Sun™ Identity Manager 8.0
ワークフロー、フォーム、およびビュー

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

Part No: 820-5454

目 次

はじめに

本書『Sun™ Identity Manager ワークフロー、フォーム、およびビュー』では、Sun™ 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 Identity Manager ワークフロー、フォーム、およびビュー』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、画面、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 ワークフロー、フォーム、およびビュー』は、次の章で構成されています。

- 第 1 章、「Identity Manager のワークフロー」- Sun™ Identity Manager (Identity Manager) のワークフローについて説明します。
- 第 2 章、「Identity Manager のフォーム」- Identity Manager の管理者向けおよびユーザー向けインタフェースの各種ページを定義するフォームをカスタマイズすることで、選択したページの外観や動作をカスタマイズする方法について説明します。
- 第 3 章、「XPRESS 言語」- Identity Manager 全体で使用される、式とスクリプトを記述するための XML ベースの言語、XPRESS の基本機能について紹介します。

- 第 4 章、「XML オブジェクト言語」- string、list、map などの一般的な Java オブジェクトを表現できる XML 要素の集合、XML オブジェクト言語の基本機能について紹介します。
- 第 5 章、「Identity Manager のビュー」- Identity Manager で使用されるデータ構造、Identity Manager ビューについて説明します。
- 第 6 章、「HTML 表示コンポーネント」- Identity Manager の HTML 表示コンポーネントライブラリについて説明します。HTML 表示コンポーネントは、フォームのカスタマイズで使用されます。
- 付録 A、「フォームとプロセスのマッピング」- Identity Manager で使用されるフォームおよびワークフロープロセスと、それに対応するシステム名の一覧を示します。

表記上の規則

この項の表は、本書で使用する表記規則について説明しています。

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し、ユーザーコンプライアンスを管理するための、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、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 の配備に関する技術概要』: Identity Manager 製品の概念に関する概要 (オブジェクトアーキテクチャーを含む) および基本的な製品コンポーネントの紹介が記載されています。
- 『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™ 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™ 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、およびこの製品を操作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、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 ワークフロー、フォーム、およびビュー』であり、Part No. は 820-5454 です。

第 1 章

ワークフロー

この章では、Identity Manager のワークフローについて説明します。

この章の内容

関連する章

- 「XPRESS 言語」- ワークフローやフォームにロジックを含めるために XPRESS 言語で記述する式の一覧を示し、それぞれについて説明します。

注	一般的なワークフローについてより深く理解するには、Identity Manager IDE プラグインを使用して、実際にサンプルワークフローの表示、実行、およびデバッグを行なってください。Identity Manager IDE をインストールおよび設定する手順については、https://identitymanageride.dev.java.net を参照してください。

ワークフローについて

Identity Manager ワークフローは、定義されている規則に従って一貫して実行される、一連のアクションとタスクを定義します。Identity Manager 統合開発環境 (IDE) のグラフィカルユーザーインタフェースを使用して、Identity Manager によって起動される各ワークフローをカスタマイズできます。

ワークフローを操作する前に、次の項目について理解してください。

- ワークフローの一般的な概念
- Identity Manager でワークフローがどのように使用されるか

ワークフローとは

一般に、ワークフローは論理的で反復可能なプロセスであり、ドキュメント、情報、またはタスクが、定義された手順の規則セットに従って、アクションの関与者から別の関与者に渡されます。関与者は人、マシン、またはその両方の場合があります。

Identity Manager では、この概念は具体的には Identity Manager のワークフローコンポーネントとして実装されています。この機能は、ユーザーアカウントの作成、更新、有効化、無効化、および削除を管理する、複数のプロセス (ワークフロー) から構成されます。

製品インタフェースのどの位置から呼び出すかに応じて、ワークフローはワークフロー、タスク、プロセス、またはタスク定義として参照されます。

ワークフローを使用する場面

実行するほとんどの Identity Manager タスクは、ワークフロープロセスのセットとして定義できます。Identity Manager にユーザーを作成するときは、たとえば、対応するワークフロープロセスが、次の処理を定義、実行します。

- パスワードポリシーを確認する
- 承認者に電子メールを送信する
- 各承認の結果を評価する
- ユーザーアカウントを作成する
- 監査レコードを作成する

ワークフローはユーザーとの対話なしで自動的に実行できますが、承認の形でユーザーとの対話が必要な場合もあります。

通常、ワークフローはビューのチェックインに付随して呼び出されます。ビューのチェックインは、フォームとビューを実装するページで「保存」をクリックしたときに行われます。

リポジトリ内のワークフロー

Identity Manager のリポジトリには、通常は、タイプが WFProcess の設定オブジェクトとしてワークフローが存在します (Create User ワークフローは、このオブジェクト定義の唯一の例外で、ProvisioningTask オブジェクトとして定義される)。taskType は常に `Workflow` です。

注 Identity Manager は、ワークフローの実行中のリポジトリオブジェクト (つまり、ユーザー) をロックしません。これは、ワークフローが数日にわたって実行される可能性があり、その間中リポジトリオブジェクトをロック状態で維持することはできないためです。ただし、同じユーザーによる、別の更新ワークフローの呼び出しは、Identity Manager によって禁止されます。

タスク定義とタスクインスタンス

`TaskDefinition` の起動インスタンスは、`TaskInstance` オブジェクトとして表されます。どちらのオブジェクトタイプも、「デバッグ」ページから表示できます。配備の現在のタスク定義またはタスクインスタンスにアクセスするには、次の手順に従います。

1. Identity Manager の管理者インタフェースの「デバッグ」ページで、「**List Objects**」ボタンの横にある「タイプ」メニューから「TaskDefinition」を選択します。
2. 「**List Objects**」をクリックします。使用可能なオブジェクトタイプのうち、アクセスできるオブジェクトタイプの一覧が表示されます。
3. オブジェクト (たとえば、`TaskDefinition`) を選択します。そのオブジェクトタイプのインスタンスのうち、表示する権限を持つすべてのインスタンスが表示されます。

ワークフロータスクが呼び出されると、ワークフローエンジンはリポジトリに *TaskInstance* を作成します。TaskInstance はリポジトリ内のオブジェクトの 1 つで、実行するワークフロープロセスの実行時状態を保持します。また、作成元となった TaskDefinition のコンテキスト変数と即時遷移情報を格納します。

TaskInstance は、TaskDefinition の生成 ID を使用して、説明的な TaskDefinition オブジェクトを参照します。TaskDefinition を編集すると、すでに実行中の TaskInstance は変更前の TaskDefinition オブジェクトを継続して使用しますが、新たに実行する TaskInstance は、新たに生成された ID を使用して修正後の TaskDefinition を使用します。

タスクインスタンスの削除のタイミング

TaskInstance の存続期間は、`resultLimit` パラメータによって決定されます。結果の有効期間の値にゼロが設定されている場合は、タスクは完了後ただちに削除されます。正の値が設定されている場合は、TaskInstance はその時間 (単位は分) だけ維持されます。

保留になっているワークフローの TaskInstance を削除するには、次の手順に従います。

1. Identity Manager の管理者インタフェースで、「サーバータスク」タブをクリックします。
2. 「すべてのタスク」を選択します。
3. 保留中の TaskInstance を選択し、「終了」をクリックします。

タスク定義パラメータ

次の表は、標準の設定パラメータを示しています。

表 1-1 ワークフローの標準設定パラメータ

パラメータ	説明
`name`	ユーザーが設定する、ワークフローの名前。Identity Manager のインタフェースには、この名前が表示されます。このタイプのほかのオブジェクトと重複する名前は付けられませんが、タイプが異なれば、同じ名前を付けることができます。
`taskType`	フィルタリング専用に使用されます
`executor`	タスクを実装するクラスの名前を識別します。デフォルトでは、ワークフローのこのパラメータのクラスは com.waveset.workflow.WorkflowExecutor です。
`suspendedable`	(ブール型) タスクを保留し、再開できることを表します。デフォルトは true です。
`syncControlAllowed`	(ブール型) 同期または非同期実行の要求をユーザーに許可するかどうかを表します。デフォルトは true です。

表 1-1 ワークフローの標準設定パラメータ (続き)

パラメータ	説明
execMode	デフォルトで使用する実行タイプを指定します。デフォルトは sync です。 この値が NULL の場合、または ExecMode.DEFAULT に設定されている場合は、ExecMode.ASYNC として扱われます。
executionLimit	タスクを実行できる制限時間を、秒単位で指定します。タスクには、実行の制限時間を指定できます。この制限時間が経過すると、スケジューラはそのタスクを終了できます。制限時間をゼロに設定すると、無制限として解釈されます。 デフォルトは 0 です。
resultLimit	タスクの完了後、インスタンスが存続できる制限時間を秒単位で指定します。デフォルトは 0 です。 タスクが完了または終了すると、タスクの結果を含む TaskInstance は、通常は指定した時間だけリポジトリに維持され、その時間が経過すると自動的に削除されます。 0 - TaskInstance は、タスクの完了後ただちに削除されます。 -1 - TaskInstance は自動的に削除されません。ただし、ユーザーが手動で削除することは可能です。 あとから分析するレポートを生成するタスクでは、通常は、このパラメータに数日間に相当する値を設定します。副次的な結果を生じる目的のみで実行され、それ自体は意味のある結果を生じないタスクでは、ゼロに設定します。
resultOption	(String 型) 繰り返しタスクの以前の実行結果をどのように扱うかを示すオプションを指定します。このオブジェクトはそのデータを定義し、データの処理方法を確認します。デフォルトは delete です。 wait - 古い結果が手動で削除される、または有効期限が切れるまで、タスクが実行されないようにします。これがスケジュールされていないタスクである場合は、起動時にエラーとなります。スケジュールされているタスクであれば、スケジューラはこれを無視します。 delete - タスクを実行する前に、古い結果を自動的に削除します。古いタスクは完了状態でなければなりません。 rename - 古い結果の名前を変更してからタスクを実行します。古いタスクは完了状態でなければなりません。 terminate - 現在実行中のタスクを終了し、削除します。これは delete オプションに似ていますが、タスクが実行中の場合は、それを自動的に終了します。

表 1-1　ワークフローの標準設定パラメータ (続き)

パラメータ	説明
`ayncExec`	このパラメータを true に設定すると、アクションの完了後も、次回の手動アクションまでワークフローの実行を継続し、ユーザーには次の作業項目を表示します。この設定は、ウィザード形式のワークフローをサポートします。 false に設定した場合は、ワークフローはバックグラウンドで実行を継続し、ユーザーがワークフローの次のステップを実行するときは、別のページ (通常は承認ページ) への移動が必要となります。
`visibility`	(String 型) このタスク定義の表示設定を宣言します。デフォルトは `run schedule` です。これ以外に、`invisible`、`run task`、および `schedule task` のオプションがあります。
`progressInterval`	Identity Manager が進行状況の変化を確認する間隔を、ミリ秒単位で指定します。 タスクには、タスクが進行状況を更新する間隔を指定できます。デフォルトは 5000 ミリ秒 (5 秒) です。間隔を短く設定するほど、より最新の状態を確認できますが、サーバーの負荷は増大します。

ワークフローエンジン

ワークフローエンジンは、実行時のワークフロープロセスの実行を可能にするソフトウェアサービスです。ワークフロープロセスをサポートするワークフローエンジンの主な機能は次のとおりです。

- プロセス定義を解釈する
- プロセスインスタンスを作成し、その実行を管理する
- アクティビティーを切り替え、そのアクティビティーの処理のために作業項目を作成する

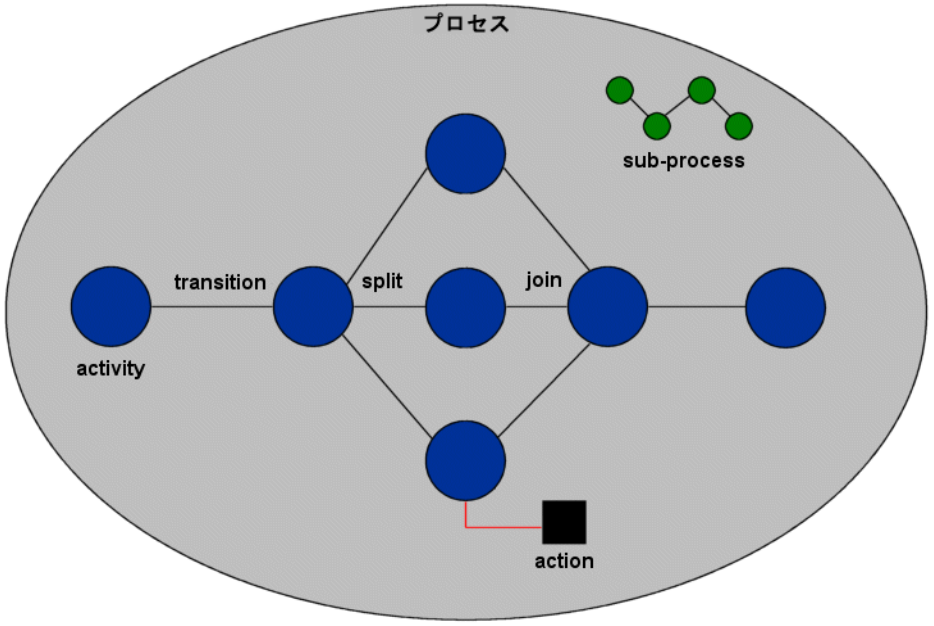
アクティビティーに手動アクションが含まれる場合は、Identity Manager はアクティビティーレベルで変数を受け取ります。ワークフロータスクに必要な記憶領域を最小限に抑えるために、ワークフローエンジンは完了したアクティビティーのほかのすべての変数をエクスポートの前に削除します。

ワークフロープロセスコンポーネントの概要

各ワークフロープロセスは、次のコンポーネントのいずれか、またはその組み合わせによって定義されます。ここでは、これらのコンポーネントについて詳しく説明します。

- Activity - プロセスに含まれる 1 つの論理手順を表します。
- Action - アクティビティーの実行方法を定義します。アクションは、簡単な式評価の場合もあれば、複雑な Java クラスの呼び出しの場合もあります。
- Transition - アクティビティーから別のアクティビティーへの移動を定義します。
- Split - 1 つのアクティビティーから複数のアクティビティーへの移動を定義します。Split は、さらに次のように定義されます。
  - OR Split - 各遷移パスをテストします。TRUE という値を持つ最初のパスが採用されます。
  - AND Split - 各遷移パスを採用します。
- Join - 複数のアクティビティーから単一アクティビティーへの移動を定義します。Join コンポーネントは、さらに次のように定義されます。
  - OR Join - 最初のパスが完了すると、次のアクティビティーが開始されるように指定します。
  - AND Join - すべてのパスが完了してから次のアクティビティーを開始するように指定します。
- Subprocess - プロセスに含まれる別のアクティビティーから呼び出されるアクティビティー、アクション、および遷移のセットを定義します。

図 1-1　　ワークフローの一般的なプロセスとコンポーネント



ワークフローアクティビティー

アクティビティーとは、ワークフロープロセスに含まれる 1 つのステップのことです。アクティビティーには、遷移、変数、アクションなど、複数のコンポーネントを含めることができますが、起動アクティビティーと終了アクティビティーを必ず含める必要があります。

起動アクティビティーと終了アクティビティーには、関連するアクションはありません。起動アクティビティーにはプロセスを開始するアクティビティーへの遷移を 1 つ以上設定しますが、終了アクティビティーには関連するアクションを設定しないようにしてください。

ワークフローアクション

ワークフローアクションには、ワークフローアクティビティー内で実行される操作を記述します。各アクティビティーには複数のアクションを含めることができます。Identity Manager には次のタイプのアクションがあります。

- アプリケーション -- WorkflowApplication インタフェースを介してリンクされた簡単な自動アプリケーション呼び出しです。Application アクションは、引数と変数を受け取り、変数を返すことができます。Identity Manager には、アクションのプロビジョニング、リポジトリアクセス、およびその他のユーティリティーに対応する広範囲のアプリケーションが用意されています。
- 手動 -- 人との対話を必要とするアクションです。Identity Manager フォームを使用して、ユーザーに要求する変数の名前を指定したり、アクションの所有者用のリポジトリに作業項目を作成したりできます。
- 式 -- XPRESS 言語の式を使用して定義される自動アクションです。スクリプトアクションは通常、XPRESS または JavaScript のプログラムです。
- サブプロセス -- ほかのワークフロープロセスを再帰的に呼び出すことによって実行されるアクションです。サブプロセスは、ルートプロセス内でしか定義できません。

ワークフローアクション変数

表 1-2

ワークフローアクション変数	説明
`name`	(省略可能) WorkflowComponent から継承されます。アクション名は通常 NULL です。
`id`	一意の数値 ID によって各アクションを識別します。これは、現時点ではアクティビティーのアクション配列へのインデックスです。
`title`	(省略可能) ワークフローの概要図でこのアクションのタイトルを算出するために使用されます。デフォルトではタイトルは名前ですが、この変数には変数値などの情報を含めることもできます。
`manual`	アクションが手動かどうかを指定します (フラグ)。この値は、ほかのすべてのアクションタイプフィールドよりも優先されます。

表 1-2　(続き)

ワークフローアクション変数	説明
request	手動アクションの作業項目の要求文字列を算出するために使用されます。このテキストは製品インタフェースに表示されるため、簡潔で、ユーザーに要求する作業を明確に記述した文字列にするようにしてください (たとえば、「Approve Role Engineering (エンジニアリングロールの承認)」、「Supply account ID (アカウント ID の入力)」、「Answer a question (質問への回答)」など)。request を入力しなかった場合は、Identity Manager は title の結果を使用します。title がない場合は、Identity Manager はアクティビティーの名前またはタイトル、あるいはその両方に基づく文字列を生成します。
requester	手動アクションの作業項目の要求元を示す文字列を算出するために使用されます。このテキストは、製品インタフェースに表示されます。要求元と見なされるユーザーまたは管理者の名前を指定するようにしてください。NULL の場合は、Identity Manager はワークフローケースを起動した主体が要求元であると見なします。
description	手動アクションの作業項目の説明文字列を算出するために使用されます。このテキストは、製品インタフェースに表示されます。通常は、要求文字列より長くなります。
timeout	1. 手動アクションに関して、Identity Manager がこのアクションの実行を待機する最大時間を分単位で定義します。タイムアウト時間に達すると、ワークフローの実行者はアクションが完了したと見なします。変数を使用して、タイムアウト後の制御フローを決定する必要があります。 項目がタイムアウトによって完了した場合は、ワークフローエンジンがこのアクティビティーの範囲内で WF_TIMEOUT 変数を true に設定します。 2. タイムアウト値を生成する式を指定します。
expression	アクションを定義する式ツリーのルートを指定します。この値を設定すると、この値が subprocess および application フィールドよりも優先されます。

