
- 警告メッセージ
- ナビゲーションボタンのテキスト。利用可能および選択状態のナビゲーションボタンの両方を含む
- テーブルヘッダー / 本文テキスト

テキスト属性

次を含む属性を表示します。

- title - font-family、font-size、font-weight、color
- button - text-alignment、background-color
- text - title と同じ属性および text-decoration、white-space

デフォルトスタイル設定

`$WSHOME/styles/style.css` および `lockhart.css` の各ファイルには、デフォルトのスタイル設定が格納されています。これらのファイルは編集しないでください。

カスタマイズするファイル

`customStyle.css` ファイルにはカスタマイズの定義が格納されます。このファイルは製品のアップグレード時に上書きされません。このファイルで定義される設定は、`style.css` および `lockhart.css` の設定より優先されます。すべてのカスタマイズ定義を customStyle.css に含めるようにしてください。

JSP ファイル

ヘッダーのデフォルト設定を格納する JSP ファイルに、userHeader.jsp、bodyEnd.jsp、および bodyStart.jsp があります。これらのファイルは編集しないでください。製品アップグレード時にカスタマイズ内容を保持するには、このファイルに代わりに、「アーキテクチャー機能」の一覧に示されている JSP のみを編集してください。

WPMessages_ja.properties ファイル

$WSHOME/WEB-INF/lib/idm_l10n_ja_JP.jar ファイルにはメッセージカタログのエントリが格納されています。JAR ファイルから WPMessages_ja.properties ファイルを抽出し、そのファイル上でエントリを編集できます。

ヘッダーのカスタマイズ

カスタマイズのタスクは両方のインタフェースで同じですが、編集する必要のあるファイルは異なります。

注	指定されたファイル以外の .jsp ファイルを編集しないでください。編集する必要がある場合は、.jsp ファイルのコピー、編集、名前変更を行う前に、まずそのファイルを安全な場所にバックアップしてください。

ヘッダー外観を変更する

最も一般的なラベリングのタスクでは、次のことを行います。

- ページのヘッダーセクションで参照される画像を、デフォルトの Sun ロゴから会社ごとの標準画像に変更します。customStyle.css を編集することにより、画像を置換または削除します。
- Identity Manager ロゴを非表示にします。
- Look & Feel に関する社内ガイドライン (特に境界、ヘッダー、背景色) を使用します。
- 「次のユーザーとしてログイン」テキストを変更します。これは .jsp ファイルからの変更はできません。カスタムメッセージカタログを編集することができます。123 ページの「Identity Manager ユーザーインタフェースのホームページに表示されるデフォルト情報を変更する」を参照してください。

Identity Manager ページをカスタマイズする

一般的なカスタマイズには、次のものがあります。

- ホームページのカスタマイズ
- Identity Manager ユーザーインタフェースのホームページのデフォルト文字列の変更

ホームページをカスタマイズする

- クイックリンクのリストの追加
- デフォルトのログインテキストの変更
- ページタイトルとサブタイトルの変更
- ブラウザのタイトルバーのカスタマイズ

クイックリンクのリストを追加する

ホームページへの一般的なカスタマイズでは、環境内でユーザーが頻繁にアクセスするタスクまたはリソースに、カスタマイズしたリンクリストを追加します。これらのクイックリンクには、製品インタフェースからよく使う項目へのショートカットとしての機能があります。

クイックリンクのリストを追加するには、次の手順を実行します。

1. `index_quickLinks.jsp` でリストにリンクを追加します。
2. 前後の <%-- および --%> タグを削除して、list セクションのコメントを解除します。
3. ファイルを保存します。アプリケーションサーバーを再起動する必要はありません。

デフォルトの「次のユーザーとしてログイン」テキストを変更する

1. 次の XML ファイルをインポートします。

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE Configuration PUBLIC 'waveset.dtd' 'waveset.dtd'>
<Configuration name='AltMsgCatalog'>
  <Extension>
    <CustomCatalog id='AltMsgCatalog' enabled='true'>
      <MessageSet language='en' country='US'>
       <Msg id='UI_BROWSER_TITLE_PROD_NAME_OVERRIDE'>Override
Name</Msg>
      </MessageSet>
    </CustomCatalog>
  </Extension>
</Configuration>
```

2. System Configuration オブジェクトの `<Configuration><Extension><Object>` 要素内に次の行を追加します。

   ```
   <Attribute name='customMessageCatalog' value='AltMsgCatalog'/>
   ```

3. 次の行を追加します。

   ```
   <msg id='UI_NAV_FOOT_LOG'>mytext{0}</msg>
   ```

4. 変更を保存し、アプリケーションサーバーを再起動します。

ページタイトルとサブタイトルを変更する

ログインページのデフォルトのタイトル、サブタイトル、「ようこそ」メッセージを変更するには、カスタムメッセージカタログの次のエントリを変更します。

- `UI_LOGIN_TITLE`
- `UI_LOGIN_TITLE_TO_RESOURCE`
- `UI_LOGIN_WELCOME2`

このテキストを変更するには、120 ページの「デフォルトの「次のユーザーとしてログイン」テキストを変更する」で詳しく説明している、`WPMessages_ja.properties` ファイルの抽出および編集の手順に従います。

コード例 7-1　　デフォルトのメッセージカタログ設定

```
UI_LOGIN_IN_PROGRESS_TITLE= ログイン （進行中）
UI_LOGIN_CHALLENGE={0} パスワードを入力してください
要求したアクションを完了するには、[PRODUCT_NAME] への
ログイン時に指定したパスワードを入力する必要があります。
UI_LOGIN_TITLE_LONG=[PRODUCT_NAME] ログイン
UI_LOGIN_WELCOME=Sun Java&#8482; System [PRODUCT_NAME] システムへようこそ。
要求される情報を入力し、<b> ログイン </b> をクリックしてください。
UI_LOGIN_WELCOME2=Sun Java&#8482; System [PRODUCT_NAME] システムへようこそ。
ユーザー ID とパスワードを入力し、「<b> ログイン </b>」をクリックしてください。パスワードを思
い出せないときは、「<b> パスワードをお忘れですか ?</b>」をクリックしてください。
UI_LOGIN_TITLE= ログイン
UI_LOGIN_TITLE_TO_RESOURCE=<b>{0}</b> へのログイン
```

ログインページの背景画像を変更する

次のように customStyle.css を編集することにより、背景画像を変更できます。

```
div#loginFormDiv {
   background:
url(.../images/other/login-backimage2.jpg)
no-repeat;
     margin-left: -10px;
     padding: 0px 0px 280px;
    height: 435px;
```

ブラウザのタイトルバーをカスタマイズする

ブラウザのタイトルバーの製品名文字列を、任意のローカライズ可能な文字列で置き換えることができるようになりました。

1. 次の XML ファイルをインポートします。

コード例 7-2 インポートする XML

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE Configuration PUBLIC 'waveset.dtd' 'waveset.dtd'>
<Configuration name='AltMsgCatalog'>
  <Extension>
    <CustomCatalog id='AltMsgCatalog' enabled='true'>
      <MessageSet language='en' country='US'>
       <Msg id='UI_BROWSER_TITLE_PROD_NAME_OVERRIDE'>Override Name</Msg>
      </MessageSet>
    </CustomCatalog>
  </Extension>
</Configuration>
```

2. Identity Manager IDE を使用して、System Configuration オブジェクトを編集のためにロードします。`<Configuration><Extension><Object>` 要素内に次の行を追加します。

   ```
   <Attribute name='customMessageCatalog' value='AltMsgCatalog'/>
   ```

3. また、System Configuration オブジェクトで、`ui.web.browserTitleProdNameOverride` を **`true`** に変更する必要があります。
4. System Configuration オブジェクトの変更を保存し、アプリケーションサーバーを再起動します。