表 1-2　　(続き)

ワークフローアクション変数	説明
subprocess	1. 呼び出すサブプロセスの名前を指定します。内部サブプロセスまたは外部サブプロセスを指定できます。外部の場合は、タイプ修飾した名前 (たとえば、TaskDefinition:foo、Configuration:foo など) を使用できます。 2. サブプロセス名を生成する規則を指定します。返されるオブジェクトは、String または ObjectRef です。String の場合は、subprocess のように、内部プロセスまたは外部プロセスを識別できます。
application	呼び出されるアプリケーションの名前を指定します。組み込みアプリケーションの抽象名を指定するか、または WorkflowApplication インタフェースを実装するクラスの完全修飾クラス名を指定します。 組み込みアプリケーションの名前は、WorkflowExecutor によって定義されます。
owner	アクションタイプが手動の場合は、作業項目を作成する所有者の名前をこのフィールドに指定する必要があります。 管理者の名前または変数参照構文 (たとえば、$(ADMIN_NAME)) を使用できます。
delegator	手動アクションの作業項目の委任者を示す文字列を算出するために使用される式を指定します。このテキストは GUI に表示されるため、要求の委任者と見なされるユーザーまたは管理者の名前を指定するようにしてください。
name	作業項目名を算出する式を指定します。通常はこの名前はランダムに生成される一意の識別子ですが、場合によっては、この名前を電子メール通知に使用したり、作業項目に直接移動する URL に埋め込んだりできるように、事前に名前を算出することが望ましい場合があります。
trackingId	作業項目名を算出する式を指定します。通常はこの名前はランダムに生成される一意の識別子ですが、場合によっては、この名前を電子メール通知に使用したり、作業項目に直接移動する URL に埋め込んだりできるように、事前に名前を算出することが望ましい場合があります。

表 1-2　　(続き)

ワークフローアクション変数	説明
localForm	手動アクションにはフォームオブジェクトを定義できます。このフォームオブジェクトは、アクションが完了したときに返してほしい変数をユーザーが指定するために、作業リストアプリケーションで使用されます。 フォームの指定には、次の 3 つの方法があります。 • _localForm に自己完結型フォームを設定する • _formRef にフォームへの参照を設定する • 名前を生成する XPRESS 規則を設定する
formRef	手動アクションの作業項目を編集するときに使用するフォームが含まれるオブジェクトへの参照を指定します。
formRule	フォームまたは Form オブジェクト自体の名前を生成する規則を指定します。
arguments	すべてのアクションタイプに関して、引数のリストを指定できます。variables に似ていますが、異なるネームスペースに存在し、ローカルのアクション変数と同じように扱えるため、変数名との衝突を心配する必要はありません。
variables	手動アクションによって使用されるローカル定義のアクション変数のリストです。これは、繰り返しを使用している場合に必要です。
returns	アクションの戻り値の宣言のリストです。ここには、アクションの結果の中で定義される変数とアクティビティーまたはプロセスで定義される変数とのマッピングを定義します。 ほかに方法がないかをさらに検討する必要がありますが、変数の混乱を避けながらマッピングを行うにはこれがもっとも簡単な方法のように思われます。この方法を使用しない場合は、ほかの範囲 (activity.action.variable など) で作成された変数を参照するパス式を導入する必要があります。
iteration	このアクションのために省略可能な繰り返し設定を定義するオブジェクトを指定します。
results	このアクションのために結果宣言のリストを指定します。
hidden	このアクションをワークフロー概要図に表示しないことを指定します (ブール型)。
condition	省略可能な条件式を指定します。これを設定した場合、アクションを実行するには式の値が true になる必要があります。false の場合、アクションは無視されます。

表 1-2　　(続き)

ワークフローアクション変数	説明
checkError	アクションが実行されるかどうかを指定します。true にすると、WF_ACTION_ERROR の値が NULL の場合にのみアクションが実行されます。これは、よく使用される <Condition> の短縮形です。
comments	(省略可能) 任意のコメントが入ります。
syncExec	手動アクションのために作成された作業項目に同期実行フラグを設定するように指定します。その結果、所有者が作業項目への変更をチェックインすると、ワークフローはバックグラウンドで実行されるようにスケジューリングされるのではなく、同期をとりながら実行されるはずです。 これは、ワークフローによってページの順序が制御される「ウィザード」型のワークフローにも適用されます。
exposedVariables	手動アクションに関して、作業項目に含める変数のリストを指定します。名前には、GenericObject パス式を指定できます。このリストが NULL の場合は、すべての変数が取り込まれます。
editableVariables	作業項目内で編集してから元のケースに同化させることができる変数のリストを指定します (手動アクションのみ)。このリストが NULL の場合は、exposedVariables リスト内のすべての変数が編集可能と見なされます。exposedVariables が NULL の場合は、すべてのワークフロー変数を編集できます。 多くの変数を複数の承認者に公開する必要があるが、各承認者が変数の一部しか変更することが許可されていない場合に、衝突を避けるために使用します。
Views	ビューを含む変数のリストを指定します。このリストにビューが含まれていると、作業項目の更新時に Identity Manager がビューを更新します。
ignoreTimeout	ユーザーが作業項目を表示または編集しているときにその項目がタイムアウトになって削除されると、通常は例外がスローされて Identity Manager にエラーページが表示されます。ただし、Identity Manager が概要情報の編集ではなく表示のみを目的として作業項目を使用する場合もあります。これらの作業項目では、ワークフローが処理を継続できるように、タイムアウトが短くできます。 このオプションを true に設定すると、作業項目ビューのハンドラが例外をスローしなくなります。プロセスの動作には影響しません。

表 1-2　　(続き)

ワークフローアクション変数	説明
authType	手動アクション用に作成された作業項目に割り当てる AuthType を指定します。この変数は、作業項目をほかの方法で認可する場合に使用します。たとえば、あるユーザーに対してリソース要求を承認する権限は与えるけれども、組織要求を承認する権限は与えない場合などです。
itemType	手動アクション用に作成された作業項目にユーザー定義のタイプ名を指定します。作業項目のカスタムカテゴリ (たとえば、「approval」や「wizard」など) を割り当てるときに、この変数を使用します。
targetId	省略可能。作業項目が処理しているオブジェクトの Identity Manager ID が入ります。
targetName	省略可能。作業項目が処理しているオブジェクトの Identity Manager 名が入ります。
targetObjectClass	省略可能。処理中のオブジェクトのオブジェクトクラス名が入ります。

アプリケーションアクションとは

アプリケーションアクションでは、スクリプトアクションより複雑な Java 呼び出しを行うことができます。

手動アクションとは

手動アクションは、手動によるやり取りを定義した、ワークフロープロセス定義の一部です。ワークフロー実行者が手動アクションを処理するときは、Identity Manager はリポジトリ内に WorkItem オブジェクトを作成します。作業項目に完了のマークが付いてからでないと、ワークフローは次に進むことができません。手動アクションには所有者を関連付ける必要があります。所有者の代わりに、所有者を決定する式でもかまいません。

ほとんどの手動アクションは、承認の要求に使用されるため、作業項目の表は管理者インタフェースの「承認」タブにあります。ワークフローに属する手動アクションはどれも、リポジトリ内の WorkItem オブジェクトで表されます。作業項目ビューには、WorkItem オブジェクト自体に関係するいくつかの属性と、ワークフロータスクからコピーされた、選択されたワークフロー変数の値が含まれます。これには、保留中の承認などの属性と、作業項目の承認に使用されるフォームが含まれます。Identity Manager では、手動アクションは標準ワークフローの一部として、組織、ロール、およびリソースを承認することができます。

手動アクションにはタイムアウトを割り当てることができます。

デフォルトのワークフロープロセス

Identity Manager IDE を使用することで、デフォルトの Identity Manager プロセスを編集し、一連のステップをカスタマイズできます。Identity Manager のワークフロー機能には、次のようなデフォルトプロセスのライブラリが含まれます。

- **ユーザーワークフロー** - ユーザーの作成、削除、更新、有効化、無効化、名前の変更など、Identity Manager ユーザーに関連するタスクのステップを定義します。
- **Identity Manager オブジェクトワークフロー** - リソース、リソースグループ、組織、組織単位など、Identity Manager のオブジェクトに関連するすべてのタスクのステップを定義します。たとえば、ロール管理やワークフロー管理などのワークフローは、単にビューの変更をリポジトリに適用するだけですが、承認や、その他のカスタマイズのきっかけを提供することもできます。
- **その他のワークフロー** - 調整、Remedy テンプレート、その他のカスタムタスクなど、Identity Manager の各種機能やオブジェクトのステップを定義します。

例

次のユーザー作成ワークフローは、タイムアウト時間に達した時点で、エスカレーションアクティビティーを呼び出すように変更されています。タイムアウト時間に達するまでは、APPROVED 変数の結果が評価されます。評価の結果に基づいて、承認と拒否に対応したアクティビティーのどちらに遷移するかが決定されます。

コード例 1-1

```
<Activity name='Wait'>
   <ManualAction name='approve' timeout='180'>
   <Owner name='$(APPROVER)'/>
   <Variable name='APPROVAL' value='false'/>
   <Return from='APPROVAL' to='APPROVED'/>
   <FormRef>
     <ObjectRef type='UserForm' id='#ID#UserForm:ApprovalForm'/>
   <FormRef>
   <ReportTitle>
     <concat>
       <s>Awaiting approval from \n</s>
       <ref>APPROVER</ref>
     </concat>
   </ReportTitle>
   </ManualAction>
   <Transition to='Escalate'>
     <eq>
       <ref>WF_ACTION_TIMEOUT</ref>
       <s>true</s>
     </eq>
   </Transition>
   <Transition to='Approved'>
     <eq>
       <ref>APPROVED</ref>
       <s>true</s>
     </eq>
   </Transition>
   <Transition to='Rejected'/>
</Activity>
```

作業項目タイプ

手動アクションには、ワークフローエンジンによって手動アクションが実行された場合に生成される作業項目に、タイプを割り当てる機能があります。カスタマイズ内容に作業項目タイプを割り当てることで、表示または操作の対象となる work items のセットをフィルタリングできます。

システムは、次のタイプの作業項目を認識します。

表 1-3 作業項目タイプ

作業項目タイプ	説明
approval	承認の作業項目であることを示します。
ウィザード	ユーザーによる任意操作の作業項目であることを表します。

表 1-3　作業項目タイプ (続き)

作業項目タイプ	説明
保留	一時的な作業項目であることを表します。ワークフローにバックグラウンドでの実行を強制するときは、このタイプを使用します。

これ以外に、カスタマイズした作業項目タイプを割り当てることもできます。たとえば、リソースの承認を表す作業項目タイプを resource に設定したり、ロールの承認を表す作業項目タイプを role に設定したりできます。

作業項目のコンテキスト

作業項目は、<ManualAction> 指令を使用して起動されます。指定されたワークフローに関連付けられたフォームでは、ベースコンテキストを variables.user に設定できます。これにより、変数名に user.variables を入れる必要がなくなります。

作業項目はネームスペースであるため、フォームの一般的な属性名は次のようになります。

- complete (WorkItem 属性)
- variables.* (タスク変数)
- variables.<view>.accounts[*].*
- variables.<view>.waveset.*
- variables.<view>.accountInfo.*
- :display.session (所有者の *session*)

カスタムタスクと管理者承認の両方に適用されます。

認証タイプ

手動アクションには、作成する作業項目の認証タイプも指定できます。認証タイプは作業項目タイプとは異なり、現在の管理者が権限を持たない項目を排除できるように、クエリーに返される作業項目をシステムが自動的にフィルタリングするタイプです。通常は、承認者としての権限を持つどの管理者にも、担当する組織のすべての作業項目を表示する権限が割り当てられます。

手動アクションに作業項目の認証タイプを指定するには、次のように authType 属性を使用します。

```
<ManualAction authType='RoleApproval'>
```

作業項目タイプの割り当て

ManualAction 定義に項目タイプを指定するには、itemType 属性を設定します。次に例を示します。

```
<ManualAction itemType='approval'>
```

WorkItem の管理表示機能の制限

通常は、承認者としての権限を持つどの管理者にも、担当する組織のすべての作業項目を表示する権限が割り当てられます。管理者が組織の作業項目のサブセットのみを表示できるようにするときは、次の手順に従います。

1. WorkItem タイプを拡張する、新しい認証タイプを定義します。たとえば、RoleApproval タイプを定義します。
2. 作業項目自体ではなく、新しい認証タイプに対して権限を持つ、新しい機能を定義します。たとえば、RoleApproval タイプに対する権限を持つ、承認者ロールの機能を定義します。
3. 一般的な承認者の機能ではなく、管理者に対して、承認者ロールの機能を割り当てます。
4. ワークフローで使用される各手動アクションに、適切な認証タイプを設定します。

作業項目の委任

ワークフロー内の作業項目 (手動アクション) を委任できるようにするには、*delegator* と *delegators* を入力引数として渡し、<ManualAction> の <WorkItemDelegator> 要素と <WorkItemDelegators> 要素でそれぞれ参照する必要があります。

delegator と *delegators* の値を取得するには、com.waveset.provision.getDelegateObjects ワークフローサービスメソッドを呼び出します。このメソッドは次の引数を取ります。

- 次の 2 つの属性のどちらか。
  - accountId - 委任情報を渡すユーザーの名前を指定します。

    または

    accountIdList - 委任情報を渡すユーザー名のリストを指定します。
- delegateWorkItemType - 委任情報を渡す作業項目タイプ (approval、roleApproval、または attestation) を指定します。有効な作業項目タイプは、WorkItemTypes 設定オブジェクトで定義されます。
- delegateWorkItemTypeObjectName - 委任情報が収集されるオブジェクトの名前を指定します。この引数が有効なのは、delegateWorkItemType が organizationApproval、resourceApproval、roleApproval、またはこれらの作業項目タイプの拡張である場合だけです。

- `delegateWorkItemTypeObjectType` - 委任情報が収集されるオブジェクトのタイプを指定します。この引数が有効なのは、`delegateWorkItemType` が organizationApproval、resourceApproval、roleApproval、またはこれらの作業項目タイプの拡張である場合だけです。

サービスから *delegateObjects* 引数で委任オブジェクトのリストが返されます。

delegateObject

各 *delegateObject* には次の属性が入ります。

- `approver` - この作業項目の承認者を指定します。
- `delegator` - 作業項目の初期 (最初の) 委任者を指定します。このユーザーは、作業項目の `<WorkItemDelegator>` として設定されます。
- `delegators` - 委任者名のリストを始めから終わり (最終承認者の前) まで指定します。このユーザーリストは、手動アクションの `<WorkItemDelegators>` 要素として設定されます。委任が見つからなかった場合、この値は NULL です。

遷移の作成

遷移は、アクティビティーが 1 つまたは複数の別アクティビティーに移動するための規則を定義します。遷移には条件を設定できます。つまり、特定の条件が満たされる場合にのみ、遷移が行われるように設定できます。単純なアクティビティーには、アクティビティーを構成するアクションが完了するとただちに実行される、1 つの無条件遷移のみを含めることができます。

Identity Manager が使用するプロセスの更新

プロセスをカスタマイズするときは、ワークフロープロセスが想定どおりに正しく完了するように、変更内容を検証してから保存してください。保存したら、Identity Manager が使用できるように、変更したワークフローをインポートします。Identity Manager IDE デバッガを使用することもできます。Identity Manager IDE によるワークフロープロセスの編集については、「Identity Manager IDE の使用」を参照してください。

本稼動環境のワークフローの編集

本稼働環境のワークフロープロセスはカスタマイズしないでください。

元のワークフローのインスタンスの実行中にワークフローのアクティビティーやアクションを編集すると、問題が生じることがあります。具体的には、TaskInstance には TaskDefinition ワークフローへの参照が含まれ、ID によって特定される現在のアクティビティーまたはアクションが格納されています。これらの ID を変更すると、実行を再開したときに、想定されているとおりにタスクが再開されない可能性があります。

本稼働環境のワークフローの編集を回避できない場合は、次の手順に従います。これにより、古い定義を使用するタスクインスタンスによって保留されている作業項目の喪失を回避できます。

1. TaskDefinition の現在の名前を、タイムスタンプを付けて変更します。たとえば、Create User ワークフローを変更する場合は、TaskDefinition の名前を、Create User から Create User 20030701 に変更します。ワークフローの TaskDefinition の名前は、Identity Manager IDE を使用して変更できます。
2. 編集したワークフローを保存し、インポートします。

この手順に従うと、Identity Manager 内で保留状態にある既存の Create User タスクに問題が生じるのを回避できます。この場合、保留中のタスクで参照される、TaskDefinition の一意の ID は維持されます。

標準ワークフロー

Identity Manager には、使用されるプロセスにマップされた、標準のワークフローが最初から用意されています。これらのデフォルトワークフローの簡単な説明については、「ワークフローのデフォルトアクティビティー」を参照してください。デフォルトワークフローを表示、編集するには、次の手順に従います。

1. Identity Manager IDE を起動します。Identity Manager IDE の使用方法については、『Identity Manager 配備ツール』に記載されている「Identity Manager IDE の概要」を参照してください。
2. 「ファイル」>「リポジトリオブジェクトを開く」>「ワークフロープロセス」を選択します。標準ワークフロープロセスと、配備に含まれるカスタムワークフローを示す「ワークフロープロセス」リストが表示されます。
3. 表示または編集するワークフローの名前をダブルクリックします。

Identity Manager の管理者インタフェースから「設定」>「フォームおよびプロセスマッピング」を選択すると、プロセスマッピングを表示できます。

デフォルトワークフローの一般的なカテゴリ

プロセスのカスタマイズ

1 つまたは複数の Identity Manager のプロセスを変更することで、ステップを削除して新しいステップを設定したり、既存のステップをカスタマイズしたりできます。プロセスの各ステップは、アクティビティーによって表されます。