Identity Manager ユーザーインタフェースのホームページに表示されるデフォルト情報を変更する

Identity Manager ユーザーインタフェースのホームページには、ログイン中のアカウントについての基本情報を表示する「ダッシュボード」領域があります。表示される情報には、次の表に示すデフォルト文字列が含まれます。

すべての属性は ui.web.user.dashboard オブジェクトに属します。

表 7-2 ui.web.user.dashboard オブジェクトのデフォルト設定

設定オブジェクトのデフォルト設定	説明
displayLoginFailures	アカウントのアカウントポリシーで最大ログイン試行回数が設定されている場合に、ログイン試行でのパスワード入力または秘密の質問への回答の失敗回数を表示します。
displayPasswordExpirationWarning	パスワードポリシーがアカウントに適用される場合に、パスワードの有効期限に関連するメッセージを表示します。
displayApprovals displayAttestationReviews displayOtherWorkItems displayRemediations	すべてのユーザーを対象に、「承認」、「アテステーション」、「是正」、および「その他」の各作業項目タイプの表示を有効にします。 **注 :** 設定オブジェクトで特定のタイプに対する設定が true であっても、アカウントユーザーに許可されたアクセス権によっては、インタフェース文字列が表示されない場合があります。
displayRequests	アカウントのロール、グループ、またはリソースの更新要求のうち未処理のものの数を示します。
displayDelegations	ユーザーが承認の委任を定義したことを示す文字列を表示します。「有効」または「無効」のオプションがあります。

デフォルトでは、これらの設定オブジェクトの値は true に設定され、これらの文字列は Identity Manager ユーザーインタフェースに表示されます。任意の文字列を非表示にするには、System Configuration オブジェクトでその文字列を `false` に設定します。

System Configuration オブジェクトは Identity Manager IDE を通じて編集できます。Identity Manager IDE の使用方法については、『Identity Manager 配備ツール』の「Identity Manager IDE の使用」を参照してください。System Configuration オブジェクトについての全般的な情報は、付録 A 「設定オブジェクトを編集する」を参照してください。

ユーザーインタフェースのナビゲーションメニューの外観を変更する

ユーザーインタフェースのナビゲーションメニューに関して、System Configuration オブジェクトで次の 2 つの設定を行う必要があります。

- `ui.web.user.showMenu` を `true` に設定する必要があります。
- `ui.web.user.menuLayout` を `horizontal` (デフォルト) に設定すると、ナビゲーションメニューの水平タブが描画されます。値を `vertical` に設定すると、垂直ツリーメニュー形式でメニューが描画されます。

フォント特性を変更する

表示属性では一般に、次の基本フォント表示特性を指定します。

表 7-3 フォント関連の表示特性

表示属性	説明
family	Helvetica、Arial など
size	ポイントサイズで指定 (例 : 14 ポイント)
weight	未指定の場合は通常の書体。指定する場合、bold など
color	通常は black を指定

一部のコンポーネントは、追加の特性によってより詳細な定義が可能です。たとえば、ボタンの背景色を定義することができます。テキストやボタンラベルの配置も特性の一つです。

フォント特性を編集する

編集を行うには、`styles.css` からコピーして `customStyle.css` にペーストします。その後、選択した設定を `customStyle.css` で修正します。

例

次のエントリは、各ページタイトルのデフォルト設定を表します。

```
.title {
    font-family: Arial, Helvetica, sans-serif;
    font-size: 16pt;
    font-weight: bold;
    color: C;
```

ラベリング演習のサンプル

次の例は、Identity Manager のロゴを非表示にし、ページのマストヘッドでカスタム画像を参照する方法を示します。sample/rebranding ディレクトリに収められたサンプルファイルを参照してください。

- 製品名の変更
- マストヘッドの色の変更
- ナビゲーションタブの色の変更
- よく使われるページでの Identity Manager の動作の変更

Identity Manager ロゴのカスタムロゴへの置き換え

管理者インタフェースまたはユーザーインタフェースのロゴを変更するには、styles/style.css の次の部分を customStyle.css にコピーし、Identity Manager のロゴ画像を独自の画像ファイルに置き換えます。

```
td.MstTdLogo {
    width: 31px;
    height: 55px;
    background: url(../images/other/logo.jpg) no-repeat 5px;
}

td.MstTdTtlProdNam {
    background: url(../images/other/xyz_masthead.jpg) no-repeat 10px
30px;
    padding:10px 10px 0px 10px;
    vertical-align:top;
```

```
    white-space:nowrap;
    height: 75px;
    width: 350px;
}
```

注　最適な結果を得るには、ロゴ画像の高さを 50 ～ 60 ピクセルで作成します。

マストヘッドの外観を変更する

Identity Manager の外観を変更するには、次の処理を実行します。

1. `styles/customStyle.css` ファイルを編集します。
2. `styles.css` の次のセクションを `customStyle.css` にコピーし、必要に応じて変更します。

コード例 7-3 styles/customStyle.css ファイル

```
MstDiv
{background-image:url(../images/other/dot.gif);background-repeat:repeat-x;
background-position:left top;background-color:#000033}
.MstTblEnd {background: url(../images/other/dot.gif);background-color:
#666;height: 1px;}
td.MstTblEndImg {
    background-color: #666;
    height: 1px;
    background: url(../images/other/dot.gif);
    font-size:1px;
}
td.MstTdLogo {
    width: 31px;
    height: 55px;
    background: url(../images/other/logo.jpg) no-repeat 5px;
}
td.MstTdSep {
    width: 1px;
    height: 61px;
    background: url(../images/other/dot.gif) no-repeat center;
}
td.MstTdTtlProdNam {
  background: url(../images/other/xyz_masthead.jpg) no-repeat 10px 30px;
  padding:10px 10px 0px 10px;
    vertical-align:top;
    white-space:nowrap;
    height: 75px;
    width: 350px;
}
```

ナビゲーションタブを変更する

Identity Manager ユーザーインタフェースのナビゲーションバーをカスタマイズする

Identity Manager のユーザーインタフェースは、ナビゲーションバーを含む 2 番目の XPRESS フォームを実装しています。これは、描画されるページに 2 つの <FORM> タグがあり、各タグで name 属性の設定が次のように異なることを意味します。

```
<form name="endUserNavigation">
```

および

```
<form name="mainform">
```

Identity Manager ユーザーインタフェースのナビゲーションバーに JavaScript コードを挿入するときは、必ず正しいフォームを参照するようにしてください。そのためには、name 属性を使用して、どちらの <FORM> タグを参照するか (document.mainform または document.endUserNavigation のどちらか) を指定します。

ナビゲーションリンクをカスタマイズする

ページの上部を横切るナビゲーションリンクをカスタマイズするには、次のコードをコピーして編集します。背景色を適切な色に変更します。

コード例 7-4 ナビゲーションリンクをカスタマイズする

```
* LEVEL 1 TABS */
.TabLvl1Div {
    background-image:url(../images/other/dot.gif);
    background-repeat:repeat-x;
    background-position:left bottom;
    background-color: #333366;
    padding:6px 10px 0px;
}
a.TabLvl1Lnk:link, a.TabLvl1Lnk:visited  {
    display:block;
    padding:4px 10px 3px;
    font: bold 0.95em sans-serif;
    color:#FFF;
    text-decoration:none;
    text-align:center;
}
table.TabLvl1Tbl td {
    background-image:url(../images/other/dot.gif);
    background-repeat:repeat-x;
    background-position:left top;
    background-color: #666699;
    border:solid 1px #aba1b5;
}
```

コード例 7-4　　ナビゲーションリンクをカスタマイズする (続き)

```
table.TabLvl1Tbl td.TabLvl1TblSelTd {
    background-color:#9999CC;
    background-image:url(../images/other/dot.gif);
    background-repeat:repeat-x;
    background-position:left bottom;
    border-bottom:none;
```

タブパネルのタブを変更する

コード例 7-5　　Identity Manager タブパネルのタブの変更

```
.TabLvl2Div {
    background-image:url(../images/other/dot.gif);
    background-repeat:repeat-x;
    background-position:left bottom;
    background-color:#9999CC;
    padding:6px 0px 0px 10px
}
a.TabLvl2Lnk:link, a.TabLvl2Lnk:visited{
    display:block;
    padding:3px 6px 2px;
    font: 0.8em sans-serif;
    color:#333;
    text-decoration:none;
    text-align:center;
}
table.TabLvl2Tbl div.TabLvl2SelTxt {
    display:block;
    padding:3px 6px 2px;
    font: 0.8em sans-serif;
    color:#333;
    font-weight:normal;
    text-align:center;
}
table.TabLvl2Tbl td {
    background-image:url(../images/other/dot.gif);
    background-repeat:repeat-x;
    background-position:left top;
    background-color:#CCCCFF;
    border:solid 1px #aba1b5;
}
table.TabLvl2Tbl td.TabLvl2TblSelTd {
    border-bottom:none;
    background-image:url(../images/other/dot.gif);
    background-repeat:repeat-x;
    background-position:left bottom;
    background-color:#FFF;
    border-left:solid 1px #aba1b5;
```

コード例 7-5　Identity Manager タブパネルのタブの変更 (続き)

```
    border-right:solid 1px #aba1b5;
    border-top:solid 1px #aba1b5;

table.Tab2TblNew td
{background-image:url(../images/other/dot.gif);background-repeat:repeat-x;back
ground-position:left top;background-color:#CCCCFF;border:solid 1px #8f989f}
table.Tab2TblNew td.Tab2TblSelTd
{border-bottom:none;background-image:url(../images/other/dot.gif);background-r
epeat:repeat-x;background-position:left
bottom;background-color:#FFF;border-left:solid 1px #8f989f;border-right:solid
1px #8f989f;border-top:solid 1px #8f989f}
```

ソートテーブルのヘッダーを変更する

```
.tablehdr {
    background-image:url(../images/other/dot.gif);
    background-repeat:repeat-x;
    background-position:left bottom;
    background-color:#9999CC;
}
```

ユーザー / リソーステーブルのコンポーネントを変更する

コード例 7-6　ユーザー / リソーステーブルのコンポーネントの変更

```
.tablesorthdr {
    /*background-color: #BEC7CC;*/
    background-image:url(../images/other/dot.gif);
    background-repeat:repeat-x;
    background-position:left bottom;
    background-color:#CCCCFF;
    border:solid 1px #aba1b5;
}

.treeContentLayout {
    background-color: #9999CC;
}
```

コード例 7-6 ユーザー / リソーステーブルのコンポーネントの変更 (続き)

```
.treeBaseRow {
    padding-top: 0px;
    padding-left: 10px;
    padding-right: 10px;
    padding-bottom: 0px;
    background-color: #fff;
    border-left: solid 1px #8F989F;
    border-right: solid 1px #8F989F;
    border-bottom: solid 1px #8F989F;
}

.treeButtonCell {
    background-image:url(../images/other/dot.gif);
    background-color:#666699;
    color: #fff;
}

}

.treeMastHeadRow {
    background-color:#333366;
}

.treeMastHeadRow {
    background-color:#333366;
}

.treeMastHeadText {
    background-color:#333366;
    background-image:url(../images/other/dot.gif);
}
```

同じ手順に従うことにより、その他のさまざまなオプション (テキストのスタイルやサイズ、配置、その他のオブジェクトの色や設定など) をカスタマイズできます。

注 サーバーまたはブラウザを再起動せずに変更を確認するには、ページ上で Ctrl キーを押しながら「更新」を実行します。

よく使われるページでの Identity Manager の動作を変更する

System Configuration オブジェクトの設定を変更することにより、よく使われるページでの Identity Manager の動作をカスタマイズできます。

System Configuration オブジェクトを使用したカスタマイズ

System Configuration オブジェクトを編集することにより、ユーザーインタフェースまたは管理者インタフェースの、よく変更される多くのプロパティーをカスタマイズすることができます。製品インタフェースを制御するのは、属性 `<Attribute name='ui'>` とそのサブオブジェクトです。この属性の下位属性を変更することにより、Identity Manager の動作を変えることができます。

その他の変更 : ファイルの admin セクション

System Configuration オブジェクトファイルの admin セクションには、管理者インタフェースに関連するさまざまな属性が含まれます。

- 管理者ログインページの「**パスワードをお忘れですか ?**」ボタンを無効にするには、`disableForgotPassword` を **true** に設定します。
- `supressHostName` を **true** に設定すると、タスク詳細情報ページでプロセスのホスト名が非表示になります。

コード例 7-7 ファイルの admin セクションの変更

```
<Attribute name='admin'>
   <Object>
      <Attribute name='disableForgotPassword'>
         <Boolean>false</Boolean>
      </Attribute>
      <Attribute name='workflowResults'>
         <Object>
            <Attribute name='suppressHostName'>
               <Boolean>false</Boolean>
            </Attribute>
         </Object>
      </Attribute>
   </Object>
</Attribute>
```

その他の変更 : ファイルの user セクション

System Configuration オブジェクトファイルの user セクションには、ユーザーインタフェース関連のオプションがあります。

- `disableForgotPassword` を `true` に設定すると、「**パスワードをお忘れですか ?**」ボタンが無効になります。

`workflowResults` 属性には、管理者でないユーザーのワークフローの表示をカスタマイズするための属性が、次の表に示すように含まれます。

表 7-4 管理者でないユーザーのワークフローをカスタマイズするための属性

属性	説明
`anonSuppressReports`	匿名ユーザーのワークフローステータスページ (`anonProcessStatus.jsp`) にワークフロー図を表示するかどうかを制御します。
`suppressHostName`	エンドユーザーのワークフローステータスページ (`processStatus.jsp`) にホスト名の情報を含めるかどうかを制御します。
`suppressReports`	匿名でないすべてのユーザーに対してワークフロー図を表示するかどうかを制御します (`processStatus.jsp`)。

コード例 7-8 管理者でないユーザーのワークフローのカスタマイズ

```
<Attribute name='user'>
   <Object>
     <Attribute name='disableForgotPassword'>
       <Boolean>false</Boolean>
     </Attribute>
     <Attribute name='workflowResults'>
       <Object>
          <Attribute name='anonSuppressReports'>
             <Boolean>false</Boolean>
          </Attribute>
          <Attribute name='suppressHostName'>
             <Boolean>false</Boolean>
          </Attribute>
          <Attribute name='suppressReports'>
             <Boolean>false</Boolean>
         </Attribute>
        </Object>
      </Attribute>
   </Object>
</Attribute>
```

- パスワードおよび秘密の質問の回答を非表示にするには、`obfuscateAnswers` を `true` に設定します。このように設定すると、管理者インタフェースとユーザーインタフェースの両方で、入力したパスワードや回答がアスタリスクでマスクされます。

  ```
  <Attribute name="obfuscateAnswers">
     <Boolean>true</Boolean>
  </Attribute>
  ```

第 8 章

メッセージカタログをカスタマイズする

メッセージカタログエントリを追加したり、システムから提供されるエントリを修正したりするために、カスタマイズしたメッセージカタログを作成できます。

カスタムメッセージカタログの利点

カスタムメッセージカタログには、次の利点があります。

- クラスタ環境での保守作業の軽減。個別のメッセージカタログを保守すると、複数の `WPMessages.properties` ファイルを編集する必要がなくなります。
- バージョン制御の簡素化。カスタマイズしたメッセージが 1 か所にあると、変更の追跡や改訂版のバックアップが簡単になります。
- 製品メッセージカタログへのアップグレードは、カスタマイズしたエントリに加えられた変更で失敗することはありません。