ワークフローの編集時または作成時に利用できる定義済みアクティビティーを提供するワークフローツールボックスは、ワークフローの変更に役立ちます。

このツールボックスを開くには、ダイアグラムビューを右クリックし、ツールボックスオプションを選択します。

ワークフローのデフォルトアクティビティー

次に、用意されているデフォルトアクティビティーをカテゴリ別に示します。

表 1-4 ワークフローのデフォルトアクティビティー

アクティビティー	説明
Add Deferred Task	延期タスクのスキャナ情報をオブジェクトに追加します。
Audit Object	監査ログレコードを作成します。
Authenticate User Credentials	
Authorize Object	リポジトリ内のオブジェクトに対して、対象となる認証をテストします。
Checkin Object	オブジェクトに変更を適用します。
Checkin View	更新されたビューを適用します。
Checkout Object	リポジトリのオブジェクトをロックし、編集のために取得します。
	延期タスクのスキャナ情報をオブジェクトに追加します。
Checkout View	更新可能なビューを取得します。
Create Resource Object	リソースオブジェクトを作成します。
Create View	新規ビューを初期化します。
Delete Resource Object	リソースオブジェクトを削除します。
Deprovision Primitive	リソースアカウントをプロビジョニング解除します。
Disable Primitive	リソースアカウントを無効にします。
Disable User	Identity Manager のユーザーアカウント、リソースアカウント、またはその両方を無効にします。
Email Notification	アクションを通知する電子メールを送信します。
Enable Primitive	リソースアカウントを有効にします。
Enable User	Identity Manager のユーザーアカウント、リソースアカウント、またはその両方を有効にします。
Get Object	リポジトリオブジェクトを取得します。
Get Property	プロパティーを取得します。
Get View	読み取り専用ビューを取得します。
List Resource Objects	

表 1-4　ワークフローのデフォルトアクティビティー (続き)

アクティビティー	説明
Query Object Names	一致する属性を持つオブジェクトを検索します。
Query Objects	一致する属性を持つオブジェクトを検索します。
Query Reference	
Refresh View	以前にチェックアウトされたビューを更新します。
Remove Deferred Task	延期タスクのスキャナ情報をオブジェクトから削除します。
Remove Property	オブジェクトの拡張プロパティーを削除します。
Reprovision Primitive	リソースアカウントを再プロビジョニングします。
Run Resource Actions	
Set Property	拡張プロパティーをオブジェクトに追加します。
Unlock Object	以前にチェックアウトされたオブジェクトのロックを解除します。
Unlock View	以前にチェックアウトされたビューのロックを解除します。
Update Resource Object	リソースによって管理されるオブジェクトを修正します。

表 1-5　Approval ワークフローのデフォルトアクティビティー

アクティビティー	説明
Approval	基本的な単一の承認者プロセスを実行します。
Approval Evaluator	複雑な承認プロセスを実装するために、承認定義オブジェクトの評価を繰り返します。 使用するフォームとテンプレートの受け渡しは許可されますが、別の設定が指定されている場合は、そちらが優先されます。
Lighthouse Approval	割り当てられている組織、ロール、およびリソースの、デフォルトの Identity Manager 承認プロセスを実行します。Approval Evaluator プロセスを使用します。
Multi Approval	複数の承認者に承認を配布します。各承認者の Approval プロセスを使用します。

表 1-5　Approval ワークフローのデフォルトアクティビティー (続き)

アクティビティー	説明
Notification Evaluator	複雑な通知プロセスを実装するために、承認定義オブジェクトの評価を繰り返します。通常は、Approval Evaluator に定義されている構造と同じ構造です。標準ワークフローでは、承認定義と通知定義は同じ構造にあります。カスタマイズされたワークフローでは、これは要件となりません。
Provisioning Notification	プロビジョニング処理が完了したあとに管理者に通知するための標準プロセス。

表 1-6　User ワークフローのデフォルトアクティビティー

アクティビティー	説明
DeProvision	既存の Identity Manager ユーザーをプロビジョニング解除するための標準ステップを実行します。リソースアカウントの削除、Identity Manager ユーザーの削除、リンク解除、および割り当て解除を詳細に制御できます。個々のリソース操作は、成功するまで再試行されます。
Provision	新しい Identity Manager ユーザーを作成し、リソースアカウントをプロビジョニングする標準ステップを実行します。個々のリソース操作は、成功するまで再試行されます。
Set Password	Identity Manager アカウントとリソースアカウントのパスワードを変更します。
Update User Object	WSUser オブジェクトをチェックアウトし、変更内容を適用してからチェックインします。
Update User View	ユーザービューをチェックアウトし、提供される一連の更新を適用してからチェックインします。
Update View	任意のビューに一連の変更を適用します。

表 1-7　End User ワークフローのデフォルトアクティビティー

アクティビティー	説明
End User Update Groups	マネージャーのいずれかのレポートに割り当てられるリソースのグループ割り当てを更新します (グループをサポートするリソースが対象)。

表 1-7　End User ワークフローのデフォルトアクティビティー (続き)

アクティビティー	説明
End User Update My Groups	ログインしているアカウントに割り当てられるリソースのグループ割り当てを更新します (グループをサポートするリソースが対象)。
End User Update Roles	マネージャーのいずれかのレポートのロール割り当てを更新します。
End User Update My Roles	ログインしているアカウントに割り当てられるロール割り当てを更新します。
End User Update Resources	マネージャーのいずれかのレポートの、リソース割り当てと、関連付けられている属性を更新します。
End User Update My Resources	ログインしているアカウントの、リソース割り当てと、関連付けられている属性を更新します。

表 1-8　Compliance ワークフローのデフォルトアクティビティー

アクティビティー	説明
Access Review Remediation	1 つのユーザーエンタイトルメントを操作する 1 人の是正者のための是正。
Attestation	各アテスターの作業項目を作成し、すべての作業項目が承認済みの状態で完了した場合は、ユーザーのエンタイトルメントレコードに APPROVED のマークを付け、いずれかの作業項目が拒否された場合は、その時点でただちに REJECTED のマークを付けます。1 つの作業項目が拒否されると、その他すべての作業項目はキャンセルされます。
Launch Access Scan	Access Review タスクから得られる設定に基づいて、Access Scan タスクを呼び出すか、またはスケジューリングします。このアクティビティーは、Access Review ワークフロー / タスクから直接呼び出されます。
Launch Entitlement Rescan	1 人のユーザーのためにアクセススキャンの再スキャンを起動します。
Launch Violation Rescan	1 人のユーザーのために監査ポリシースキャンの再スキャンを起動します。
Multi Remediation	1 つのコンプライアンス違反と複数の是正者のための是正。
Remediation	1 つのコンプライアンス違反を操作する 1 人の是正者のための是正。

表 1-8　Compliance ワークフローのデフォルトアクティビティー (続き)

アクティビティー	説明
Scan Notification	各アクセススキャンの最後に、保留中のアテステーション作業項目があることをアテスターに通知します。保留中の作業項目の数に関係なく、各アテスターに 1 つの通知を送信します。また、スキャンに所有者が存在する場合は、スキャンの開始と完了をその所有者にも通知します。このワークフローは、次の入力をとります。 *scanName* -- アクセススキャンの名前 *scanOwner* -- アクセススキャンの所有者の名前 *recipients* -- 通知先となる Identity Manager ユーザーの名前リスト *notificationType* -- 通知タイプ (begin、end、attest などのタイプが有効) *userCount* -- スキャン対象ユーザーの数 (begin のみ)
Standard Attestation	指定された各アテスターのアテステーションサブプロセスを作成します。
Standard Attestation	指定された各アテスターのアテステーションサブプロセスを作成します。
Test Auto Attestation	アテステーション作業項目を作成せずに新しいレビュー決定規則をテストできるようにします。このワークフローは、作業項目を作成せず、開始してすぐに終了するだけです。すべてのユーザーエンタイトルメントオブジェクトは、アクセススキャンによって作成されたときと同じ状態のままです。terminate オプションと delete オプションを使用すると、このワークフローで実行されたアクセススキャンの結果がクリーンアップされます。
Update Compliance Violation	コンプライアンス違反を受け入れます。

スキャンタスク変数

監査ポリシースキャンタスクおよびアクセススキャンタスクのタスク定義には、タスクの開始時に使用するフォームを指定します。これらのフォームには、制御する必要があるスキャンタスク変数の、全部ではないが大部分に対応するフィールドが含まれています。

表 1-9 スキャンタスク変数

変数名	デフォルト値	目的
maxThreads	5	1 つのスキャナで 1 度に作業する同時ユーザー数を指定します。この値を大きくすると、非常に低速なリソース上にアカウントを持つユーザーをスキャンするときのスループットが向上する可能性があります。
userLock	5000	スキャンするユーザーのロックを取得するために消費する時間 (ミリ秒単位) を指定します。1 度に複数のスキャンによって同じユーザーがスキャンされ、そのユーザーのリソースが低速である場合は、この値を大きくすることでロックエラーを減らすことができますが、スキャン全体の速度は低下します。
scanDelay	0	新しいスキャンスレッドを発行する間隔 (ミリ秒単位) を指定します。正の数にすると、スキャナによる CPU 負荷を減らすことができます。

ワークフロータスク

表 1-10

アクティビティー	説明
Add Result	名前を付けたデータ項目をタスク結果に追加します。
Add Result Error	タスク結果にエラーメッセージを追加します。
Add Result Message	タスク結果に情報メッセージを追加します。
Background Task	Identity Manager の管理者インタフェースから呼び出された場合に、ワークフローを強制的にバックグラウンドで実行します。
Get Resource Result	最後のプロビジョニング操作でリソースアダプタから返された結果オブジェクトを取得します。
Get Resource Result Item	最後のプロビジョニング操作でリソースアダプタから返された結果オブジェクトから、1 つの結果項目を取得します。

表 1-10　　(続き)

アクティビティー	説明
Rename Task	リポジトリ内のタスクインスタンスの名前を変更します。
Scripted Task Executor	渡されたスクリプトに基づいて BeanShell または JavaScript を実行します。タスクとして定期的に実行するようにスケジューリングできます。たとえば、リポジトリからデータベースにデータをエクスポートして報告や分析を行うときなどに使用できます。カスタム Java コードを作成しなくてもカスタムタスクを作成できることも利点です。カスタム Java コードはアップグレードのたびに再コンパイルし、すべてのサーバーに配備する必要がありますが、スクリプトはタスクに埋め込まれるため、再コンパイルや配備の必要はありません。
Set Result	タスクに入力される結果にエントリを追加します。これは、ワークフローの概要レポートに記載されます。
Set Result Limit	終了したタスクインスタンスをリポジトリ内で維持する時間を、秒単位で指定します。正の値は、タスクの完了後に、その秒数の間だけタスクインスタンスを維持することを表します。 負の値は、タスクインスタンスが自動的に削除されないことを表します。ただし、タスクを手動で削除することは可能です。

デフォルトの名前変更タスクの使用

カスタマイズを行わずにデフォルトの名前変更タスクを使用する場合は、ワークフローに次のアクションを指定します。

```
<Action process='Rename Task'>
   <Argument name='name' value=' 新しいタスク名 '/>
</Action>
```

ワークフローの進行状況の追跡

タスクに指定されている所有者は、ワークフロータスクの状態を常に確認できます。多くの場合、所有者はタスクを開始した人物ですが、所有者情報を定義し直すことができます。タスクはリポジトリ内のオブジェクトであるため、適切な権限を持つユーザーであれば、誰もがそれを表示できます。

通常、ワークフローの状態は **executing**、**pending**、**creating**、および **suspended** という文字列で「状態」列に表示されます。ワークフローの状態を要約した、より説明的な文字列をこの列の表示に追加できます。

この機能を実装するには、追加可能な 2 つの式のいずれかを WFProcess ファイルに追加します。

コード例 1-2

```
<WFProcess name='queryRoleTask' maxSteps='0'>
   <Status>
     <s>Customized Status</s>
   </Status>
     <Activity id='0' name='start'>
       <Transition to='GetReferencingRoles'/>
     </Activity>
     <Activity name='GetReferencingRoles'>
       <Action id='0'>
     <expression>
<Status> には、文字列となる任意の XPRESS ステートメントを設定できます。次に例を示します。
<Status>
   <s> カスタム文字列 </s>
</Status>
または
<Status>
   <block>
     <s> 表示されない </s>
     <s> カスタム文字列 </s>
   </block>
</Status>
```

この例の式の結果は、条件が当てはまる結果が保留中の場合に、「状態」列に表示されます (たとえば、**pending (カスタムステータス)**)。

ワークフロープロパティーの設定

ワークフローの設定プロパティーは、System Configuration オブジェクトによって制御されます。次の表は、頻繁に使用される設定プロパティーを示しています。

表 1-11 システム設定オブジェクトのワークフロープロパティー

属性名	データ型	デフォルト値
workflow.consoleTrace	String	false
workflow.fileTrace	null	
workflow.maxSteps	String	5000
workflow.resultTrace	String	false
workflow.retainHistory	String	false
workflow.traceAllObjects	String	false
workflow.traceLevel	String	1
workflow.validationLevel	String	CRITICAL
serverSetting.default.reconciler		

consoleTrace

コンソールにトレースメッセージを送信するかどうかを指定します。true に設定した場合は、トレースメッセージが送信されます。デフォルトは false です。

fileTrace

名前が付けられたファイルにトレースメッセージを送信するかどうかを指定します。特定のファイルにトレースメッセージを送信するには、そのファイルの名前を入力します。

maxSteps

ワークフロープロセスまたはサブプロセスで許容される、最大ステップ数を指定します。このレベルを超過すると、その時点で Identity Manager はワークフローを終了します。この設定は、無限ループによるワークフローのスタックを検出するための安全装置として使用されます。ワークフロー自体に設定されたデフォルト値は 0 です。SystemConfiguration オブジェクトの workflow.maxSteps 属性に格納されたグローバル設定から実際の設定値が抽出されることはずであることを示しています。このグローバル設定の値は 5000 です。

resultTrace

タスクの結果オブジェクトにトレースメッセージを維持するかどうかを指定します。true に設定した場合は、タスクの結果オブジェクトにトレースメッセージが累積されます。

retainHistory

タスクの完了時に、履歴全体を返すかどうかを指定します。true に設定した場合は、Identity Manager は、ケース履歴全体を返します。これは、履歴を分析してプロセスの問題を診断する場合には便利ですが、結果全体のサイズが大きくなります。

traceAllObjects

汎用オブジェクトをワークフローのトレース処理の対象とするかどうかを指定します。

traceLevel

ワークフローのトレースに含める詳細度を指定します。値を指定しないか、または 0 を指定した場合は、もっとも詳細な情報がトレースされます。デフォルトは 1 です。

validationLevel

実行前にワークフローの検証に適用される厳密度を指定します。設定したレベル以上のエラーが検出された場合は、ワークフローの実行は開始されません。有効な値は、CRITICAL、ERROR、WARNING、または NONE です。NONE を指定した場合は、検証を完全に無効にします。

デフォルトは CRITICAL です。

serverSettings.default.reconciler 属性

次の 2 つの調停サーバー属性を使用して、アカウント単位ワークフローを調整できます。

- `maxRetries` - 失敗した調整要求の最大再試行回数を指定します。デフォルト値は、3 です。この値を大きくすると、調整の回数が増えますが、失敗した要求が最終的に成功する可能性も高くなります。
- `lockwaitThreshold` - 調整要求の最小再試行間隔 (ミリ秒) を指定します。デフォルトは 5000 ミリ秒 (5 秒) です。この値を大きくすると、失敗した要求を再試行するのに必要な時間が増えますが、未処理の問題を解決するために使用できる時間も増えます。

これら 2 つの値の積が、ユーザーのアカウント単位ワークフローが成功するのに必要な時間より大きくなるようにしてください。この間隔の間に、調停サーバーの内部でロック競合の問題が解決されます。

次の例を考えてみましょう。

- アカウント単位ワークフローが完了するのに平均 30 秒かかる

- maxRetries の値は 3
- lockwaitThreshold の値は 5000

再試行に消費される合計時間は 15 秒 (3 x 5000 ミリ秒) です。

再試行に消費される合計時間 (15 秒) が 30 秒より小さいため、この設定は最適ではありません。この例のユーザーは、lockwaitThreshold と maxRetries の値を大きくして、その積がアカウント単位ワークフローの合計時間より大きくなるようにするべきです。

ワークフローサービスの使用

Identity Manager には、ユーザーアカウントのプロビジョニングと操作を目的とするプロセスを定義するために、デフォルトワークフローが用意されています。Identity Manager をカスタマイズするときは、これらのワークフローを変更して、配備環境に固有のビジネスルールを反映することができます。ワークフローを使用することで、アカウントプロビジョニングに関する独自のビジネスルールを Identity Manager に実装できます。

ここでは、ワークフローサービスに関連する次のトピックの概要について説明します。

- メソッドのコンテキスト
- 組み込みのワークフローサービス変数
- デフォルトのプロビジョニングメソッド
- デフォルトのセッションメソッド
- ビューの操作
- 使用例

特定のワークフローメソッドについては、<*distribution*>\REF\javadoc (<*distribution*> はインストールディレクトリ) を参照してください。

メソッドのコンテキストについて

Lighthouse コンテキストは Identity Manager リポジトリにアクセスするための認証済みセッションを表現しているので、可視性およびアクションの制限を適用するための認証チェックが必要となります。ユーザーやその他の Identity Manager オブジェクトを作成、変更、および削除するには、リポジトリプロビジョニングツールへの認証済みアクセス権が必要になります。このアクセス権は、コンテキストオブジェクト (通常は LighthouseContext または WorkflowContext) によって確立されます。これらの各コンテキストオブジェクトには、呼び出し側にリポジトリへのアクセス権を与える認証済みセッションオブジェクトが含まれています。

ログインした環境 (Web ブラウザ内) のコンテキストで操作しているときのコンテキスト (セッション) はかなり単純です。しかし、Identity Manager の内部では、ワークフロープロセスはブラウザセッションから独立した別個のプロセス (実際には TaskInstance) であるので、Identity Manager リポジトリにアクセスする必要があります。これは、実行中のワークフローのアクティブコンテキストはワークフローの開始時に割り当てられ、ワークフローが中断しても保持され再開時に復元されるためです。

ユーザーが (通常は WorkItem または ManualAction で) ワークフローと対話するときは、ワークフロー独自のコンテキストは保持されます。作業項目と対話するユーザーのコンテキストは想定されていません (ユーザーには、作業項目への適切なアクセス権を与えるコンテキストが必要です)。作業項目と対話するユーザーがフォームを読み込むと、Identity Manager はワークフローのコンテキストではなく、ユーザーのコンテキストでフォームを処理します。より正確に言えば、作業項目と対話するユーザーはワークフローとはまったく対話しません。ユーザーは作業項目とだけ対話します。ユーザーが作業項目を変更したあとは、スケジューラが必要に応じてワークフローを再起動します。

ワークフローの組み込み変数

ワークフローエンジンは、いくつかの組み込み変数を使用します。これらの変数のほとんどは、ワークフロー内で宣言する必要はありません。しかし、組み込み変数はワークフローの実行状態を確認するために使用できます。また、多くの変数は、ワークフローサービスの影響を受けて設定されます。

注 ワークフロー変数の大文字と小文字は区別されます。たとえば、WF_ACTION_ERROR は wf_action_error と同じではありません。

表 1-12　ワークフローの組み込み変数

名前	説明
WF_ACTION_ERROR	この組み込み変数は、直前に実行されたアクションがエラーを含む結果を返したり例外をスローしたりした場合に `true` に設定されます。
WF_ACTION_RESULT	この組み込み変数には、直前のアクションから返される WavesetResult オブジェクトが設定されます。この変数は、アクションの WavesetResult を取り込み、それをグローバル TaskInstance の結果に追加せずに処理したい場合に使用します。この変数はもともと、リソースの再試行をサポートするために追加されました。再試行のたびにタスク結果にリソースのエラーメッセージを追加することは必ずしも望ましくはないためです。あまり頻繁に使用される変数ではありませんが、アクションの結果をタスク結果に追加する前に調整する場合には便利です。
WF_ACTION_SUPPRESSED	この組み込み変数は、`<Condition>` 式の評価が false であるためにアクションが実行されない場合に `true` に設定されます。
WF_ACTION_TIMEOUT	この組み込み変数は、直前に実行されたアクションがタイムアウトになった場合に `true` に設定されます。
WF_CASE_OWNER	この組み込み変数には、ワークフロータスクを呼び出した管理者の名前が設定されます。
WF_CASE_RESULT	この組み込み変数には、TaskInstance の WavesetResult が設定されます。これは XPRESS または JavaScript で実装されるアクションにおいて結果を取得するために使用されます。WorkflowApplication クラスとして実装されるアクションは WorkflowContext を経由して結果を取得することが可能です。WorkflowContext の全体は WF_CONTEXT 変数を使って参照できるため、この変数は絶対に必要であるとは言えませんが、便利ではあります。
WF_CONTEXT	この組み込み変数には、WorkflowContext オブジェクトが設定されます。これは XPRESS または JavaScript で実装されるアクションにおいて WorkflowContext を取得するために使用できます。WorkflowApplication クラスとして実装されているアクションに対して、コンテキストが渡されます。

一般的なセッションワークフローサービスの呼び出しの構造

ワークフローの内部には、制御の流れと変数の範囲の両方を制限する階層構造があります。ワークフローは Case または WFCase とも呼ばれ、ワークフローアクティビティーのリストが含まれています。ワークフローアクティビティーには、Action 要素のリストが含まれます。WFCase で宣言される変数は、すべての Activity 要素および Action 要素で参照できます。`color` という名前の変数を WFCase レベルで宣言し、同時に Action でも宣言した場合、これらは実質的に異なる 2 つの変数です。たとえば、Action 内の `color` 変数の値を変更しても、WFCase の `color` 変数の値には影響しません。

ワークフローサービスは、ワークフローアクションから呼び出されます。WorkflowServices は、*op* 引数の値によって選択された一連の操作を提供します。各操作の引数はそれぞれ異なるため、呼び出し側の「署名」が指定されたサービスと一致する必要があります。次のコード例は、ワークフローサービスアクションの一般的な形式を示しています。

```
<Action class='com.waveset.session.WorkflowServices'>
   <Condition/>
   <Argument name='op' value=workflowServiceOp/>
   <Argument name=argname1>
      <expression>value1expression</expression>
   </Argument>
   <Argument name=argname2>
      <expression>value2expression</expression>
   </Argument>
   <Argument name=argnameN>
      <expression>valueNexpression</expression>
   </Argument>
</Action>
```

サポートされる各ワークフローサービスには、それぞれ異なる数の、必須および省略可能な引数があります。セッションワークフローサービスの呼び出しに渡す *op* 引数には、提供されているサービスのいずれかを指定する必要があります。これは、リフレクションによるメソッド呼び出しに似ています。ここでは、呼び出されるメソッドの名前が、実行されるワークフローサービスの名前に相当します。

後続のリストにない *op* 引数が渡された場合、ワークフローサービスは次の結果を返します。

```
'Unknown WorkflowServices op'
```

また、ワークフローコンテキスト変数 *WF_ACTION_ERROR* は、NULL 以外の値に設定されます。

先行のリストにない *op* 引数が渡された場合は、ワークフローサービスは次の結果を返します。

```
'Unknown WorkflowServices op'
```

また、ワークフローコンテキスト変数 *WF_ACTION_ERROR* は、NULL 以外の値に設定されます。

プロビジョンワークフローサービス

`com.waveset.provision.WorkflowServices` クラスにもサービスのセットがありますが、これらのサービスは `com.waveset.session.WorkflowServices` のメソッドほど使用頻度が高くありません。これらのサービスは、アカウント管理を実行するための低レベルの基本サービスで、主にデフォルトワークフローによって呼び出されます。カスタムワークフローでは、これらのサービスを使用することもできますが、`checkoutView` と `checkinView` によってより高レベルのビューを使用し、これらのビューから標準のワークフローを起動することもできます。

プロビジョンサービスの主な目的は、プロビジョニングツールへのアクセス権をワークフローに与えることで、リソースおよびリソースアカウントにアクセスできるようにすることです。これらのサービスは、リソースそのものに実際に読み込み / 書き込み操作を実行します。

一般的なプロビジョンワークフローサービスの呼び出しの構造

ワークフローの内部には、制御の流れと変数の範囲の両方を制限する階層構造があります。ワークフローは Case または WFCase とも呼ばれ、ワークフローアクティビティーのリストが含まれています。ワークフローアクティビティーには、アクションのリストが含まれています。WFCase で宣言される変数は、すべての Activity 要素および Action 要素で参照できます。「color」という名前の変数を WFCase レベルで宣言し、同時にアクションでも宣言した場合は、実質的に 2 つの異なる変数が存在することになります。たとえば、アクション内の「color」変数の値を変更しても、WFCase の「color」変数の値には影響しません。ワークフローサービスは、ワークフローアクションから呼び出されます。WorkflowServices は、「op」引数の値によって選択された一連の「操作」を提供します。各操作の引数はそれぞれ異なる可能性があるため、呼び出し側の「署名」がサービスと一致する必要があります。次のコード例は、セッションワークフローサービスアクションの一般的な形式を示しています。

```
<Action class='com.waveset.provision.WorkflowServices'>
   <Condition/>
   <Argument name='op' value=workflowServiceOp/>
   <Argument name=argname1>
      <expression>value1expression</expression>
   </Argument>
   <Argument name=argname2>
      <expression>value2expression</expression>
   </Argument>
            ...
   <Argument name=argnameN>
      <expression>valueNexpression</expression>
   </Argument>
</Action>
```

サポートされる各ワークフローサービスには、それぞれ異なる数の、必須および省略可能な引数があります。

サポートされるプロビジョンワークフローサービス

次のリストは、Identity Manager が現在サポートしているプロビジョンワークフローサービスを示しています。ワークフローサービスの呼び出しに指定する *op* 引数は、次のいずれかの値である必要があります。

表 1-13

プロビジョンワークフローサービス	説明
approve	ユーザーアカウントの承認を記録します。
auditNativeChangeToAccountAttributes	リソースアカウントの 1 つまたは複数の監査可能属性に加えられたネイティブ変更のレポートを生成します。
authenticateUserCredentials	パスワードを使用して、リソースに対しユーザーを認証します。
bulkReprovision	このメソッドは、クエリーのセットを実行し、指定した条件と一致するすべてのユーザーを検索します。その後、このリストを繰り返し、ユーザーを一度に再プロビジョニングします。
changeResourceAccountPassword	1 つまたは複数のリソースアカウントのパスワードを変更します。

表 1-13　(続き)

プロビジョンワークフローサービス	説明
checkDeProvision	削除前に、アカウントがプロビジョニング解除を必要とするかどうかを調べます。
cleanupResult	タスク結果から NULL の ResultError を削除します。このメソッドは、引数として *op* をとります。有効な値は、cleanupResult です。
createResourceObject	リソースオブジェクト (たとえば、グループ) を作成します。
deleteResourceAccount	リソースアカウントを削除します。
deleteResourceObject	リソースオブジェクト (たとえば、グループ) を削除します。
deProvision	Identity Manager アカウント、リソースアカウント、またはその両方を削除します。
deleteUser	Identity Manager ユーザーを削除します。
disable	Identity Manager アカウント、リソースアカウント、またはその両方を無効にします。
enable	Identity Manager アカウント、リソースアカウント、または両方を有効にします。
getApprovals	既存のアカウントに割り当てられたロール、組織、およびリソースの承認のリストを決定します。
getApprovers	
lockOrUnlock	指定されたユーザーに関連付けられている Lighthouse アカウントポリシーがロックの有効期限の時刻を指定している場合に、そのユーザーをロックまたはロック解除します。
notify	通知を送信します。ほとんどの場合は、電子メールが使用されます。
provision	新しい Identity Manager アカウントと、(オプションとして) リソースアカウントを作成します。
questionLock	ユーザーをロックしますが、ロックの有効期限の日時は設定しません。
reject	リソースアカウントの却下を記録します。
reProvision	既存の Identity Manager アカウントを更新します。
runResourceAction	リソースの指定されたリソースアダプタに対して、リソースアクションを実行します。
updateResourceObject	
unlinkResourceAccountsFromUser	

表 1-13　(続き)

プロビジョンワークフローサービス	説明
validate	

op 引数に上記以外の値を指定した場合、ワークフローサービスは次の結果を返します。

```
'Unknown WorkflowServices op'
```

また、ワークフローコンテキスト変数 WF_ACTION_ERROR は、NULL 以外の値に設定されます。

ビュー操作メソッドについて

ほとんどのワークフロー処理にはビューが使用されます。このため、ワークフローで使用されるビュー関連のメソッドでもっともよく使用されるのは checkoutView メソッドと checkinView メソッドになります。ビューオブジェクトをチェックアウトすると、基本となるオブジェクトがロックされ、ほかのユーザーが書き込めなくなります。これは、ワークフロー処理では特に重要です。ワークフローがユーザーをチェックアウト (ロック) し、手動アクションのために処理を中断すると、手動アクションが完了するまで (数時間後または数日間後の可能性がある) そのユーザーがロックされたままになる可能性があるためです。デフォルトでは、オブジェクトに対するロックの有効期限は 5 分であり、これがワークフローに関する 2 つ目の懸念事項です。オブジェクトをチェックインするには、オブジェクトが呼び出し側によってすでにロックされている必要があります。したがって、ワークフローがビューをチェックアウトし (暗黙的にオブジェクトがロックされる)、処理を 10 分間中断してからビューをチェックインすると、10 分間が経過して呼び出し側のロックがすでに終了しているため、ワークフローが失敗します。

次の例は、一般的なチェックアウト操作を示しています。

コード例 1-3

```
<Action id='-1' application='com.waveset.session.WorkflowServices'>
   <Argument name='op' value='checkoutView'/>
   <Argument name='type' value='User'/>
   <Argument name='id' value='mfairfield'/>
     <Variable name='view'/>
   <Return from='view' to='user'/ >
</Action>
```

チェックアウトおよびチェックイン呼び出しとオプションのマップの使用

チェックアウトおよびチェックイン呼び出しのオプション引数として (UserViewConstants クラスの一部として定義される)、どのオプションを使用できるかを判断するのが難しいことがあります。Javadoc には、次の形式でオプションが記載されています。

```
OP_TARGETS

OP_RAW

OP_SKIP_ROLE_ATTRS
```

Identity Manager では、オプションを確認するときにこれらのリテラル文字列をコードにハードコードする代わりに、コード全体で使用できるように文字列を表現する定数が用意されています。これはよいコーディング手法ですが、XPRESS やワークフローでは UserViewConstants の静的フィールド (OP_TARGET_RESOURCES など) を参照できません。

正しい値を渡すことができる有効な文字列であることを調べるために、正しい文字列であることを確認するためテスト規則を作成できます。たとえば、次の規則は `TargetResources` を返します。

コード例 1-4

```
<block>
   <set name='wf_srv'>
      <new class='com.waveset.provision.WorkflowServices'/>
   </set>

   <script>
      var wfSvc = env.get('wf_srv');
      var constant = wfSvc.OP_TARGETS;
      constant;
   </script>
</block>
```

この規則は文字列を調べるのに便利ですが、すべての呼び出しに同じ文字列を返すため、本稼働配備には適していません。この問題を最小限に抑えるため、ビュー処理に使用される Identity Manager 定数が変更されることはありません。定数をワークフローにコーディングしたあとは、リリースが異なっても、ビューによるその定数の解釈は変わりません。

有効な文字列であることがわかったら、ビューのチェックアウト呼び出しを次のように更新できます。このコードでは、Identity Manager および Active Directory に変更が反映されるだけのビューがチェックアウトされます。

コード例 1-5

```
<Action id='-1' application='com.waveset.session.WorkflowServices'>
   <Argument name='op' value='checkoutView'/>
   <Argument name='type' value='User'/>
   <Argument name='id' value='mfairfield'/>
   <Argument name='options'>
      <map>
       <s>TargetResources</s>
       <list>
          <s>Lighthouse</s>
          <s>AD</s>
       </list>
     </map>
   </Argument>
<Variable name='view'/>
<Return from='view' to='user'/ >
</Action>
```

ベストプラクティス

パフォーマンス上の観点から、可能な場合はユーザービューのサイズを制限することをお勧めします。ビューを小さくすると、リソースから取得されてネットワークに送信されるデータが少なくなります。ビューはワークフロー内に変数として保持されるため、ワークフロータスクが中断されるたびに TaskInstance オブジェクトにビューを書き込む必要があります。システムが数千個のワークフローを同時に実行していて、各ワークフローに大きなビューがある場合は、リポジトリとの間でビューを転送するのにかかる時間 (直列化および非直列化を含む) が大きなボトルネックになります。

ビューの最適な使用方法は、ユーザーへの変更をインテリジェントに実行して承認するために必要なデータだけを保持することです。呼び出し側の目的が一連のリソース上にあるユーザーのパスワードを変更することである場合は、このプロセスを実行する上で知る必要があるものは、そのユーザーに関連付けられたアカウントのリストだけです。通常、このプロセスに特定のアカウントのデータは必要ありません。

シナリオ例 1

ある顧客が、ユーザーが特定のリソースへのアクセスを要求できるようなカスタムワークフローを実装することにしました。このワークフローは、適切な承認が得られるまでの間、変更を送信できるようにユーザービューをチェックアウトするはずです。この例では、取得する必要がある情報はビューの Identity Manager ユーザー部分だけで、`waveset.resources` リストおよび `accountInfo` オブジェクトがそれに合わせて更新されるようです。このような場合は次のようにして、ユーザービューをチェックアウトするときに `TargetResources` オプションを使用して、ユーザービューの Identity Manager ユーザー部分とオプションマップだけがチェックアウトされるようにします。

コード例 1-6

```
<map>
   <s>TargetResources</s>
   <list>
      <s>Lighthouse</s>
   </list>
</map>
```

作業項目ビューでは、ユーザービューのサイズを小さくできます。作業項目とは、承認、委任、ユーザーが追加情報を入力するためのワークフロー内での中断などの、個人に割り当てられたタスクを表現するオブジェクトのことです。カスタムワークフローでは手動アクションによって、承認が作成され、フォームとの対話が行われます。手動アクションとは、簡単にいえば、現在の <ManualAction> 要素の範囲内にあるすべてのワークフロー変数をコピーすることです。タスクのコンテキストだけでなく、WorkItem オブジェクト内の変数オブジェクトにコピーします。承認に関して、完全なユーザービューやコンテキスト変数が必要になることはほとんどありません。このため、ManualAction を定義するときに ExposedVariables 引数を指定することにより、WorkItem のサイズを小さくすることを検討してください。

<Exposed Variables> 要素により、後続処理の作業項目に取り込む必要がある変数がワークフローに正確に伝わります。1 つのワークフローが複数の承認者をサポートする場合は、承認者ごとに 1 つ、複数の作業項目が作成されることに注意してください。ワークフローに 10 人の承認者がいれば、メモリーが 100K バイトずつ消費されてゆき、すぐに実際のメモリー量を検討することになります。

シナリオ例 2

前述のシナリオは、手動アクションが 1 つだけの比較的単純な実装を示しています。しかし、一般的な配備にはより複雑なシナリオが必要になります。たとえば、ある顧客は、ユーザーが作成されると承認を「バケット」に入れ、約 25 人の承認者のいずれかが承認を選択できるようにする必要があります。バケットを模倣するには、バケット内に承認者ごとの作業項目を作成します。承認者が行う作業は、アクションの受け入れだけです。受け入れが終わると、Identity Manager は残りの 24 個の作業項目を破棄することができ、バケット要求を受け入れた承認者の実際の承認作業項目を生成します。

エスカレーション層として機能する 3 つのバケットがあるとします。バケット 1 で承認が処理されると、Identity Manager は新しい要求をバケット 2 にエスカレーションし、再びこのプロセスが始まります。このプロセスにより、バケット 1 つあたりおよそ 26 (25 + 1) 個の作業項目が作成され、3 つのバケットで 3 倍になります。1 つのユー

ザー要求によって作成される作業項目は 78 個で、しかもすべて一度に作成されるとは限りません。しかし、500,000 人規模のユーザーベースを持ち、各ユーザーがこれらの要求を毎日数百個も作成する配備環境でこれが発生することを考えてください。このシナリオの実装者が *ExposedVariables* 引数を使用して WorkItem ビューのサイズを慎重に決定しないと、大量の不要なデータが作業項目に格納され、JDBC 接続を介して渡されることになります。

コード例 1-7

```
<Activity id='0' name='activity1'>
<ManualAction id='-1'>
 <FormRef>
 <ObjectRef type='UserForm' name='Access Review Abort Confirmation Form'/>
</FormRef>
<ExposedVariables>
<List>
<String>user.waveset.accountId</String>
<String>user.waveset.resources</String>
 <String>user.accounts[Lighthouse].idmManager</String>
<String>user.accounts[Lighthouse].fullname</String>
</List>
   </ExposedVariables>
     <ViewVariables>
     <List>
<String>user</String>
</List>
</ViewVariables>
</ManualAction>
<WorkflowEditor x='53' y='111'/>
</Activity>
```