Identity Manager がメッセージカタログエントリを取得する方法

Identity Manager は、次の順序でメッセージカタログエントリを取得します。

- ユーザー定義のメッセージカタログ (許容されるユーザー定義メッセージカタログは 1 つのみ)
- システム定義の `defaultCustomCatalog` メッセージカタログ
- `config/WPMessages.properties` ファイル
- `idmcommon.jar` ファイル内の `WPMessages.properties` ファイル。

メッセージカタログの形式

WPMessages.properties ファイルでは、エントリは *KeyName=MessageText* の形式で定義されます。カスタマイズしたメッセージカタログでは、各エントリは個別の Msg 要素で指定されます。*KeyName* は id 属性で指定されます。*MessageText* は <Msg> タグと </Msg> タグの間にあるテキストです。次の例は、メッセージカタログエントリを示しています。

```
<Msg id='UI_REMEMBER_PASSWORD'>Remember to set your password.</Msg>
```

メッセージテキストに HTML タグを含めてテキストのレンダリング方法を制御できますが、これはお勧めできません。HTML タグを使用する必要がある場合は、< や > などの記号の代わりに < や > などのコードを使用してください。

メッセージテキストには、文字列が表示されるときに Identity Manager によって文字列に挿入されるデータの変数を含めることもできます。次の例は、AR_CORRELATED_USER キーのデフォルトメッセージです。

```
Correlated account with user {0}.
```

描画されたものは次のようになります。

```
Correlated account with user jdoe.
```

カスタマイズしたメッセージカタログを作成する

次の手順では、ユーザー定義のメッセージカタログを作成する方法について説明します。

1. デフォルトのメッセージカタログエントリを上書きする場合は、WPMessages.properties ファイルで該当するエラーメッセージキーの場所を特定します。これらのキーは、カスタマイズしたメッセージカタログで指定する必要があります。

   新しいメッセージを作成する場合は、これらのキーが WPMessages.properties ファイルにないことを確認します

2. 次の構造で XML ファイルまたはブロックを作成します。

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE Configuration PUBLIC 'waveset.dtd' 'waveset.dtd'>
<Configuration name='CatalogName'>
  <Extension>
    <CustomCatalog id='CatalogName' enabled='true'>
      <MessageSet language='en' country='US'>
        <Msg id='KeyName'>MessageText</Msg>
        <Msg id='KeyName'>MessageText</Msg>
        ...
      </MessageSet>
    </CustomCatalog>
  </Extension>
</Configuration>
```

各表記の意味は次のとおりです。

CatalogName は、メッセージカタログの名前です。この値は、System Configuration オブジェクトのカタログの定義にも使用されます。

KeyName はメッセージキー名です。

MessageText は、グラフィカルユーザーインタフェースに表示される文字列です。このテキストには、HTML タグと変数を含めることができます。

en_US 以外のロケールをサポートする場合は、`language` 属性と `country` 属性を変更します。複数のロケールをサポートする場合は、ロケールごとに個別の `MessageSet` 要素を作成します。

参考例については、「例」の項を参照してください。

3. ファイルまたはブロックを Identity Manager にインポートします。
4. System Configuration オブジェクトを読み込み、次の行を `<Configuration><Extension><Object>` 要素内に追加します。

```
<Attribute name='customMessageCatalog' value='CatalogName'/>
```

5. 変更を System Configuration オブジェクトに保存します。
6. アプリケーションサーバーを再起動します。これで、新しいメッセージカタログエントリがシステムで利用できるようになりました。

例

次の例では、myCustomCatalog という名前のカスタマイズしたメッセージカタログを作成します。ここでは、「**交換ファイルのインポート**」サブタブのラベルとヘルプテキストを置き換えます。

コード例 8-1　カスタマイズした myCustomCatalog メッセージカタログを作成する

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE Configuration PUBLIC 'waveset.dtd' 'waveset.dtd'>
<Configuration name='myCustomCatalog'>
  <Extension>
    <CustomCatalog id='myCustomCatalog' enabled='true'>
      <MessageSet language='en' country='US'>
        <Msg id='UI_SUBNAV_CONFIGURE_IMPORT_EXCHANGE'>XML ファイル
のインポート </Msg>
        <Msg id='UI_SUBNAV_CONFIGURE_IMPORT_EXCHANGE_HELP'> 指定し
た XML ファイルをインポートします。< /Msg>
      </MessageSet>
    </CustomCatalog>
  </Extension>
</Configuration>
```

設定オブジェクトを編集する

この章では、設定オブジェクトと呼ばれる Identity Manager コンポーネントについて説明します。設定オブジェクトは、持続的なカスタマイズを Identity Manager に格納します。設定オブジェクトはキャッシュされるオブジェクトタイプなので、すべての設定オブジェクトはメモリーに格納され、その後設定オブジェクトが変更されるごとにそのキャッシュはフラッシュされます。概して、`User` オブジェクトおよび `TaskInstances` を除いて、Identity Manager リポジトリ内の大部分のオブジェクトは設定オブジェクトです。

設定オブジェクトプロパティーの編集は、Identity Manager の動作に持続的な変更を実装する方法の 1 つです。この付録では、設定オブジェクトの参照および編集方法について説明します。情報は次のように構成されています。

データ記憶領域

Sun Identity Manager リポジトリでは、次のテーブルに設定オブジェクトデータが格納されます。

注	これはすべてのテーブルが網羅された総合的なリストではありません。関連するテーブルのみが列挙されています。

- **object** - Identity Manager アプリケーションの設定オブジェクトの大部分を格納します。次のオブジェクトが含まれています。
  - `SystemConfiguration` オブジェクト
  - `TaskDefinition` オブジェクト
  - `WorkItem` オブジェクト
  - `Form` オブジェクト
  - `Resource` オブジェクト
- **task** - `TaskInstances` および `WorkItems` (つまり、ワークフローの静的でないインスタンス) が格納されます。
- **org** - すべての Identity Manager 組織情報が格納されます。
- **userobj** - すべての Identity Manager エンタープライズアイデンティティーが格納されます。これらのアイデンティティーは、単に管理対象システムのアカウント用のコンテナです。
- **account** - 管理対象システムで識別されるアカウントが格納されます。

注	これは、「アカウントインデックスを構築する」ときに参照されるテーブルです。調整またはアカウントのリンク後は、このテーブルには管理対象システムごとにアカウントのセット番号があります。

- **log** - すべての監査イベントおよびログタイプ情報が格納されます。

Identity Manager リポジトリで理解すべき主要な概念は、すべてのデータは 2 つの方法で格納され、各テーブルにはオブジェクトのクエリーに使用されるインデックスおよびキー付きの列があり、各テーブルにはオブジェクト (通常は、データベースエンジンに応じて、`BLOB` または `MEDIUM TEXT` データ型) の ASCII 表現全体の格納に使用される XML 列があります。すべての Identity Manager オブジェクトはリポジトリへの格納のために Java オブジェクトから ASCII XML に非直列化されるため、Identity Manager ではこの方法でデータが格納されます。

アプリケーションは、高レベルでは、インデックス付きの列でクエリーを実行し、XML ASCII テキストを戻したあとで、XML を Java オブジェクトに直列化します。通常、これらのオブジェクトはビュー (UserView、PasswordView など) を使用してアクセスできます。

オブジェクトの命名規則

Identity Manager オブジェクト名には、次の文字を使用しないでください。

文字	説明
'	単一引用符
=	等号
.	ピリオド
|	垂直バー
[	左角括弧
]	右角括弧
,	コンマ
:	コロン
$	ドル記号
\	バックスラッシュ
"	二重引用符