シナリオ例 3

ユーザープロビジョニング要求のカスタムワークフローを作成し、ユーザーが実行できるようにする必要があります。このタイプのワークフローは、部下のプロビジョニング要求をマネージャーに送信させ、かつ組織内のすべてのマネージャーを Identity Manager 管理者にしないことを希望する顧客のときにお勧めします。

これらの要件に留意して、次のようなカスタムワークフローを作成します。

- マネージャーが部下を作成または更新することを許可する
- チェックアウトされたビューや新規作成されたビューをマネージャーが編集できるようにするための ManualAction を組み込む
- チェックインまでに、プロセスが適切な承認処理を (カスタム手動アクションを使用して) すでに完了しているか、または (既存の承認処理を使用して) 完了することを保証する

ビューも更新する必要があります。ビューを更新すると、Identity Manager は割り当てられているリソースの情報を再検査し、ネイティブリソースからその情報を再取得して、最新の情報でユーザービューを更新します (ビューがチェックアウトされてから変更されている場合)。

Identity Manager でビューが確実に更新されるようにするために、手動アクション内の `ViewVariables` 要素に、Identity Manager が作業項目の `variables` オブジェクトに含まれるどの変数をビューとして扱い、要求時に更新すべきかを指定します。変数 *user* を ViewVariable として指定する例については、前述の例を参照してください。

ワークフローの監査の有効化

ワークフローの監査は、通常の監査と類似していますが、ワークフローの監査では、時刻を計算するための追加情報も記録されます。通常の監査では、イベントの発生をレポートしますが、イベントがいつ開始され、終了したのかは示されません。Identity Manager の監査の詳細については、『Identity Manager 管理ガイド』を参照してください。

ワークフロー監査の処理では、事前に定義されている属性の名前とその値が、監査対象イベントごとに記録されます。この機能を使用してワークフロー、プロセス、およびアクティビティーの最初と最後に `auditWorkflow` イベントを配置することによって、ワークフローを計測することができます。

注 ワークフローまたはプロセスを計測するときに、終了アクティビティーにイベントを配置することは許可されていません。最後の `auditWorkflow` イベントを実行してから終了イベントに無条件に遷移する、終了前アクティビティーを作成する必要があります。

ワークフローの監査を有効にするには、ワークフロー、または 1 つまたは複数のワークフローアクティビティーに、`audit` 属性を追加します。この属性を追加し、管理者インタフェースの適切なタスクテンプレートで「ワークフロー全体の監査」ボックスにチェックマークを付けることで、ワークフローの監査が有効になります。タスクテンプレートで監査を有効にする手順については、『Identity Manager 管理ガイド』の「タスクテンプレート」の章を参照してください。

また、`auditWorkflow` イベントのペアリングを可能にするために `auditconfig.xml` ファイルに定義されている次のアクションに、適切な値を渡す必要があります。追加のアクションを定義してもかまいません。

- `StartWorkflow`
- `EndWorkflow`

- StartProcess
- EndProcess
- StartActivity
- EndActivity

次の呼び出し例は、呼び出し側から渡す必要のある情報だけを示しています。

```
<Action application='com.waveset.session.WorkflowServices'>
   <Argument name='op' value='auditWorkflow'/>
   <Argument name='action' value='StartWorkflow'/>
</Action>
```

注 Identity Manager には、実際には前述の例で示す情報以外の情報も保存されます。詳細については、次の項で説明します。

ワークフロー監査の処理では、事前に定義されている属性の名前とその値が、監査対象イベントごとに記録されます。ワークフロー内の監査を有効にするには、WFProcess 要素、あるいは 1 つまたは複数の Activity 要素に、audit 属性を値 true で追加します。WFProcess レベルで属性を設定すると、ワークフロー全体が監査対象となり、個々の Activity 要素に属性を追加した場合は、特定のアクティビティーのみが監査対象となります。監査属性を設定しない場合、監査は無効になります。また、ワークフローを呼び出すタスクテンプレート内でも監査を有効にする必要があります。

記録される情報と記録される場所

デフォルトでは、ワークフローの監査は、通常の監査イベントで記録されるほとんどの情報が収集されます。これには、次の属性が含まれます。

- WORKFLOW: 実行対象ワークフローの名前
- PROCESS: 現在の実行対象プロセスの名前
- INSTANCEID: 実行対象ワークフローの一意のインスタンス ID
- ACTIVITY: イベントがログに記録されるアクティビティー
- MATCH: ワークフローインスタンス内の一意の識別子
- ORGANIZATION: ユーザーの組織の名前

これらの属性は、logattr テーブルに格納され、auditableAttributesList から取得されます。

Identity Manager は、workflowAuditAttrConds 属性が定義されているかどうかについても調べます。

プロセスまたはワークフローの 1 つのインスタンスの中で、特定のアクティビティーを何度も呼び出すことができます。特定のアクティビティーインスタンスの監査イベントを一致させるために、Identity Manager は、ワークフローインスタンス内の一意の識別子を logattr テーブルに記録します。

ワークフローの logattr テーブルに追加属性を記録するには、workflowAuditAttrConds リスト (GenericObjects のリストと見なされる) を定義する必要があります。workflowAuditAttrConds リストに attrName 属性を定義すると、Identity Manager は、コード内のオブジェクトから attrName を抽出します。まず、attrName をキーとして使用し、次に、attrName の値を記録します。すべてのキーと値は、大文字の値として記録されます。

アプリケーションの追加

Identity Manager IDE からアクセスできるように、独自に用意した Java メソッドを登録できます。次の手順で実行します。

1. idm/config/workflowregistry.xml ファイルを編集します。
2. 次の例のような形式で、アプリケーションの定義を追加します。

```
<WorkflowApplication name='Increment Counter'
  class='com.waveset.util.RandomGen' op='nextInt'>

   <ArgumentDefinition name='start' value='10'>
      <Comments>Get the next counter</Comments>
   </ArgumentDefinition>
</WorkflowApplication>
```

3. Identity Manager IDE を再起動します。

アプリケーションメニューに新しいアプリケーションが追加されます。

第 2 章

Identity Manager フォーム

この章では、Sun™ Identity Manager の管理者向けおよびユーザー向けインタフェースの各種ページを定義するフォームを編集することで、選択したページの外観や動作をカスタマイズする方法について説明します。

Identity Manager IDE を使用して、Identity Manager のフォームおよびその他の汎用オブジェクトを表示および編集できます。Identity Manager IDE をインストールおよび設定する手順については、https://identitymanageride.dev.java.net を参照してください。

この章の内容

この章は、次の各節で構成されています。

- フォームについて - フォームの基本概念を紹介し、Identity Manager にフォームを統合する方法について説明します。
- フォームのカスタマイズ - フォームを操作するときに使用する、プログラミングの構文とロジックに関するガイドラインについて説明し、各種フォーム要素の例を示します。
- カスタマイズしたフォームの検証 - フォームの構文を検証したり、カスタムフォームのフィールドロジックを追跡したりする方法について説明します。

関連する章

- 「Identity Manager のビュー」 - ユーザービューと呼ばれる Identity Manager の内部データ構造と連携する Identity Manager フォームについて説明します。フォームをカスタマイズするときは、ビューの属性を呼び出します。
- 「HTML 表示コンポーネント」 - フォームを編集するときは、フィールド定義の作成に HTML コンポーネント言語を使用します。

- 「XPRESS 言語」- 式を使用して、フォームにロジックを設定します。

フォームについて

Identity Manager の Web ベースのユーザーインタフェースの外観と機能をカスタマイズするには、編集する Web ページに関連するフォームを修正します。

「フォーム」という用語は、ユーザーが情報を入力する Web ページと、ビューへのデータの表示方法に関する規則を含むオブジェクトの両方を意味します。このガイドでは、「フォーム」という用語を、主に、ビューへのデータの表示方法に関する規則を含むオブジェクトという意味で使用します。

この節で説明する内容は次のとおりです。

- フォームとは
- フォームの編集が必要な理由
- フォームが使用される Identity Manager ページ
- 編集されたフォーム
- フォームが機能するしくみ

フォームとは

フォームは、ページに関連付けられたオブジェクトであり、そのページでユーザービュー属性をブラウザ上にどのように表示するかについての規則が含まれています。フォームにはビジネスロジックを組み込むことができ、通常はユーザーに表示する前に、表示データを処理するためにフォームが使用されます。

たとえば、新しいユーザーアカウントを作成するときは、新規ユーザーに関する情報を入力するための「ユーザーの作成」ページを使用します。このページは、Tabbed User Form という Identity Manager リポジトリに含まれるオブジェクト (フォーム) から生成されます。このフォームは、「ユーザーの作成」ページにどのフィールドを表示するか、および各フィールドの表示にどの HTML フォーム要素 (たとえば、テキストボックス、チェックボックス、選択ボックスなど) を使用するかを指定します。また、このフォームは、フィールドの無効化、空のフィールドへのデフォルト値の挿入、別フィールドの値からのフィールド値の計算を行う追加ロジックも指定します。

フォームが制御する内容

フォームによって制御されるオブジェクトやアクティビティーは次のとおりです。

- **ページのレイアウトや表示特性**

フォームはフィールドから構成されます。表示できるフィールドのタイプには、単純なテキストボックス、ラジオボタン、複数の値を持つ選択ボックスなどがあります。フィールドには、別のフィールドから算出される値を割り当てることができ、さらに、それを読み取り専用にしたり、非表示にしたりすることができます。

- **ページで使用されるデータ**

  データは、リソースから動的に取得したり、別のフィールドから算出したりすることができます。Identity Manager の式言語である XPRESS を使用することで、フィールドデータを計算、結合、および論理的に評価することができます。

- **システムに渡されるデータ**

  フォームは、Web ページ用のインタフェースにもなりますが、ActiveSync リソースのように対話的でないシステムのインタフェースとなることもできます。この場合、フォームは表示フィールドを持ちませんが、デフォルト値やその他のフィールド値を設定する規則を提供できます。

  たとえば、「Full Name」フィールドは、ページを使用する管理者には表示されないかもしれませんが、ユーザーが「First Name」、「Middle Name」、「Last Name」の各フィールドに入力した値に基づいて設定させることができます。別のフィールドからフィールドに値を取り込むことで、ユーザーや管理者が行うデータ入力が減り、そのためデータ入力エラーが生じる可能性も低くなります。同様に、テキスト入力フィールドにオプションメニューを用意することで、管理者は部署名を入力する代わりに、リストから選択することができます。デフォルトの Identity Manager フォームを定義する各種 HTML コンポーネントについては、「HTML 表示コンポーネント」を参照してください。

- **Identity Manager のバックグラウンド処理**

  Identity Manager では、バックグラウンド処理にもフォームが使用されます。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。

  ユーザーに表示されないフォームでは、外観は意味を持たないため、バックグラウンドでデータを操作するフォームを作成するときは、主にエンコーディングのロジックに注意します。非表示のコンポーネントの詳細については、「非表示コンポーネントの使用」の節を参照してください。

フォームの例

次の XML コード例は、ユーザーがアカウント ID、姓、名、およびフルネームを入力するときに使用するフォームフィールドを定義しています。この例は、「First Name」フィールドと「Last Name」フィールドに入力された情報から、ユーザーのフルネームがどのように構築されるのかを示しています。

コード例 2-1

```
<Field name='waveset.accountId'/>
   <Display class='text'>
     <Property name='title' value='AccountID'/>
   </Display>
</Field>
<Field name='global.firstname'>
   <Display class='Text'>
      <Property name='title' value='First Name'/>
      <Property name='size' value='32'/>
      <Property name='maxLength' value='128'/>
   </Display>
</Field>
<Field name='global.lastname'>
   <Display class='Text'>
      <Property name='title' value='Last Name'/>
      <Property name='noNewRow' value='true'/>
      <Property name='size' value='32'/>
      <Property name='maxLength' value='128'/>
   </Display>
</Field>
<Field name='global.fullname'>
   <Display class='Text'>
      <Property name='title' value='FullName'/>
      <Property name='size' value='32'/>
      <Property name='maxLength' value='32'/>
   </Display>
      <Expansion>
         <concat>
            <ref>global.firstname</ref>
            <s> </s>
            <ref>global.lastname</ref>
         </concat>
      </Expansion>
</Field>
```

フォームの編集が必要な理由

デフォルトの Identity Manager ページには、製品でのアクションの実行に必要なすべてのフィールドがすでに用意されていますが、なぜこれをカスタマイズするのでしょうか。これは、デフォルトのフォームをカスタマイズすることで、企業のポリシーとプロセスをより確実に適用できるからです。

- **画面に表示されるユーザーのアカウント情報を制限することで、プライバシーを保護します。**プライバシーに対する懸念や、不要な情報が招く混乱を回避するために、ユーザーに応じて、特定の情報をユーザーアカウントに表示しないようにすることが必要な場合があります。
- **各フィールドの使い方に適したヘルプを表示します。**これにより、混乱を少なくし、ヘルプデスクへの問い合わせも減らすことができます。
- **特定のタスクを実行するユーザーに不要な情報による混乱を減らします。**通常、情報をもっとも効果的に表示する方法は、現在のタスクの実行に必要なフィールドのみを表示することです。

Identity Manager フォームのデフォルトフィールドをカスタマイズすることで、実際の環境に合わせてアプリケーションを拡張し、カスタマイズすることができます。具体的には、デフォルトフォームを次のようにカスタマイズできます。

- **組織内のユーザーの固有のニーズに対応します。**これは、複数の異なる種類の管理者が、同じビューデータの異なる部分にアクセスする必要があり、記録されているすべてのデータ属性を表示できない場合に特に重要です。たとえば、人事担当の管理者は、ヘルプデスクの管理者とは異なるサブセットのユーザーアカウント属性にアクセスする必要があります。
- **ユーザーアカウント属性の表示と内容、特に「ユーザーの作成」ページと「ユーザーの編集」ページに表示される属性を制御します。**これらのページには、制御が必要なほとんどの属性が含まれます。
- **ユーザービュー属性のデフォルト値、および関連するその他の属性を定義します。**たとえば、管理者がホームディレクトリのパスを入力する代わりに、ユーザーのデフォルトのホームディレクトリを定義できます。
- **画面に表示する前にユーザービュー属性を処理できます。**たとえば、頭字語や数値 ID としてリソースに記録されている部署コードを、ユーザーにわかりやすいフルネーム表記で表示することができます。
- **入力後にユーザービュー属性データを処理できます。**たとえば、場所のフィールドの値に基づいて、メールアカウントを自動的に作成できます。
- **複数のフィールドを 1 つの行に配置することで、画面の構成を制御できます。**Identity Manager フォームのフィールドの配置をカスタマイズすることで、印刷時の形式や、以前からの Web フォームにより近づけることができます。
- 非表示の属性を計算するための**規則を定義します。**たとえば、ユーザーの名前、ピリオド、ユーザーの姓、次にメールドメイン (`joe.user@sun.com`) となるように、ユーザーの電子メールアドレスを計算できます。

シナリオの例

さまざまなニーズや目的を持つ人々が同じデータにアクセスする環境では、フォームは特に便利です。

たとえば、企業で採用を担当するマネージャー用のフォームを作成し、新規従業員アカウントの作成に使用できます。デフォルトの Tabbed User Form には、採用責任者に必要な情報より多くの情報が表示されます。99 個のフィールドがすべて表示される冗長で込み入ったフォームでは、ユーザータスクが複雑になる可能性があります。その代わりに、採用責任者は 10 個の属性フィールドだけを記入し、それ以外の 89 個の属性は管理者が定義した規則に従って設定されるようなフォームを作成できます。

フォームが使用される Identity Manager ページ

通常は、Identity Manager フォームは次のいずれかのカテゴリに分類されます。

- **グラフィカルユーザーインタフェースを駆動するフォーム**。これらのフォームは、Identity Manager の管理者インタフェースやユーザーインタフェースの一部として使用することができ、次のような場合にユーザーが使用するページを含みます。
  - パスワードを変更する
  - セルフサービスを実行する
  - アカウント作成、システム設定、およびワークフロータスクに関連する管理タスクを実行する

  Identity Manager に付属するデフォルトのフォームは、独自のカスタムフォームを作成するためのたたき台として使用できます。これらのフォームのサブセットのみをコピーし、直接編集する必要がある場合は (「編集されたフォーム」の節を参照)、たとえば、特定の属性や動作をエンコードする方法について、別のフォームを参考にできます。
- **外部リソースから Identity Manager にインポートされる情報のバックグラウンド処理を行うフォーム**。たとえば、PeopleSoft データベースから Identity Manager への情報の読み込み処理の一部として、受信レコードに含まれる、従業員の状態を調べるフォームなどがあります。従業員の状態がアクティブでない場合 (A)、そのユーザーの Identity Manager アカウントを無効にするフィールドがフォームによって定義されます。

次の表は、最初のタイプのフォームを使用する Identity Manager ページの一部を示しています。編集するページの表示特性を制御するフォームを特定するときは、この表を使用してください。

表 2-1 ページ、および関連 JSP とフォーム

編集するページ	関連する JSP	関連するフォーム
ユーザーの作成 / ユーザーの編集	`account/modify.jsp`	タブ付きユーザーフォーム
ユーザーアカウント属性の変更	`user/changeAll.jsp`	エンドユーザーフォーム
ようこそ	`user/anonmmain.jsp`	匿名ユーザーメニュー
作業項目の編集	`approval/itemEdit.jsp`	承認フォーム

編集されたフォーム

Identity Manager に付属するデフォルトフォームの中で、次の 5 つのフォームのいずれかの編集が必要となることが考えられます。

- エンドユーザーメニューフォーム
- Anonymous User Menu Form
- Tabbed User Form
- エンドユーザーフォーム
- 承認フォーム

これらの編集されたフォームは、ユーザーの作成と修正を制御し、ユーザーに表示されるメインメニューの表示を制御します。各フォームについては、次の節でさらに詳しく説明します。

注 ビューとフォームが要求を呼び出すために管理者インターフェース JSP を介して対話する間 (ワークフローを呼び出すまで) は、ビューは直接編集されます。このため、フォームはフォーム属性に指定されているネームスペース内で実行されます。代表的な属性ネームスペースには、次のものがあります。

- `accounts[*].*`
- `waveset.*`
- `accountInfo.*`
- `:display.session` （ 管理者用のセッション ）