その他、次のような特殊文字もエラーの原因となるので、オブジェクト名の中で使用しないでください。

文字	説明
_	下線
%	パーセント
*	アスタリスク
#	番号記号
^	キャレット

設定オブジェクトを表示および編集する

設定オブジェクトプロパティーの編集は、Identity Manager の動作に持続的な変更を実装する方法の 1 つです。

Sun Identity Manager 統合開発環境 (Identity Manager IDE) を使用すると、配備用に Identity Manager オブジェクトを表示および編集できます。Identity Manager 統合開発環境 (Identity Manager IDE) のインストールと設定の説明書は、https://identitymanageride.dev.java.net から入手できます。

このセクションでは、次の設定オブジェクトの参照および編集方法について説明します。

- IDM Schema Configuration オブジェクト
- UserUIConfig オブジェクト
- RepositoryConfiguration オブジェクト
- WorkItemTypes Configuration オブジェクト
- SystemConfiguration オブジェクト
- Role Configuration オブジェクト
- End User Tasks オブジェクト

注

オブジェクトの authType によって、設定オブジェクトを参照または編集するユーザーが決まります。

オブジェクトに認可タイプを割り当てるには、PersistentObject クラスに定義された新しいフィールドを設定します。Java API から次のメソッドを使用すると、認可タイプにアクセスできます。

```
public void setAuthType(String name);

public String getAuthType();
```

オブジェクトの XML では、root 要素に authType 属性を設定できます。例:

```
<TaskDefinition name='Request More Space'
authType='EndUserTask'

    executor='com.waveset.workflow.WorkflowExecutor'
...>

    ...

</TaskDefinition>
```

注 Type.USER のインライン属性またはクエリー可能な属性を変更する場合、すべての User オブジェクトを更新する必要があります。

詳細については、155 ページの「User オブジェクトを更新する」を参照してください。

IDM Schema Configuration オブジェクト

IDM Schema Configuration オブジェクトの User および Role 拡張属性、クエリー可能な属性、および概要の属性を設定します。

注 IDM ObjectClass Configuration オブジェクトで提供されるスキーマカスタマイズは、サーバーの起動時に読み込まれます。スキーマを変更するたびに、変更を読み込むためにサーバーを再起動する必要があります。

Identity Manager では、スキーマの読み込み時に発生した問題がシステムログメッセージに記録されます。これらのメッセージを参照するには、次の方法のいずれかを使用します。

- lh syslog コマンドを実行する
- IDM 管理者インタフェース (「レポート」タブ) から「Recent System Messages」レポートを実行する

スキーマのサンプルは、サンプルディレクトリの schema.xml ファイルにあります。

配備中に拡張属性を複数のオブジェクトタイプに追加するには、IDM Schema Configuration 設定オブジェクトを編集します。具体的には、次の処理を実行できます。

- ユーザー、ロール、ビジネスロール、IT ロール、アプリケーションロール、アセットロール、および任意のカスタムロールに対して、拡張属性、クエリー可能な属性、および概要の属性を設定する
- 拡張属性および組み込み属性をクエリー可能または概要としてマークする

注 IDM Schema Configuration オブジェクトは、IDMSchemaConfig authType で保護されます。

ユーザーまたはロールの Identity Manager スキーマを参照または編集する必要のある管理者は、IDMSchemaConfig AdminGroup (機能) を割り当てる必要があります。コンフィギュレータのユーザーには、デフォルトでこの AdminGroup が割り当てられています。

拡張属性をオブジェクトに追加する

拡張属性を追加するには、IDMAttributeConfiguration を使用して属性を定義する必要があります (属性が組み込み属性でない場合)。

IDMAttributeConfigurations には名前と構文が必要です。有効な構文オプションは BOOLEAN、DATE、INT、または STRING です。オプションで、IDMAttributeConfiguration を使用すると、属性が複数の値を持つかどうかを指定したり、(現在未使用の) 表示名と説明を指定したりできます。

拡張属性を追加したり、または属性 (拡張か組み込みのいずれか) をクエリー可能または概要としてマークするには、適切な IDMObjectClassConfiguration (User など) に IDMObjectClassAttributeConfiguration を指定します。既存の (同一の設定オブジェクトに組み込まれた、または設定された) IDMAttributeConfiguration と一致する名前を指定する必要があります。また、IDMObjectClassAttributeConfiguration をクエリー可能または概要としてマークすることもできます。

次の例では、firstname, lastname、および fullname は拡張属性です。firstname および lastname ユーザー属性はクエリー可能で概要の属性ですが、fullname は違います。

図 A-1 拡張属性の例

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE Waveset PUBLIC 'waveset.dtd' 'waveset.dtd'>
<Waveset>
   <Configuration name="IDM Schema Configuration"
                  id='#ID#Configuration:IDM_Schema_Configuration'
                  authType='IDMSchemaConfig'>
       <IDMSchemaConfiguration>
           <IDMAttributeConfigurations>
...
               <IDMAttributeConfiguration name='firstname'
                                          description='User's first name'
                                          syntax='STRING'/>
               <IDMAttributeConfiguration name='lastname'
                                          description='User's last name'
                                          syntax='STRING'/>
               <IDMAttributeConfiguration name='fullname'
                                          description='User's full name'
                                          syntax='STRING'/>
...
           </IDMAttributeConfigurations>
           <IDMObjectClassConfigurations>
...
               <IDMObjectClassConfiguration name='User'
                                            extends='Principal'>
...
                   <IDMObjectClassAttributeConfiguration name='firstname'
                                                         queryable='true'
```

図 A-1 拡張属性の例 (続き)

```
                                                    summary='true'/>Configuration
list/read/write.
                  <IDMObjectClassAttributeConfiguration name='lastname'
                                                    queryable='true'
                                                    summary='true'/>
                  <IDMObjectClassAttributeConfiguration name='fullname'/>
...
              </IDMObjectClassConfiguration>
           </IDMObjectClassConfigurations>
       </IDMSchemaConfiguration>
   </Configuration>
</Waveset>
```

注 今後のリリースの Sun Identity Manager の新しいコア属性との競合を避けるために、拡張属性に配備固有の接頭辞を付けます。

たとえば、employeeNumber を記録するために拡張属性を User に追加するには、acme_employeeNumber などの会社に関連付けられた接頭辞を選択します。将来の Identity Manager のリリースに employeeNumber という組み込みユーザー属性が組み込まれても、2 つの属性は区別されます。それ以外の場合は、組み込み属性が優先されます。

Role ObjectClass を拡張する

IDMObjectClassConfiguration を使用してロールを拡張できます。次の組み込みロール拡張はすべて Role objectclass を拡張します。

- BusinessRole
- ITRole
- AssetRole
- ApplicationRole

拡張属性を AssetRole などの特定のロール拡張に追加するには、IDMObjectClassAttributeConfiguration を AssetRole IDMObjectClassConfiguration に追加します。拡張属性を全種類のロールに追加するには、IDMObjectClassAttributeConfiguration を Role IDMObjectClassConfiguration に追加します。この属性は、Role のすべての拡張で継承されます。

Role のカスタム拡張または Role の任意の拡張を定義できます。
たとえば、AssetRole のカスタム拡張を追加するには、次の例に示すように、新しいロールに対して新しい IDMObjectClassConfiguration (IDM Schema Configuration 内) を定義し、extends フィールドを使用して親ロールを指定します。