承認ページには適用されません。

エンドユーザーメニューフォーム

エンドユーザーメニューフォームは、Identity Manager のユーザーインタフェースのメインメニューの表示を制御します。通常、このフォームには、ユーザーパスワードの変更、アカウント属性の編集、秘密質問の回答の変更などを行うためのリンクが含まれます。

エンドユーザーメニューフォームをカスタマイズして、ユーザーがアクセスできる特別なワークフロープロセス (たとえば、システムへのアクセスを要求するプロセス) を呼び出すリンクを追加できます。

注 End User Interface Change Password Form では、ユーザーが自分のアカウントのパスワードを変更するときに現在のパスワードの再入力を求めるように RequiresChallenge プロパティーを設定できます。このプロパティーの設定方法の例については、enduser.xml で Basic Change Password Form を参照してください。

たとえば、エンドユーザーページでクリックするリンクとしてエンドユーザーテストプロセスを表示するときは、次のコード例のようなエントリを追加します。

コード例 2-2 エンドユーザーメニューフォームへのエンドユーザーテストプロセスリンクの追加

```
<Configuration id='#ID#Configuration:EndUserTasks' name='End User Tasks' >
   <Extension>
     <List>
        <List>
        <String>End-User Test Process</String>
        <String>An example end-user workflow</String>
        <List>
     </List>
```

Identity Manager のユーザーインタフェースには、選択可能なセルフサービスプロセスリストが表示されます。これは、リストのサブリストとなります。サブリストの最初の要素はプロセス名を示し、2 番目の要素はプロセスの機能を示します。

注 ページの表示を更新するたびに、Identity Manager はフォームの <Default> 式を再評価します。エンドユーザーメニューフォームに doNotRegenerateEndUserMenu プロパティー (true に設定) を追加することで、このフォームの再生成を無効にできます。

ページの表示を更新するたびに、Identity Manager はフォームの <Default> 式を再評価します。エンドユーザーメニューフォームに doNotRegenerateEndUserMenu プロパティー (true に設定) を次のように追加することで、このフォームの再生成を無効にできます。

```
<Properties>

   <Property name='doNotRegenerateEndUserMenu'>

      <Boolean>true</Boolean>

   </Property>

</Properties>
```

Anonymous User Menu Form

Anonymous User Menu Form は、未知のユーザーがログインしたときに、Identity Manager のユーザーインタフェースのメインメニューの表示を制御します。

ユーザーセルフプロビジョニングのプロセス経由でシステムに定義されていないユーザーに対しては、Identity Manager は匿名エンドユーザーページを使用します。たとえば、Identity Manager 管理者は、Active Directory リソースのパススルー認証を設定することができます。それにより、Active Directory アカウントを持つ人であれば、Identity Manager のユーザーインタフェースにログインできるようになります。ユーザーが Identity Manager アカウントログインを持たない場合に、Identity Manager のユーザーオブジェクトが作成され、Active Directory リソースが追加されるように、これらのページをカスタマイズできます。それに続いて一連の質問を設けることで、システムはユーザーのロール、組織、およびその他のリソースを設定できます。

Identity Manager ユーザーが作成される前に、サービスを要求するワークフロープロセスを呼び出すよう Anonymous User Menu Form をカスタマイズすることができます。

Tabbed User Form

Tabbed User Form は、Identity Manager の管理者インタフェースでユーザーを作成、修正するときに使用されるデフォルトフォームです。独自にデザインしたフォームとして、このフォームの拡張版を作成するためには、このフォームのコピーをカスタマイズします。

ヒント Tabbed User Form を直接編集しないでください。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ことをお勧めします。こうすることにより、カスタマイズしたコピーがサービスパックの更新やアップグレードの際に上書きされるのを防止できます。

Tabbed User Form のコピーは、次の処理を行うようにカスタマイズできます。

- **「ユーザーの編集」ページに表示される属性の数を制限します**。デフォルトでは、このページにリソースのスキーママップに定義されているすべての属性が表示されるため、採用責任者に入力や設定が求められる属性のリストが多くなり過ぎてしまいます。
- **デフォルトのフィールドタイプを、より便利な選択ボックス、チェックボックス、および複数の値を示すフィールドに設定します**。デフォルトでは、ユーザーに割り当てられるリソースに定義されるすべての属性は、「ユーザーの作成」ページと「ユーザーの編集」ページにテキストボックスとして (または、Boolean 型の場合はチェックボックスとして) 表示されます。
- **複数のページで共通のフォームを使用できるように、追加のフォームを含めます**。

Tabbed User Form には、次のフィールドが含まれます。

- accountId
- firstname
- lastname
- role
- organization
- password
- confirm password
- email
- resource list
- application list
- MissingFields

注 本稼働環境で MissingFields 要素は使用しないでください。この要素は、教育の目的にのみ用意されています。

MissingFields 要素は、Tabbed User Form からユーザーフォームを作成またはカスタマイズするときに、割り当てられているリソースにプッシュできる属性への明示的な参照によって置き換える必要があります。この置き換えは、グローバルネームスペースを使いすぎることでよく発生する問題を回避するために行う必要があります。たとえば、グローバル構文を使用しない場合は、リソースは割り当てられません。

MissingFields フィールドは、実際にはフィールドではありません。Tabbed User Form で明示的に宣言されていない割り当て済みリソースへプッシュ可能なすべての属性に対して、グローバルネームスペースにテキストフィールドを自動生成するようにフォームジェネレータに指示する要素です。

デフォルトでは、ユーザーに割り当てられるリソースに定義されるすべての属性は、「ユーザーの作成」ページと「ユーザーの編集」ページにテキストボックスとして (または、ブール型の値ではチェックボックスとして) 表示されます。

エンドユーザーフォーム

エンドユーザーフォームは、ユーザーが Identity Manager のユーザーインタフェースで /user/main.jsp から**アカウント属性**を選択した場合に、システムによって自動的に表示されるページを制御します。ユーザーはこのページで、自分自身のパスワード、秘密の質問、および電子メールアドレスを変更できます。

エンドユーザーフォームをカスタマイズして、電話番号、住所、事務所の地理的な場所などを扱うフィールドをユーザーが制御できるように権限を与えることができます。

承認フォーム

承認フォームは、ユーザー要求の承認者を指定するときに、リソース、ロール、または組織の所有者に表示される情報を制御します。デフォルトでは、このページには、プロセスを開始した管理者の名前を示す、一連の読み取り専用フィールドが表示されます。また、アカウント ID、ロール、組織、電子メールアドレスなど、ユーザーに関する情報も表示されます。

このフォームは、ユーザーの作成前に、ユーザーの値を変更する最後の機会をリソースの所有者に提供できるようにします。デフォルトでは、ユーザーを承認すると、読み取り専用フィールドにすべてのユーザー属性が表示されます。

承認フォームは、次の処理を行うようにカスタマイズできます。

- ユーザーに関する情報の追加、削除。
- ユーザーの初期フォームに入力された情報を変更するため、この情報を編集する権限の承認者への割り当て。
- 異なる目的に合わせて独自の承認フォームを作成。たとえば、管理者やリソース所有者がアカウント作成またはユーザー削除を開始するときに使用される専用の承認フォームを作成できます。

フォームが機能するしくみ

ブラウザによるフォームの表示には、さまざまな要因が影響します。ただし、ブラウザでのフォームの動作は、主に次の要素によって決定されます。

- **フォームに関連付けられているビュー**。すべてのフォームは、ビューとともに使用されます。フォームとともに使用されるもっとも一般的なビューは、ユーザービューです。ビューは、フォームが評価されるときに使用できるデータを定義します。
- **未定義の属性**。Tabbed User Form は、フィールドが明示的に定義されていないリソースアカウント属性を編集するためのテキストフィールドを自動的に生成するメカニズムを提供します。この機能は、フォームで無効にすることができます。
- **フォームとその他の Identity Manager コンポーネントとの連携**。これには、Identity Manager がフォームまたは フォーム評価 を評価するためのプロセスが含まれます。フォームによって駆動されるすべてのページは、同様に処理されます。Identity Manager がフォームを評価する方法については、この章の「フォームの評価」を参照してください。
- **フォームで使用される表示コンポーネント**。フォームフィールドには、ブラウザにフィールドをどのように表示するかを決定する、表示コンポーネントを関連付けることができます。

ユーザービューとフォーム

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、次の項目が含まれます。

- Identity Manager の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、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ます。ビュー属性とは、ビュー内にある名前付きの値です。たとえば、`waveset.accountId` は、Identity Manager のアカウント名を値とするユーザービュー中の属性です。

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。詳細については、「フィールド名の定義」を参照してください。

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」を参照してください。

未定義の属性

管理者インタフェースでユーザーにリソースやロールを割り当てると、表示が更新されます。新しいリソースアカウントの属性は、ユーザービューで定義されます。Tabbed User Form の `<FormRef name = 'Missing Fields'/>` は、対応するフィールドがフォームに明示的に定義されていないリソースアカウント属性について、テキストフィールドを生成するようにフォーム生成ツールに伝えます。この機能を無効にするときは、Tabbed User Form から `<FormRef name = 'Missing Fields'/>` を削除します。

フォームの評価

システムがフォームをどのように処理するかを理解することは、ブラウザでのフォームの動作を決定するときに役立ちます。フォームによって駆動されるすべてのページは、次のように同様に処理されます。

1. **ページは、**Identity Manager のユーザーインタフェースまたは管理者インタフェースから要求されます。
2. **インタフェースは、サーバーからビューを要求します。**ビューは、編集が可能な、名前が付けられた値の集合です。それぞれのビューは、ビューの値をユーザーにどのように表示するかを定義するフォームと関連付けられます。
3. サーバーは、リポジトリ内の 1 つまたは複数のオブジェクトからデータを読み込むことで、**ビューを構築します。**ユーザービューの場合は、リソースアダプタを通じてリソースからアカウント属性も取得されます。

4. **Derivation 式が評価されます**。これらの式は、リソースからの暗号化され、エンコードされた値を、ユーザーにとって意味のある値に変換するときに使用されます。この Derivation 式は、フォームが最初に読み込まれるとき、あるいは、1 つまたは複数のリソースからデータがフェッチされるときに評価されます。

5. **Default 式が評価されます**。フィールドが NULL の場合は、フィールドにデフォルト値が設定されます。

6. **HTML コードが生成されます**。システムはビューデータとフォームを処理し、HTML ページを生成します。この処理では、式に含まれる `allowedValues` プロパティーが評価され、`Select` または `MultiSelect` HTML コンポーネントが構築されます。

7. ブラウザにページが表示されます。ユーザーは、表示される値を編集できます。編集時にユーザーがフィールドを修正すると、多くの場合は、それによってページの表示が更新されたり、再計算されたりします。これにより、**ページが再生成されます**。ただしこの時点では、システムは編集後のデータをリポジトリにまだ格納しません。

8. **変更した値が、ビューに同化されます**。表示更新イベントが発生した場合は、インタフェースはブラウザで編集されたすべてのフォームフィールドの値を受信します。

9. **Expansion 式が評価されます**。これにより、追加の値がビューに配置される場合があります。Expansion 規則は、ページを再計算するとき、またはフォームを保存するときに実行されます。

10. **ビューの表示が更新されます**。インタフェースは、ビューの表示を更新し、編集された値の現在の設定を提供するようにサーバーに要求します。サーバーは、リポジトリまたはリソースからデータを読み込んで、より多くの値をビューに挿入する場合があります。

11. **Derivation 式が評価されます**。通常は、ビューの表示を更新するときには、Derivation 式は評価されません。一部の複雑な事例では、表示の更新後にシステムが導出を要求する場合があります。

12. システムは、表示が更新されたビューとフォームを処理し、**ブラウザに返される別の HTML ページを構築します**。ユーザーは、表示更新の結果を確認し、編集を継続します。ユーザーは、変更を保存またはキャンセルするまで、任意の回数だけビューの表示を更新できます (毎回、手順 7 〜 12 を繰り返す)。

13. A. 編集をキャンセルすると、ビューに取り込まれたすべてのデータは破棄され、サーバーにそれが通知されます。これにより、サーバーは任意のリポジトリロックを解除し、別のページへのパスを制御できます。

14. B. 編集を保存すると、インタフェースは変更された値を受信し、それをビューに同化させます (手順 8 を参照)。

15. Validation 式が評価されます。フィールドの値が必要な仕様を満たしていない場合は、エラーが示され、フィールドの値を修正できます。変更を加えると、プロセスは手順 13 に戻ります。
16. 最後にもう一度、Expansion 式が評価されます (手順 9 を参照)。
17. サーバーがビューを保存すると、通常は、**リポジトリ内の 1 つまたは複数のオブジェクトが修正されます**。ユーザービューでは、リソースアカウントも更新される場合があります。

これらの手順の一部は、フォームのすべてのフィールドに対して繰り返す必要があります。これには、Derivation 式の評価、Default および Validation 式の評価、HTML の生成、および Expansion 式の評価が含まれます。すべてのフィールドに対する繰り返しでは、このフィールドの処理が必要であるかどうかを判断するために、Disable 式が評価されます。Disable 式の評価結果が true であれば、そのフィールド、および入れ子になっているすべてのフィールドは無視されます。これら特別な種類の式の詳細については、この章の「フィールド名の定義」を参照してください。

空フォーム

空フォームとは、フィールドのないフォームです。空フォームを配備することで、フォーム処理中にユーザービューに変更が適用されるのを防止できます。

コード例 2-3 空フォームオブジェクト

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE Configuration PUBLIC 'waveset.dtd' 'waveset.dtd'>
<Configuration wstype='UserForm'name='Empty Form'>
 <Extension>
    <Form name='Empty Form'>
    </Form>
</Extension>
   <MemberObjectGroups>
      <ObjectRef type='ObjectGroup' name='Top'/>
   </MemberObjectGroups>
</Configuration>
```

注 空フォームは、MissingFields の代わりとして実装できます。MissingFields は、ユーザービューのアカウントリスト全体に対して、自身の機能を実行しようとします。一方、空フォームを使用することで、フォーム処理中にユーザービューに変更が適用されるのを効果的に防止できます。

空フォームを使用する状況

空フォームは次のようなときに使用します

- 二次的な影響を最小限に抑えてビューをチェックアウトするとき。
- Active Sync リソースから、および「ファイルから読み込み」時にデータを同期するとき。デプロイヤは通常、データを同期するためにカスタマイズした入力フォームを作成します。Identity Manager はすべての属性変換をその入力フォーム内で行うため、追加のフォーム処理は必要ありません。ただし、この処理には管理コンテキストが必要であり、すべての管理者がフォームを使用する必要があるため、管理者のフォームを空フォームに割り当てて、対象のビューで不必要なフォーム処理が行われないようにするべきです。
- ユーザービューに対して特定のアクションを実行するが、予期しないその他の副次的な影響を回避する必要があるカスタムワークフローを実装するとき。このようなカスタムワークフローでは、一般にビューのチェックアウト処理時に空フォームを指定します。

Active Sync 処理時に空フォームを使用

Active Sync を正常に処理するには、空フォームを使用する必要があります。

Identity Manager が Active Sync リソースのレコードを処理するときは、Identity Manager システム用に設定されたデフォルトフォーム (Tabbed User Form)、または管理者アカウント用に定義されたフォームがある場合はそれを使用します。

デフォルトの Tabbed User Form では Active Sync リソースのデータに予期しない効果が発生する可能性があるため、デフォルトフォームを使用するのではなく、空フォームを作成してプロキシ管理者に直接割り当てることをお勧めします。

注 空フォームをプロキシ管理者に関連付ける場合は、管理者ロールを使って関連付けるのではなく、直接関連付けてください。管理者ロールレベルで空フォームを設定すると、Active Sync 処理時に Tabbed User Form が呼び出されてしまいます。

Active Sync フォーム

Active Sync 対応アダプタでは、Active Sync 処理時にリソースフォームとユーザーフォームの 2 種類のフォームが一般的に使用されます。

フォームの処理は、次の 3 段階で行われます。

1. Active Sync フィールドに属性とリソースの情報が設定されます。リソースの属性を取得して設定するときは、`activeSync` ネームスペースを使用してください。

2. リソースフォームが展開されて値が取得されます。この展開を行うときには、あらゆるユーザービュー属性を取得できます。
3. ユーザーフォームが展開されて値が取得されます。

`$WSHOME/sample/forms` ディレクトリには、`ActiveSyncForm.xml` で終わるサンプルフォームが用意されています。これらのサンプルフォームには、新規および既存のユーザーのケースを処理するロジック、およびリソース上で削除が検出されたときに Identity Manager ユーザーを無効または削除するロジックが含まれます。

ActiveSync ユーザーフォーム

Identity Manager が ActiveSync ユーザーイベントを処理するときには、次の 2 つのフォームが使用されます。

- Proxy Admin Form。Identity Manager は、ユーザービューを作成およびチェックインするときにこのフォームを処理します。
- ActiveSync Form。Identity Manager は、ユーザービューを最初に作成したあと、および Proxy Admin Form を処理したあとにこのフォームを処理します。適切なフォームロジックを ActiveSync イベントに適用し、デフォルトの Tabbed User Form によって問題が発生することを回避するために、空フォームを Proxy Admin に割り当てることをお勧めします。

リソースフォーム

リソースフォームは、リソースが作成または編集されるとき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。選択したフォームへの参照が、リソースオブジェクトに格納されます。

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

- スキーママップから受け取った属性を変換する。
- パスワード、ロール、組織などのフィールドを生成する。
- 新規および既存のユーザーのケースを処理するロジックや、削除が検出されたときに Identity Manager ユーザーを無効または削除するロジックなど、カスタム処理用の簡単な制御ロジックを提供する。
- `activeSync` から取り込んだ属性をユーザーフォームが入力として使用するフィールドへコピーし、必要に応じて変換する。作成処理に必要なフィールドは、`waveset.accountId` および `waveset.password` です。`accounts[AD].email`、`waveset.resources` など、その他のフィールドも設定できます。
- `IAPI.cancel` を true に設定することで、ユーザーの処理をキャンセルする。特定のユーザーに対する更新を無視するときによく使用されます。

リソースフォームには、新規および既存のユーザーのケースを処理するロジック、および削除が検出されたときに Identity Manager ユーザーを無効または削除するロジックが含まれます。

ユーザーフォーム

ユーザーフォームは、Identity Manager インタフェースから編集するために使用されます。割り当てるときは、アダプタに「プロキシ管理者」を割り当てます。プロキシ管理者にユーザーフォームが割り当てられている場合は、このフォームは処理時に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

アダプタのプロキシ管理者は、`ProxyAdministrator` 属性を使用して設定します。任意の Identity Manager 管理者に設定できます。Active Sync 対応アダプタのすべての処理は、プロキシ管理者が実行している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は、デフォルトのユーザーフォームが指定されます。

属性を処理するための代替フォーム

姓と名からフルネームを作成するなどの一般的な変更は、ユーザーフォームで行うことをお勧めします。リソースフォームでは、HR 状態が変わったときはそのユーザーを無効にするなど、リソース固有の変更を行うようにしてください。ただし、受け取った属性などの必要な属性が共通パスに置かれたあとに、リソースフォームを別のフォームに取り込むことはできます。

ActiveSync フォームの処理

ActiveSync フォームの処理は、大きく分けて次の段階で構成されます。

- Proxy Admin Form を使用してビューを作成します。`MissingFields` FieldRef によって Proxy Admin Form からグローバル属性が取り込まれることを回避するために、空フォームを管理者に割り当ててください。
- ActiveSync Form が処理される前に、Identity Manager は特定の ActiveSync ビュー属性をビューに追加します。
- Identity Manager は、入力フォームまたはパラメータ化した ActiveSync Form を処理します。このフォーム処理手順中に、通常はリソース、ロール、および組織と、関連付けられたリソースアカウント属性が指定されます。イベントのタイプに基づいてフィールドを無効または有効にするために、`feedOp` フラグが使用されます。特定のイベントを無視するために、`IAPI.cancel` 属性が設定されることもあります。`viewOptions.process` フラグを使用して、ActiveSync イベントを処理するためのカスタムプロビジョニングタスクを指定することもできます。
- Identity Manager は、ビューをチェックインして Proxy Admin Form を処理します。

エンドユーザーフォーム

このセクションで説明するフォームは、エンドユーザーインタフェースにあります。

エンドユーザー委任フォーム

エンドユーザーインタフェースでは、「委任」タブから次のフォームにアクセスできます。

- End User View WorkItem Delegations Form
- End User Delegate WorkItems Form

End User View WorkItem Delegations Form

このフォームは、現在および過去のすべての委任が表示される 1 つのテーブルで構成されます。このテーブルは、委任状態に基づいてフィルタリングできます。有効な状態値は、「現在」、「将来の委任」、「終了済み」です。ユーザーは現在の委任を終了したり、新しい委任を作成したり、既存の委任を編集したりできます。

このフォームに表示される過去の委任のリストを設定するために、SystemConfiguration オブジェクトの delegation.historyLength 属性を設定できます。詳細については、『Identity Manager の配備に関する技術概要』の設定オブジェクトの編集に関する章を参照してください。このフォームのその他の動作を設定するときは、このガイドの「Identity Manager のビュー」章の「作業項目の委任ビュー」を参照してください。

End User Delegate WorkItems Form

このフォームは、新しい委任を作成したり、既存の委任を編集したりするときに使用されます。

エンドユーザー匿名登録フォーム

Identity Manager アカウントを持たないユーザーは、アカウントを要求するときにこれらのフォームを使用します。

- End User Anonymous Enrollment Form。インタフェースのこの部分のメインフォームで、エンドユーザー匿名登録ワークフローを実行するときの呼び出しフォームとして参照されます。このフォームは次の 2 つのフォームを参照します。
- End User Anonymous Enrollment Validation Form。このフォームは、ユーザーから初期情報を収集して、完了フォームを表示する前にユーザーの関係 (雇用) を検証するためのサンプルです。
- End User Anonymous Enrollment Completion Form。このフォームは、ユーザーアカウントを作成するプロビジョニングタスクの呼び出しに必要な情報を収集します。

フォームのカスタマイズ

Identity Manager 製品のデフォルトの動作を理解すると、カスタマイズが必要なページを特定できます。

1. 編集可能なページと、それに対応するフォームについては、「編集されたフォーム」の節を参照してください。
2. フォームを編集するときは、Identity Manager IDE を起動し、「リポジトリオブジェクトを開く」を選択します。表示されるポップアップダイアログから、編集するフォームを選択します。

この節で説明する内容は次のとおりです。

- カスタマイズの概要
- カスタマイズに関連するその他のトピック
  - フォームの構造
  - フォームフィールドとは
  - フィールドの表示を定義するフォームの作成に関するガイドライン
  - フォームフィールド内の式の最適化
  - 新しいリソースとユーザーの自動リンク設定の無効化
  - 結果ページでのクリアテキストによる属性の表示の回避
  - フォームへのガイダンスヘルプの追加

カスタマイズの概要

フォームをカスタマイズして、よりわかりやすくし、表示特性を変更したり、フィールドデータを処理するロジックを含めたりすることができます。

基本的な手順

Identity Manager システムでフォームをカスタマイズする基本的な手順は次のとおりです。

- **カスタマイズするフォームを選択します。**カスタマイズするフォームを特定する方法を指定します。
- **フォームを編集し、保存します。**製品に付属する、エンドユーザーと管理者のデフォルトのフォームの変更について、基本的な情報を示します。
- **変更をテストします。**運用環境に読み込む前に変更をテストし、エラーのログ記録を有効にするためのガイドラインを示します。

一般的なタスク

フォームを編集するときは、通常、次のタスクを実行します。

- **フォームのフィールドを追加、削除します。**一般的なタスクには、一部のデフォルトフィールドの削除と、環境に合わせてカスタマイズされたフィールドの追加が含まれます。
- **フィールドをフォームにどのように表示するかを定義します。**この操作には、Identity Manager に付属する HTML コンポーネントのライブラリを使用する必要があります。フィールドの表示特性の編集については、「フィールドの表示プロパティー」の節を参照してください。
- **フィールドの値を定義する論理式を設定します。**この操作では、XPRESS 言語を使用して論理式を作成する必要があります。XPRESS の使用については、「XPRESS 言語」を参照してください。

表 2-2　フォームの要素

プロパティー	説明
Title	フォームフィールドの横に表示されるテキストを指定します。
Class	要素が属す HTML 表示クラスを指定します。
Required	フォームの処理にその要素が必須であるかどうかを指定します。送信の時点では、このフィールドに NULL 以外の値が入力されている必要があります。必須に設定すると、フィールドの右に赤いアスタリスクが表示されます。フォームの下部にはメッセージテキストが表示され、フィールドの横の赤いアスタリスクが、送信時にフィールドへの値の入力が必須であることを示していることを通知します。
Action	これを設定すると、変更によって、すべての `Select` コントロールまたは `MultiSelect` コントロールの表示が更新されます。Identity Manager の管理者インタフェースでは、この設定によって、基本となるビューの表示が更新されます。ロールの選択は、この動作のよい例です。Tabbed User Form で新しいロールを選択すると、編集セッション中にそのロールを通じて割り当てられるリソースを反映して、ビューの表示が更新されます。ビューの表示が更新されると、新たに割り当てられたリソースのリソースアカウント属性を明示的に設定できるようになります。
No New Row	フォームのレイアウトに限定して使用されます。true に設定すると、そのフィールドが直前のフィールドの右に強制的に表示されます。たとえば、Name フィールドでは、この設定が便利です。ユーザーは、姓、名、およびミドルネームのイニシャルを上から下へではなく、右から左に入力できるようになります。

表 2-2　フォームの要素 (続き)

プロパティー	説明
Hidden	そのフィールドをユーザーに表示しないことを指定します。このようなフィールドは、姓と名を連結してフルネームを構築する場合のように、通常は別のフィールドから計算される属性値の設定に使用されます。
Title	コントロール (テキストボックス) の文字の幅を指定します。
Class	コントロールバッファー (テキストボックス) の文字の幅を指定します。ユーザーが、`size` プロパティーに指定した値より長い文字列を入力した場合は、文字がスクロールされます。
Required	このフォームフィールドの名前を指定します。通常は、このフォームで使用されるビューのパス式が指定されます。

次の特性は、メインタブビューから設定します。

表 2-3　メインタブビューから設定する特性

フィールド	説明
Name	このフィールドの名前を入力します。多くの場合、フィールド名は、このフォームで使用されるビューのパス式です。テキストボックス、チェックボックス、選択フィールドなどの編集コンポーネントとして表示されるすべてのフィールドには、ビューのパスを示す名前が必要です。`SectionHead` や `Javascript` のように編集コンポーネントとして表示されないフィールドは、名前を必要としません。ただし、フィールド参照を通じて別のフォームから参照する必要がある場合は、編集コンポーネント以外のフィールドにも名前を付けることができます。
Title	そのフィールドのタイトルを指定します。このタイトルは、フォーム上のフィールドの横に表示されます。このフィールドの横にあるドロップダウンメニューから、この要素のデータ型を選択してください。このフィールドに表示されるテキストを編集するには、横に表示される「**編集**」ボタンをクリックします。
Sub Title	(省略可能) フォームタイトルの下に表示されるテキストを指定します。このフィールドの横にあるドロップダウンメニューから、この要素のデータ型を選択してください。このフィールドに表示されるテキストを編集するには、横に表示される「**編集**」ボタンをクリックします。
Help Catalog	フィールドにガイダンスヘルプを関連付けるヘルプキーを指定します。この値は、フォームによって指定される、関連するヘルプカタログに含まれるエントリの名前です。ヘルプキーを指定すると、フィールドの左にアイコンが表示されます。このアイコンにカーソルを合わせると、ヘルプカタログから参照されるテキストが表示されます。

表 2-3　メインタブビューから設定する特性 (続き)

フィールド	説明
Base Context	(標準のユーザーフォームでは、通常、使用されない) すべてのフィールドに完全パスを指定する必要をなくすためのベースコンテキストを指定します。ベースコンテキストは、基本となるマップを特定します。具体的には `com.waveset.object.Genericobject` で、通常は `user` または `userview` という名前が付けられます。Identity Manager の管理者インタフェースでは、編集コンテキストはユーザーであり、ベースコンテキストの参照は空白のまま残されます。承認のように、手動アクションによって呼び出されるフォームでは、フォームのコンテキストはワークフローコンテキストとなります。
Options	フィールドの、1 つまたは複数の表示オプションを選択します。 **Required** - フォームの処理にその要素が必須であるかどうかを指定します。送信の時点では、このフィールドに NULL 以外の値が入力されている必要があります。必須に設定すると、フィールドの右に赤いアスタリスクが表示されます。フォームの下部にはメッセージテキストが表示され、フィールドの横の赤いアスタリスクが、送信時にフィールドへの値の入力が必須であることを示していることを通知します。 **Button** - フォーム下部の 1 つの水平行にフィールドを表示します。このオプションを選択しない場合、フィールドはフォームの次の行に表示されます。ほとんどの場合、このオプションは Button 表示クラスを使用するフィールドに設定されます。 **Action** - これを設定すると、変更によって、すべての `Select` コントロールまたは `MultiSelect` コントロールの表示が更新されます。Identity Manager の管理者インタフェースでは、この設定によって、基本となるビューの表示が更新されます。ロールの選択は、この動作のよい例です。Tabbed User Form で新しいロールを選択すると、編集セッション中にそのロールを通じて割り当てられるリソースを反映して、ビューの表示が更新されます。ビューの表示が更新されると、新たに割り当てられたリソースのリソースアカウント属性を明示的に設定できるようになります。 **Library** - 宣言したときではなく、参照した場合にのみフィールドを表示することを指定します。このオプションは、フォームで評価されるフィールドの順序が、ユーザーに表示されるフィールドの順序と異なる場合に便利です。
Default	フィールドのデフォルト値を計算するための式を指定します。このフィールドの現在の値が NULL の場合は、フォームの表示前に Default 式が呼び出されます。
Derivation	表示前にフィールドの値を計算するための式を指定します。これは Default 式に似ていますが、現在のフィールド値が NULL 以外でも、この式は評価されます。Derivation 式は、フォームが最初に表示される前に評価され、その後、フォームの表示が更新されるたびに評価されます。

表 2-3　メインタブビューから設定する特性 (続き)

フィールド	説明
Validation	フォームに入力した値が有効であるかどうかを判断するためのロジックを指定します。Validation 式は、成功を示す場合は NULL を返し、失敗を示す場合は判読可能なエラーメッセージを含む文字列を返します。
Expansion	フォームの送信後にフィールドの値を計算するための式を指定します。多くの場合、Expansion 式は非表示のマークが付けられたフィールドで使用されます。非表示フィールドは、ユーザーが直接編集することができないため、値は Expansion 式を使用して計算できます。「フィールドの非表示」を参照してください。
Disable	true と評価された場合に、フィールドと、その入れ子フィールドを無効にする式を指定します。無効化されたフィールドは、フォームに表示されません。このオプションは、ユーザーが特定タイプのリソースを持っているかどうかを判断するときに使用されます。ユーザーがこのようなリソースを持っている場合は、フォームにはそのリソースに適したフィールドが表示されます。
Display Class	このフォームコンポーネントをブラウザに表示するときに使用される HTML コンポーネントクラスを指定します。デフォルトでは、EditForm 表示クラスが選択されます。フォームがエンドユーザーメニューのようなリンクフォームの場合は、Display Class オプションから LinkForm を選択します。 「HTML 表示コンポーネント」の HTML 表示クラスの表を参照してください。
size	コントロール (テキストボックス) の文字の幅を指定します。
maxLength	この要素の最大文字数を指定します。

ヒント　多くの場合、フィールド名はこのフォームで使用されるビューのパス式で、通常はリソースの特定の属性と関連付けられます。リソースとその属性のリストを表示するには、「**リソースの表示**」をクリックします。展開可能なリソースタイプのツリーを示す「リソースの表示」ダイアログが表示されます。リソースインスタンスと、その属性名のリストを表示するときは、リソースタイプの名前をクリックします。新しいフォームフィールドの名前として、リソースの属性名を使用するときは、リソースの属性名をクリックし、「**OK**」をクリックします。これにより、Name フィールドに属性名が挿入されます。

表 2-4　表示クラスのオプション

HTML コンポーネント	目的
Apple	ページにアプレット参照を挿入します。
BackLink	前のページに戻るリンクを表示します。
BorderedPanel	コンポーネントを、東西南北と中央の 5 つの領域に配置するコンテナ。
Button	ボタンを表示します。
ButtonRow	それぞれの間に隙間を設けて、コンポーネントを水平の行に配置するコンテナ。通常は、Button コンポーネントの行の表示に使用されます。
CheckBox	それぞれの間に隙間を設けて、コンポーネントを水平の行に配置します。通常は、Button コンポーネントの行の表示に使用されます (Container)。
DatePicker	ページにカレンダのアイコンを表示します。ユーザーは、このアイコンをクリックして日付を選択し、ページフィールドに値を適用できます。
EditForm	フォームのデフォルトコンテナ。1 つの列にコンポーネントのタイトルを表示し、もう 1 つの列にコンポーネントを表示します。各行の背景は、交互に灰色と白になります。
FileUpload	アップロードするファイルの名前を指定するための、Text コンポーネントの形式。
Hidden	表示されないデータを HTML ページに含めるときに使用されるコンポーネント。
Html	事前にフォーマットされている HTML をページに挿入します。
Javascript	JavaScript 関数を定義します。
Label	読み取り専用のテキストを表示します。
Link	ページにリンクを配置します。
LinkForm	縦の箇条書きリストに、タイトルなしでコンポーネントを配置します。通常は、Link コンポーネントのリストを含むページで使用されます。EditForm コンテナの代わりに使用されます (Container)。

表 2-4　表示クラスのオプション (続き)

HTML コンポーネント	目的
MultiSelect	複数の選択項目を持つボックスを表示します。これは、1 つのボックスに含まれる定義済みの値セットを選択ボックスに移動できる、2 つの部分から構成されるオブジェクトです。
NameValueTable	ベージュ色の背景を持つ簡易テーブルに、名前と値のペアのリストを表示します。
Panel	コンポーネントを水平の行、または垂直の列に配置します (Container)。
Radio	1 つまたは複数のラジオボタンの水平リストを表示します。ユーザーが一度に選択できるラジオボタンは 1 つだけです。コンポーネントの値が NULL の場合、または許可される値のいずれとも一致しない場合は、ボタンは選択されません。
SectionHead	セクションの見出しを表示します。これは EditForm コンテナで認識され、タイトルとコンポーネントの列を太字で表示します。
Select	1 つの項目を選択できるリストボックスを表示します。
SimpleTable	列のタイトル行を持つ、簡単なグリッドにコンポーネントを配置します。
SubTitle	フォームタイトルの下に表示されるテキストを指定します。
Text	読み取り専用のテキストを表示します。
TextArea	ページにリンクを配置します。
Title	フォーム上部に表示されるテキストを指定します。

表 2-5　フォームの要素

フォーム要素	説明
Name	このフィールドの名前を入力します。多くの場合、フィールド名は、このフォームで使用されるビューのパス式です。テキストボックス、チェックボックス、選択フィールドなどの編集コンポーネントとして表示されるすべてのフィールドには、ビューのパスを示す名前が必要です。SectionHead や Javascript のように編集コンポーネントとして表示されないフィールドは、名前を必要としません。ただし、フィールド参照を通じて別のフォームから参照する必要がある場合は、編集コンポーネント以外のフィールドにも名前を付けることができます。
Title	そのフィールドのタイトルを指定します。このタイトルは、フォーム上のフィールドの横に表示されます。このフィールドの横にあるドロップダウンメニューから、この要素のデータ型を選択してください。このフィールドに表示されるテキストを編集するには、横に表示される「**編集**」ボタンをクリックします。
Help Key	フィールドにガイダンスヘルプを関連付けるヘルプキーを指定します。この値は、フォームによって指定される、関連するヘルプカタログに含まれるエントリの名前です。ヘルプキーを指定すると、フィールドの左にアイコンが表示されます。このアイコンにカーソルを合わせると、ヘルプカタログから参照されるテキストが表示されます。

表 2-5　フォームの要素 (続き)

フォーム要素	説明
Options	フィールドの、1 つまたは複数の表示オプションを選択します。 **Required** - このフィールドへの入力または選択が、フォームを処理する上で必須であるかどうかを指定します。 **Button** - フォーム下部の 1 つの水平行にフィールドを表示します。このオプションを選択しない場合、フィールドはフォームの次の行に表示されます。ほとんどの場合、このオプションは `Button` 表示クラスを使用するフィールドに設定されます。 **Action** - これを設定すると、変更によって、すべての `Select` コントロールまたは `MultiSelect` コントロールの表示が更新されます。Identity Manager の管理者インタフェースでは、この設定によって、基本となるビューの表示が更新されます。ロールの選択は、この動作のよい例です。Tabbed User Form で新しいロールを選択すると、編集セッション中にそのロールを通じて割り当てられるリソースを反映して、ビューの表示が更新されます。ビューの表示が更新されると、新たに割り当てられたリソースのリソースアカウント属性を明示的に設定できるようになります。 **Library** - 宣言したときではなく、参照した場合にのみフィールドを表示することを指定します。このオプションは、フォームで評価されるフィールドの順序が、ユーザーに表示されるフィールドの順序と異なる場合に便利です。
Default	フィールドのデフォルト値を計算するための式を指定します。このフィールドの現在の値が NULL の場合は、フォームの表示前に Default 式が呼び出されます。
Derivation	表示前にフィールドの値を計算するための式を指定します。これは Default 式に似ていますが、現在のフィールド値が NULL 以外でも、この式は評価されます。 Derivation 式は、フォームが最初に表示される前に評価され、その後、フォームの表示が更新されるたびに評価されます。

表 2-5　フォームの要素 (続き)

フォーム要素	説明
Validation	フォームに入力した値が有効であるかどうかを判断するためのロジックを指定します。Validation 式は、成功を示す場合は NULL を返し、失敗を示す場合は判読可能なエラーメッセージを含む文字列を返します。Validation 規則は、フォームの送信時にのみ評価され、表示を更新したり、再計算したりした場合には評価されません。
Expansion	フォームの送信後にフィールドの値を計算するための式を指定します。多くの場合、Expansion 式は非表示のマークが付けられたフィールドで使用されます。非表示フィールドは、ユーザーが直接編集することができないため、値は Expansion 式を使用して計算できます。
Disable	true と評価された場合に、フィールドと、その入れ子フィールドを無効にする式を指定します。無効化されたフィールドは、フォームに表示されません。このオプションは、ユーザーが特定タイプのリソースを持っているかどうかを判断するときに使用されます。ユーザーがこのようなリソースを持っている場合は、フォームにはそのリソースに適したフィールドが表示されます。
Display Class	このフォームコンポーネントをブラウザに表示するときに使用される HTML コンポーネントクラスを指定します。デフォルトでは、EditForm 表示クラスが選択されます。フォームがエンドユーザーメニューのようなリンクフォームの場合は、Display Class オプションから LinkForm を選択します。 「HTML 表示コンポーネント」の HTML 表示クラスの表を参照してください。
value	プロパティーの属性を指定します。通常は文字列です。
maxLength	この要素の最大文字数を指定します。

表 2-6　フォームツールボックスのデフォルトサービス

サービス	説明
Text	通常のテキスト入力ボックスを表示します。
Secret Text	テキストをアスタリスク (*) として表示します。通常は、パスワードのように暗号化されるデータで使用されます。
Select	1 つの項目を選択できるリストボックスを表示します。このリストボックスの値は、`allowedValues` プロパティーで指定します。
MultiSelect	複数の選択項目を持つテキストボックスを表示します。これは、1 つのボックスに含まれる定義済みの値セットを選択ボックスに移動できる、2 つの部分から構成されるオブジェクトです。左のボックスの値は `allowedValues` プロパティーで指定します。多くの場合、この値は `FormUtil.getResources` などの Java メソッドを呼び出すことで動的に取得されます。複数選択ボックスの右側部分に表示される値には、フィールド名によって特定される関連ビュー属性の現在値が適用されます。
Checkbox	チェックボックスを表示します。チェックマークが付けられると、そのボックスの値は `true` となります。選択されていないボックスの値は false となります。
Label	複数行のテキスト入力ボックスを表示します。
TextArea	1 つまたは複数のラジオボタンの水平リストを表示します。ユーザーが一度に選択できるラジオボタンは 1 つだけです。コンポーネントの値が NULL の場合、または許可される値のいずれとも一致しない場合は、ボタンは選択されません。
Radio	ページにリンクを配置します。
Link	ボタンを表示します。
Button	このフォームで使用されるビューによって定義される変数を参照します。
accountId	複数行のテキスト入力ボックスを表示します。

カスタマイズに関連するその他のトピック

ここでは、次のトピックについて説明します。

- フォームの構造
- フォームのコンポーネント
- フィールドの定義
- フォームの作成に関するガイドライン

フォームの構造

フォームは、XML オブジェクトとして Identity Manager リポジトリに格納されます。各フォームは、次の構造を持つ独自のオブジェクトとして格納されます。

注 フォームの XML 構造を理解している必要はありません。Identity Manager IDE を使用することで、フォームの構造を簡単に操作できます。次に示す情報は、参照のみを目的としています。

次のスタブフォームは、フォームの一般的な構造を示しています。

コード例 2-4

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE Configuration PUBLIC 'waveset.dtd' 'waveset.dtd'>
<!--  id="#ID#UserForm:EndUserMenu" name="End User Menu"-->
<Configuration id='#ID#UserForm:EndUserMenu' name='End User Menu'
reateDate='1012185191296' lastModifier='Configurator'
lastModDate='1013190499093' lastMod='44' counter='0' wstype='UserForm'>
   <Extension>
   <Form name='End User Menu'>
      <Display class='LinkForm'>
         <Property name='title' value='User Self Service'/>
         <Property name='subtitle' value='Select one of the following options'/>
      </Display>
Field content
   </Form>
   </Extension>
<MemberObjectGroups>
   <ObjectRef type='ObjectGroup' id='#ID#Top' name='Top'/>
   </MemberObjectGroups>
</Configuration>
```