```
<IDMObjectClassConfiguration name='MyAssetRole'
                             extends='AssetRole'
                             description='My Asset Role
Description'/>
```

新しい Role objectclass を追加する場合は、新しい Role タイプを Role Configuration オブジェクトに追加する必要があります。さらに、新しい Role タイプの名前は、新しい Role objectclass の名前と一致する必要があります。詳細については、151 ページの「Role Configuration オブジェクト」を参照してください。

UserUIConfig オブジェクト

注 ここで、UserUIConfig オブジェクトではなく、スキーマ設定でユーザー (WSUser) の拡張属性、クエリー可能な属性、および概要の属性を設定します。詳細については、「IDM Schema Configuration オブジェクト」を参照してください。

SummaryAttrRoleCountLimit

ユーザーの概要の属性文字列に表示されるロール数を制御します。この数を制御するには、ここに値を指定します。このオブジェクトに値を指定しない場合は、Identity Manager では最大で 3 つのロールが表示されます。

RepositoryConfiguration オブジェクト

RepositoryConfiguration オブジェクトには、Identity Manager リポジトリの動作を制御する設定が含まれています。最上位の <RepositoryConfiguration> 要素の各 XML 属性では、リポジトリ動作の一部が設定されます。

たとえば、次の行は、デフォルトでリポジトリロックの期限が切れる時間を 5 分に指定します。

```
<RepositoryConfiguration ... lockTimeoutMillis='300000' ... >
```

警告 RepositoryConfiguration の設定は、その効果について理解しているのでない限り、どれも変更しないでください。

RepositoryConfiguration オブジェクトには、User オブジェクトに固有の設定もいくつか含まれています。たとえば、User オブジェクトの TypeDataStore 要素は、User オブジェクトのインライン属性を指定します。

インライン属性は単一値の属性であり、リポジトリによってタイプごとに主要なオブジェクトテーブルに直接格納されます (この場合、USEROBJ テーブルの attr1 ～ attr5)。大部分の属性値は、USERATTR テーブル (属性ごとに個別の結合が必要) に格納されます。属性をインラインで指定すると、その属性を使用するクエリーのパフォーマンスが改善されます。

サンプルの RepositoryConfiguration オブジェクトには、次のように、User オブジェクトのデフォルトインライン属性が指定されます。

```
<TypeDataStore typeName='User' ... attr1='MemberObjectGroups' \
  attr2='lastname' attr3='firstname' attr4='' attr5='' />
```

attr1='MemberObjectGroups' に設定されている attr1 の値は変更しないでください。ただし、クエリー可能な単一値である任意の属性名を残りのインライン列 (attr2 ～ attr5) の値として指定できます。

注 Type.USER のインライン属性を変更する場合、すべての User オブジェクトを更新する必要があります。

詳細については、155 ページの「User オブジェクトを更新する」を参照してください。

注 RepositoryConfiguration オブジェクトへの変更は、各 Identity Manager サーバーを再起動するまで有効になりません。Identity Manager サーバーを再起動すると、そのサーバーのリポジトリも再起動されます。その結果、リポジトリで RepositoryConfiguration オブジェクトが再度読み込まれます。

RepositoryConfiguration オブジェクトを参照または編集するには、Debug および Security Administrator 機能が必要です。

詳細については、『リリースノート』の「アップグレードの問題点」セクション、および『Identity Manager Tuning, Troubleshooting, and Error Messages guide』を参照してください。

WorkItemTypes Configuration オブジェクト

この設定オブジェクトは、sample/workItemTypes.xml で定義され、init.xml および update.xml によってインポートされます。このオブジェクト要素では、サポートされる作業項目のタイプ名、拡張、および表示名が列挙されます。

extends 属性では、作業項目タイプ (workItem タイプ) の階層が許可されます。workItem タイプが次の場合、Identity Manager で作業項目が作成されると、指定されたユーザーに作業項目が委任されます。

- 委任されたタイプ
- 委任されるタイプの下位の workItem タイプのいずれか

表 A-1 workItem タイプ

タイプ	拡張	表示名
workItem	なし	すべての作業項目
approval	workitem	承認
organizationApproval	approval	組織の承認
resourceApproval	approval	リソースの承認
roleApproval	approval	ロールの承認
roleChangeApproval	approval	ロール変更の承認
applicationRoleApproval	roleApproval	アプリケーションの承認
applicationRoleChangeApproval	roleChangeApproval	アプリケーション変更の承認
assetRoleApproval	roleApproval	アセットの承認

表 A-1　workItem タイプ (続き)

タイプ	拡張	表示名
assetRoleChangeApproval	roleChangeApproval	アセット変更の承認
businessRoleApproval	roleApproval	ビジネスロールの承認
businessRoleChangeApproval	roleChangeApproval	ビジネスロール変更の承認
itRoleApproval	roleApproval	IT ロールの承認
itRoleChangeApproval	roleChangeApproval	IT ロール変更の承認
attestation	workItem	アクセスレビューアテステーション
accessReviewRemediation	workItem	アクセス
review	workItem	是正

SystemConfiguration オブジェクト

SystemConfiguration オブジェクトでは、数多くのシステム動作の中央制御点、およびシステムの動作に対する持続的なカスタマイズを格納する方法が提供されます。カスタマイズの重要性およびデプロイヤによるカスタマイズ頻度を考慮して、ここでは可能なカスタマイズのすべては網羅していません。ここでは、一般的なカスタマイズのいくつかを示します。

パスワード確認ポップアップの表示を制御する

System Configuration オブジェクトの forgotPasswordChangeResults 属性では、ログイン時に「パスワードをお忘れですか?」ボタンをクリックして、ユーザーまたは管理者がパスワードの変更を開始したあとに、Identity Manager で確認ページを表示するかどうかを制御します。

- forgotPasswordChangeResults.User のデフォルト値は true です。
- forgotPasswordChangeResults.Admin のデフォルト値は false です。

委任履歴リスト長を設定する

delegation.historyLength 属性では、「End User View workItem Delegation」フォームで表示される、現在と完了済みの両方の委任リストのサイズが制御されます。この属性では、委任テーブルに表示できる委任の最大数が指定されます。ここで設定する値に関係なく、テーブルには現在の委任がすべて表示されることに注意してください。

SystemConfiguration オブジェクトには、記録された以前の委任の数を制御する security.delegation.historyLength 属性が含まれています。

スケジューラの起動を登録する (クラスタ環境の場合)

scheduler.hosts 属性では、Identity Manager アプリケーションインスタンスごとにスケジューラの起動動作が登録されます。

scheduler.hosts の値は、制御する各ホストのエントリが含まれるマップです。キーは Identity Manager アプリケーションインスタンスの hostname です。

注 hostname 値を参照するには、Identity Manager インストールの debug/GetStatus.jsp ページに移動します。

次の値が有効です。

- enabled (デフォルト)
- disabled
- manual (中断)

値が指定されない場合、または無効な値が指定された場合は、デフォルト値が使用されます。

注 Waveset.properties ファイルの task.scheduler.enabled および task.scheduler.suspended プロパティーで、System Configuration オブジェクトに設定された値が上書きされます。

Configuration:System 設定の scheduler 属性の例は次のとおりです。

```
<Attribute name='scheduler'>
   <Object>
      <Attribute name='hosts'>
         <Map>
            <MapEntry key='goliad' value='enabled'/>
            <MapEntry key='sanjacinto' value='manual'/>
            <MapEntry key='washington' value='disabled'/>
         </Map>
      </Attribute>
   </Object>
</Attribute>
```

Role Configuration オブジェクト

Role Configuration オブジェクトでは、サポートされるロールタイプ、アクション、およびリスト列が定義されます。次のセクションでは、ロールタイプの定義でサポートされる要素について説明します。

タイプ

ロールタイプ属性は、Role Configuration オブジェクトの types セクションで設定されます。リスト内のロールタイプ (ビジネスロールや IT ロールなど) ごとに、次の属性を指定する必要があります。

displayName

値がメッセージカタログキーであるタイプの表示名を指定します。

authType

ロールタイプに関連付けられた認可タイプを指定します。認可タイプを使用すると、このロールタイプを参照および管理できるユーザーに対して、詳細な認証を付与できます。authType を定義していない場合は、AuthorizationTypes 設定オブジェクトに追加します。この authType のロールにアクセス権を付与する Permission 内のタイプとして、AdminGroup (機能) 内の authType を参照する必要があります。

注	すべてのロールには認可タイプがあります。認可タイプなしでロールを読み込むと、認可タイプのデフォルトが ITRole に設定されます。

workItemTypes

ロール割り当ての承認およびロール変更の承認のために作成できる作業項目のタイプです。特定の workItem タイプを定義していない場合は、WorkItemTypes 設定オブジェクトに追加します。

機能

features 属性には、次の機能が含まれます。

- changeApproval - 指定した場合、ロールに指定された所有者が、このタイプのロールへの変更をすべて承認する必要があることを示します。所有者が指定されていない場合は、承認は発生しません。
- changeNotification - 指定した場合、このタイプのロールになんらかの変更が行われると、指定されたロールの所有者に電子メールで通知が送信されることを示します。
- containedTypes - 必須機能であり、その値はこのタイプに含めることができるロールタイプのリストです。許可される値は次のとおりです。
  - BusinessRole
  - ITRole
  - ApplicationRole
  - AssetRole
  - カスタムロールタイプ
- assignResources - 指定した場合、リソースグループをこのタイプのロールに割り当てることができることを示します。指定しない場合は、デフォルトでこのタイプのロールにリソースを割り当てることができなくなります。
- userAssignment - 指定した場合、このタイプのロールをユーザーに直接割り当てることができるかどうかを示します。このロールタイプをユーザーに直接割り当てることができる場合、この機能では手動および自動でユーザーを割り当てることができるかどうかも指定されます。指定しない場合は、デフォルトでユーザーの割り当ては許可されなくなります。

  注 このリリースでは自動割り当てはサポートされていませんが、将来のリリースでサポートされる予定です。

  - manual - 指定した場合 (true または false)、このタイプのロールをユーザーに手動で割り当てることができるかどうかを示します。
  - activateDate - 指定した場合 (true または false)、ユーザーに、このタイプのロールを割り当てるときに将来の有効化 (開始) 日付を指定できるかどうかを示します。この機能は、userAssignment.manual が true の場合にのみ有効であることに注意してください。
  - deactivateDate - 指定した場合 (true または false)、このタイプのロールを割り当てるときに将来の無効化 (終了) 日付を指定できるかどうかを示します。この機能は、userAssignment.manual が true の場合にのみ有効であることに注意してください。

注 userAssignment.manual が true に設定されていない場合でも、activateDate と deactivateDate の両方を true に設定できます。あるロールタイプに対して両方の属性を true に設定した場合、およびオプションで別のロールにそのロールが含まれている場合、オプションのロールをユーザーに割り当てるときに有効化および無効化の日付を指定できます。

- roleExclusions - 指定した場合、ユーザーに割り当てることができないロールのリスト (除外リスト) をロールエディタで指定できることを示します。

アクション

アクション属性では、ロール管理者が「ロールのリスト」テーブル内の 1 つ以上のロール上で、既存のロールに含まれるロールにロールの除外を追加するときに実行できるアクションセットが定義されます。

ロール設定には、次の 3 つのアクションセットが指定されます。

- actions - 主要なロールリストおよび「ロールの検索結果」ページに表示されるアクション
- addContainedRoleActions - 管理者が含まれるロールをロールに追加しているときに表示されるアクション
- addRoleExclusionsActions - 管理者がロールの除外をロールに追加しているときに表示されるアクション

各アクションは、次の属性を使用して定義されます。

- action - コマンドを指定します。
- label - 表示名メッセージキーを指定します。
- requiredPermissions - 管理者のアクセス権に応じて、アクションが表示されるかどうかを制御するアクセス権です。
  - Type - 管理者が特定の権限を持つ必要があるオブジェクトのタイプです。
  - Rights - 管理者が特定のオブジェクトタイプに対して持つ必要がある権限のリストです。
- selectionRequired - このアクションに対してロールを選択する必要があることを示します。
- type - ロールアクションタイプ (create、update、delete、または task) を指定します。
- view - create、update、および delete のロールアクションタイプに対するアクションの実行中に、この属性の内容をロールビューにコピーします。

- task - タスクアクションタイプに対して起動するタスクを指定します。
- skipTaskLaunchForm - true に設定した場合、タスク起動フォームをスキップします。それ以外の場合は、タスク起動フォーム (存在する場合) が表示されます。task アクションタイプに適用されます。

リスト列

リスト列属性では、ロールのリスト (リストロールやロール結果の検索など) を参照するときに、列ヘッダーとして表示する属性名とラベルのセットが定義されます。

リスト列ヘッダーとして表示する一意の属性セットを指定できます。定義された各列の属性は次のとおりです。

- name - 表示するロール属性の名前です。
- displayName - 列ヘッダーに表示する名前を表示します。
- rule - 属性値の形式を指定できるオプションのルールです。rule はリストの行ごとに呼び出され、ルールで返される値が各テーブルセルに表示されます。

その他のオプション

Role Configuration オブジェクトでは、次のオプションも設定できます。

- roleListMaxRows - 表示するロールの最大数です。
- roleListPageSize - 1 ページに表示するロールの数です。

End User Tasks オブジェクト

End User Tasks オブジェクトでは、Identity Manager ユーザーインタフェースから実行できるタスクが定義されます。EndUserTask 認可タイプを任意の TaskDefinition オブジェクトに割り当て、EndUserRule 認可タイプを表示する必要のある任意の Rule オブジェクトに割り当てることができます。

User オブジェクトを更新する

特定のタイプの変更では、管理者がすべての User オブジェクトを更新する必要があります。たとえば、RepositoryConfiguration の Type.USER のインライン属性を変更する場合は、すべての User オブジェクトを更新する必要があります。IDMSchemaConfiguration オブジェクトで属性をクエリー可能または概要としてマークするたびに、古い、未変更のオブジェクトに変更を反映するために、すべての User オブジェクトを更新する必要があります。新しいバージョンの Identity Manager で新しい属性が追加されるとき、または新しいバージョンの Identity Manager で既存の属性の値が変更されるときに、同様のロジックが適用されます。アップグレードプロセスまたは管理者は、古い、未変更のオブジェクトに変更を反映するために、すべての User オブジェクトを更新する必要があります。

既存のユーザーを再直列化するには、次の 3 つの方法があります。

- 通常処理中に個々のユーザーオブジェクトを変更する。

  たとえば、ユーザーインタフェースからユーザーアカウントを開き、変更ありまたは変更なしで保存します。

  欠点: この方法には時間がかかり、管理者はすべての既存ユーザーが再直列化されていることを注意深く確認する必要があります。

- lh refreshType ユーティリティを使用して、すべてのユーザーを再直列化する。refreshType ユーティリティの出力は、更新済みのユーザーリストです。

  ```
  lh console

  refreshType User
  ```

  欠点: refreshType ユーティリティはバックグラウンドではなくフォアグラウンドで動作するため、このプロセスに時間がかかる場合があります。ユーザー数が多い場合は、すべてを再直列化するのに長時間かかります。

- 延期タスクスキャナを使用する。

注 延期タスクスキャナのプロセスを実行する前に、Identity Manager 統合開発環境 (Identity Manager IDE) またはその他の方法を使用して、System Configuration オブジェクトを編集する必要があります。

'refreshOfType' を検索し、'2005Q4M3refreshOfTypeUserIsComplete' および '2005Q4M3refreshOfTypeUserUpperBound' の属性を削除します。

System Configuration オブジェクトを編集後、その変更を反映するには、リポジトリにそのオブジェクトをインポートする必要があります。

欠点：この方法では、ほぼすべての User オブジェクトが確認および再書き込みされるため、次回の延期タスクスキャナの実行には時間がかかります。ただし、その後の延期タスクスキャナは通常の速度と期間で実行されるはずです。

国際化サポートを有効にする

この付録では、複数の言語を使用したり、英語以外の言語を表示したりするように Identity Manager を設定する方法について説明します。

アーキテクチャーの概要

表 B-1　Identity Manager の国際化のコンポーネント

ファイル	説明
`WPMessages.properties`	デフォルトのメッセージファイルは、`$WSHOME/idm/web/WEB-INF/classes/com/waveset/msgcat` にあります。`idmcommon.jar` ファイルの一部として出荷されています。 メッセージテキストを英語で表示します。これは、Identity Manager のインストール動作をカスタマイズして変更しないかぎり、デフォルトでロードされます。
`Waveset.properties`	`$WSHOME/config` にあります。 複数言語のサポートを有効にするには、`Internationalization.enabled` を true に設定する必要があります (デフォルトは *true*)。
System Configuration オブジェクト	カスタムメッセージカタログを指定します。
サポートされる各言語の追加のメッセージファイル	サポートされる追加の言語には、それぞれ独自のメッセージファイルが必要です。`WPMessages_`*xx*`_`*XX*`.properties` (ここで、*xx* は言語を表し、*XX* は国を表す。) たとえば、`WPMessages_en_US.properties` には、アメリカ英語のメッセージが含まれています。各国際化カタログには独自の `.jar` があります。

注

- 新しいカタログをデフォルトカタログと同じ名前を使用した /config にロードした場合、その新しいカタログはデフォルトより優先されます。
- 複数のメッセージファイルがある場合は、catalogname:keyname を指定して、メッセージキーを取得するカタログを指定できます。

次のカタログエントリでは、製品名の表示方法が制御されます。

```
PRODUCT_NAME=Identity Manager
LIGHTHOUSE_DISPLAY_NAME=[PRODUCT_NAME]
LIGHTHOUSE_TYPE_DISPLAY_NAME=[PRODUCT_NAME]
LIGHTHOUSE_DEFAULT_POLICY=Default [PRODUCT_NAME] Account Policy
```

一般的なエントリ

メッセージは、キーとテキストの組み合わせに含まれており、次の 3 つの部分で構成されます。

- テキスト文字列、またはキー。データを取得するためにコードで使用される識別子です。この必須コンポーネントは翻訳しないでください。このコンポーネントは製品設定で使用され、翻訳用のプレースホルダとして機能します。
- キーとテキストを区切る等号 (「=」)。このエントリは必須です。
- アプリケーションの実行時に表示されるデータを含む文字列。このエントリは翻訳されたものであり、ブラウザでページが描画されるときにキーの代わりに使用されます。

リソース配列の各行には 2 つの文字列が含まれています。各行の、引用符で囲まれた 2 番目の文字列を翻訳します。

翻訳対象の特定の文字列には、文字列が表示されるときに文字列に挿入されるデータ用の特殊コードが含まれています。たとえば、次の文字列を翻訳するとします。

```
UI_USER_CONNECT={0}, connected at 100 mbs
```

描画されたものは、「jfaux, connected at 100 mb」などとなります。

通常、翻訳文字列はブラウザ内に表示されるため、次に示すように、HTML タグを追加して文字列の書式を設定することをお勧めします。

```
_FM_ACCOUNT_ID_HELP=<b>Account ID</b><br>Enter a name for this
user.This field is required.
```

複数言語のサポートの有効化

Identity Manager 複数言語のサポートを有効にするには、このシナリオで説明する手順を使用します。

手順 1: ローカライズされたファイルをダウンロードしてインストールする

インストールの前に

ローカライズされたファイルをインストールする前に、次のタスクを実行します。

1. Identity Manager をインストールします。詳細なインストール手順については、『Identity Manager インストール』を参照してください。
2. アプリケーションサーバーの次のロケールが UTF-8 に設定されていることを確認します。
   - アプリケーションサーバーインスタンス
   - データベース
   - Java 仮想マシン (JVM)

ロケールの設定に関する情報については、これらの製品のマニュアルを参照してください。

メッセージカタログファイルのダウンロード

Identity Manager ソフトウェアダウンロード Web サイトでは、次のローカライズ済みのメッセージカタログが提供されています。適切なメッセージカタログの jar ファイルをダウンロードし、そのファイルを WEB-INF/lib ディレクトリに配置します。

表 B-2　メッセージカタログファイル

ファイル名 (.zip)	言語	ロケール
IDM__8_0_l10n_de	ドイツ語	de_DE
IDM__8_0_l10n_es	スペイン語	es_ES
IDM__8_0_l10n_fr	フランス語 (フランスおよびカナダ)	fr_FR
IDM__8_0_l10n_it	イタリア語	it_IT
IDM__8_0_l10n_ja	日本語	ja_JP
IDM__8_0_l10n_ko	韓国語	ko_KR
IDM__8_0_l10n_pt	ポルトガル語 (ブラジル)	pt_BR
IDM__8_0_l10n_zh	簡体字中国語	zh_CN
IDM__8_0_l10n_zh_TW	繁体字中国語	zh_TW

ZIP ファイルを一時的な場所にダウンロードします。デフォルトでは、ZIP ファイルの内容は FileName¥IDM_8_0_l10n ディレクトリに抽出されます。ここで、FileName は、ZIP 拡張子を除いたダウンロードファイルの名前と一致します。

Zip ファイルの内容

抽出したすべての ZIP ファイルには次のものが含まれます。

- ローカライズされたメッセージカタログを含む JAR ファイル、ヘルプファイル、およびその他の重要なファイル。JAR ファイルの名前は IDM_7_0_l10n_Locale.jar です。
- 『Identity Manager ローカリゼーション README』

その他の翻訳物が利用できることもあります。

ローカライズされたファイルのインストール

次の手順を使用して、ローカライズされたファイルをアプリケーションサーバーにインストールします。

1. JAR ファイルを、一時的な場所から IdentityManagerInstallation/WEB-INF/lib ディレクトリにコピーします。

手順 2: Waveset.properties ファイルを編集する

Waveset.properties ファイルを編集するには、次の処理を実行します。

1. 任意のエディタで、IdentityManagerInstallation/config/Waveset.properties ファイルを開きます。
2. Internationalization.enabled プロパティーを true に変更します。
3. 変更を保存してファイルを閉じます。
4. Identity Manager を再起動するか、または次の場所にあるデバッグページで「Reload Properties」をクリックします。

   http://*host*:*port*/idm/debug.url

匿名登録処理中に ASCII アカウント ID と電子メールアドレスを保守する

デフォルトでは、Identity Manager の匿名登録処理中に、employeeId のほかにユーザーが用意した名 (firstName) と姓 (lastName) を使用すると、accountId および emailAddress の値が生成されます。匿名登録処理では、電子メールアドレスとアカウント ID に ASCII 以外の文字が含まれる可能性があるため、国際的なユーザーは、匿名登録処理中に Identity Manager で ASCII アカウント ID と電子メールアドレスが保守されるように、EndUserRuleLibrary 規則を変更する必要があります。

匿名登録処理中に ASCII でアカウント ID と電子メールアドレスを保守するには、次の 2 つのステップに従います。

1. 次の表で示すように、EndUserRuleLibrary の次の規則を編集します。

表 B-3 EndUserRuleLibrary 規則を編集する

編集する規則	編集の目的
getAccountId	employeeId のみを使用する (firstName と lastName を削除する)

表 B-3　EndUserRuleLibrary 規則を編集する (続き)

編集する規則	編集の目的
getEmailAddress	employeeId のみを使用する (firstName、lastName、および「.」を削除する)
verifyFirstname	アジア人の 1 文字の名前を許可するように、長さのチェックを 2 から 1 に変更する

2. EndUserAnonEnrollmentCompletionForm を編集して、getAccountId および getEmailAddress 規則への呼び出しから firstName および lastName 引数を削除します。

索引

L

M

O

P

R

S

W

X

あ

い

う

え

お

か

き

く

け

こ

さ

し

す

そ

ち

て

に

は

ひ

ふ