注　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">
```

フォームのコンポーネント

次の表は、フォームに表示される順序でフォームコンポーネントを示しています。各フォームコンポーネントの詳細については後述します。

表 2-7　フォームのコンポーネント

フォームコンポーネント	目的
ヘッダー	フォームのオブジェクト定義に関する情報を示します。<Form>、<Extension>、および <Configuration> 要素の開始タグが含まれ、フォームのプロパティー (フォームの呼び出し時に表示される title、subtitle、titleWidth など) を定義します。
フォーム本文	フィールド定義、フォーム関数、およびフォーム変数が含まれます。これは、フォームの中で編集する部分です。
フッター	<Form>、<Extension>、および <Configuration> 要素の終了タグが含まれます。

ヘッダー

フォームヘッダーに含まれる内容は次のとおりです。

- この XML ファイルと関連する DTD を含む、XML 宣言やドキュメント宣言など、XML ファイルに含める標準の導入情報。前述の例には、次のような導入情報が含まれます。

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

  このシステムは、ファイルにこの情報を追加します。編集しないでください。

- <Extension> および <Configuration> 要素の開始タグ。フォームの外観と動作を表現する HTML コンポーネントは、これらの要素で囲まれます。Configuration 要素には、フォームオブジェクトのプロパティーを表現する属性が含まれます。

ヘッダーには、作成日、ファイルを最後に修正したユーザーのログイン、フォームタイプなどの内部識別情報をはじめとする、フォームに関する情報が含まれます。この情報は、通常、ページプロセッサによって生成されます。

注 システムが生成する次の情報は、内部使用限定です。これらの属性は編集しないでください。

表 2-8 フォームヘッダーのコンポーネント

要素	定義	構文 / 例
<Extension>	<Form> 要素を囲むために使用されます。	<Element>…</Element>
<Configuration>	最後の修正日や、このフォームを最後に修正したユーザーのログインなど、フォームオブジェクトを処理するときにシステムが内部的に使用する情報が含まれます。これらの情報のほとんどは、通常、Identity Manager リポジトリに格納されているいずれかの持続オブジェクトと関連付けられています。多くの場合、この情報を編集する必要はありません。	Configuration id='#ID#UserForm:EndUser Menu' name='End User Menu' createDate='1012185191296' lastModifier='Configurator' lastModDate='1013190499093 ' lastMod='44' counter='0' wstype='UserForm'>

フォーム本文

フォーム本文は、次の内容から構成されます。

- title、subtitle、width などのフォームプロパティー。これらのプロパティーは、Form Properties というテーブルに定義されます。
- Field 要素。これは、製品インタフェースでユーザーに表示するフィールドの外観と機能の設定に使用されます。フィールドには、情報を計算するための XPRESS ロジックも含まれます。XPRESS 言語の使用の詳細については、「XPRESS 言語」を参照してください。

次の表は、フォームヘッダーのプロパティーを示しています。

表 2-9　フォームヘッダーのプロパティー

プロパティー	目的
title	フォーム上部に表示されるテキストを指定します。多くの場合、このタイトルは、画面のその他のフォントより大きな太字で表示されます。フォームの title は、Identity Manager ページの下に表示されます。タイトルの表示特性は編集できません。 「フォームのコンポーネント」の節で示した例では、title の値は User Self Service です。
subtitle	このフォームによって定義されるページのフォームタイトルの下に表示されるテキストを指定します。タイトルの表示特性は編集できません。 前述の例では、subtitle の値は Select one of the following options です。
titleWidth	ブラウザウィンドウのタイトルの幅を、ピクセル単位で指定します。 例 <Display> <Property name='titleWidth'> <Integer>120</Integer> </Property> </Display>

次の表は、フォーム本文で使用されるすべての要素を示しています。

表 2-10　フォーム本文で使用される要素

コンポーネント	定義	例
defun	XPRESS 関数を定義します。この要素は、フォーム内のどのフィールド要素からも呼び出すことができます。	<defun name='add100'> <def arg name='x'/> <add>x100 </add> </defun>

表 2-10　フォーム本文で使用される要素 (続き)

コンポーネント	定義	例
defvar	計算結果を保持するための、XPRESS 変数を定義します。	<defvar name='nameLength' <length> <ref>fullname</ref> </length> </defvar>
Display	フィールドの外観を定義する表示コンポーネントを指定します。詳細については、「Display 要素」の節を参照してください。	<Display class='LinkForm'> <Property name='title' value='User Self Service'/> <Property name='subtitle' value='Select one of the following options'/> </Display>
フィールド	フォーム本文で使用される主要要素。詳細については、「Field 要素」の節を参照してください。	<Field name='fullname'/>
FieldRef	含まれるフォームに定義されているフィールドへの参照を設定します。	<FieldRef name='fieldName'/>
Include	別のフォームオブジェクトへの参照を設定します。現在のフォームにこのコンポーネントを含めると、そのフォームに定義されるフィールドを参照し、表示できるようになります。	<Include> <ObjectRef type='UserForm' id='#ID#UserForm:UserFormLibrary'/> </Include>
FormRef	別のフォームオブジェクトへの参照を設定します。	<FormRef name='formName'/>
Namespace	ビューへのショートカットを定義するときに使用されます。フィールド名と参照に、長い名前の代わりに短縮名を使用できます。代替名を使用するときは、名前のあとにコロン (:) を続けます。	<Namespace name='w' value='waveset'/>

Form 要素

<Form> 要素は、すべての Field 要素を囲み、フォームの一意の名前を含みます。前のページに示される要素は、Form の開始タグと終了タグの間に記述されます。

コード例 2-5

```
<Form name='Create User Form'
   <Field name='waveset.accountId'>

additional fields

</Form>

別の例 :

<Form name='Task Launch Form'>
   <Display class='EditForm'>
      <Property name='title' value='Task Launch'/>
      <Property name='subTitle' value='Enter task launch parameters'/>
   </Display>
...
</Form>
```

Display 要素

Form 要素内の Display 要素は、フォームの表示に使用されるコンポーネントを表現します。デフォルトでは、この Display 要素は EditForm コンポーネントで使用されます。Form コンポーネントクラスの変更が必要となることはあまりありませんが、コンポーネントのプロパティーは設定可能です。もっとも多く指定される 2 つのプロパティーは title と subTitle です。

EditForm は、隣接するフィールドのタイトルの幅を設定する adjacentTitleWidth プロパティーもサポートします。このプロパティーが定義されていない場合は、デフォルト値のゼロが適用されます。

adjacentTitleWidth の値をゼロに設定すると、列のタイトルの大きさは自動的に変更されます。ゼロ以外の値を設定した場合は、隣接する列 (たとえば、2 番目と 3 番目の列) のタイトルの幅は adjacentTitleWidth の値に設定されます。

```
<Form name='Default User Form' help='account/modify-help.xml'>

   <Display class='EditForm'>

     <Property name='titleWidth' value='120'>

     <Property name='adjacentTitleWidth' value='60'>

   </Display>
```

Field 要素

Field 要素は、フォーム本文で使用される主要要素です。フィールドは、ユーザーの各属性の定義に使用されます。Field 要素を使用して、フォームフィールドに XPRESS ロジックを含めることができます。フォームの Field 要素の操作の詳細については、「フィールドの定義」の節を参照してください。

次の例は、Email Address というラベルを持つ編集フィールドを作成します。

```
<Field name='waveset.email'>
   <Display class='Text'>
      <Property title='Email Address'/>
      <Property size='60'/>
      <Property maxLength='128'/>
   </Display>
...
</Field>
```

編集フィールドの名前は、多くの場合、フォームで使用されるビュー内のパス式です。この例で waveset.email は、Identity Manager リポジトリ内のユーザーオブジェクトに関連付けられている電子メールアドレスを参照します。

フッター

フッターには、フォームが関連付けられている Identity Manager オブジェクトグループまたは組織に関する情報が含まれます。また、</Form>、</Extension>、および </Configuration> 要素の終了タグ、またはヘッダー内で開始したその他の要素の終了タグも含まれます。前述の例には、次のようなフッターが含まれます。

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

<MemberObjectGroups> は、システムがオブジェクトを格納するオブジェクトグループまたは組織を特定します。オブジェクトグループを指定しない場合は、システムはデフォルトの組織である Top にオブジェクトを割り当てます。多くの場合、フォームを含む Configuration オブジェクトは、次の構文で All グループに含まれます。

```
<MemberObjectGroups>
```

```
    <ObjectRef type='ObjectGroup' name='All'/>
</MemberObjectGroups>
```

フォームフィールドとは

フォーム本文には、Web ページの各要素の外観と動作を定義する `Field` 要素が含まれます。各 `Field` 要素には、それぞれが独自の表示コンポーネントを持つその他のフィールドを含めることができます。

フォームフィールドは、`<Field>` タグのセットに囲まれた、いくつかの部分から構成されます。

- **値の式** : フィールドには、フィールドの値を計算したり、許可される値のセットを定義したりする XPRESS 式を数多く含めることができます。たとえば、フィールドのデフォルト値を定義するときは `<Default>` を使用し、フォームを最初に読み込んだときに計算されるフィールド値を定義するときは `<Derivation>` を使用します。すべてのフィールド要素が式を含むわけではありません。「フィールド名の定義」の節を参照してください。
- **HTML 表示コンポーネント** : 表示コンポーネントは、可視要素をどのように表示するかを決定します。Identity Manager のフォームフィールドでは、フォーム内の `<Display>` 要素で定義された表示コンポーネントが、フォームフィールドの動作と外観を設定します。それぞれのフィールドに設定できる表示コンポーネントは 1 つだけです。これらの表示コンポーネントの詳細については、第 6 章「HTML 表示コンポーネント」を参照してください。
- **Disable 式** : `Disable` 式を使用することで、条件に応じて、フィールドをフォームに含めることができます。`Disable` 式が **true** と評価される場合は、フィールドは無視されます。

変数の作成

定数または静的データの長いリストを含む変数を作成するときは、次の構文を使用します。この構文は、静的なリストを 1 回作成し、それぞれの参照で再利用します。

```
<defvar name='states'>
    <List>
      <String>Alabama</String>
            ...
    </List>
</defvar>
```

参照のたびに新しいリストを作成するときは、`<list><s>Alabama</s>...</list>` という構文を使用します。

注

フォーム変数の値や、Identity Manager オブジェクトに名前を付けるときは、かっこの対応を取る必要があります。オブジェクト名やフォーム変数でかっこを使用する場合は、対応が取れている (つまり「(」のそれぞれに対応する「)」がある) 必要があります。これにより、すべてのフォーム変数を適切に展開できます。かっこの対応が取れていないと、エラーになります。

たとえば、リソース名でかっこの対応が取れていないと、このリソースが割り当てられたユーザーのユーザーオブジェクトを編集または削除することはできません。ただし、特定のリソース名でかっこが対応していないためにエラーが発生した場合には、それに関連するエラーメッセージが表示されます。

フィールドの定義

ここでは、フォームをカスタマイズするときに実行する手順について説明します。次の手順が含まれます。

- フィールド名の定義
- フィールド要素の定義
- 可視フィールドの追加
- フィールドの非表示化。フィールドを非表示に設定すると、フィールドと、その入れ子フィールドはページに表示されなくなります。ただし、フォームの処理によって、フィールドの値は設定されます。
- フィールドの無効化。フィールドを無効にすると、フィールドと、その入れ子フィールドはページに表示されなくなります。また、値の式も評価されません。無効化されたフィールドの値がすでにビューに含まれる場合、その値は変更されません。
- フィールド値の設定
- 関数の呼び出し

次に、設定するフィールド特性について、より詳しく説明します。

フィールド名の定義

リソースに定義されている属性と、Web ページに表示されるテキスト入力フィールドを一致させるには、フィールド名を使用します。リソースを定義すると、システムはリソースのアカウント属性と Identity Manager の属性をマップするスキーママップを設定します。たとえば、Active Directory リソースには、`firstname`、`lastname`、`Office Phone` などの属性が含まれる可能性があります。フォームでこれらの属性を参照するときは、Identity Manager スキーマに記録されている属性名と、ビューから属性へのパスがわかっていなければなりません。

Field 要素の name 属性を定義するには、2 つの方法があります。

- name 属性は、多くの場合に、ユーザービュー内の属性へのパスを含みます。
- name 属性は、フォーム内の別のフィールド、または FieldRef 要素がフィールドを参照できるように、フィールドの特定に使用されます。これは、フィールドが別のフィールドのコンテナを表すように定義され、ビューのその他の属性に対応しない場合に行われます。

Field 名がビューのパス式を表すか、単なる参照名を表すかは、Display 要素で選択した class 属性の値によって決まります。編集するコンポーネントクラスの名前が Display クラスである場合は、その名前はビューのパス式であると考えられます。コンポーネントクラスの詳細については、「HTML 表示コンポーネント」の節を参照してください。

ビュー属性へのパス式の作成

Field 名を定義するときは、通常、ユーザービュー内の属性へのパス (パス式) を指定します。これらの属性のリストについては、「Identity Manager のビュー」を参照してください。

次のフィールド定義は、Identity Manager の電子メールアドレスを編集できるようにテキストフィールドを表示します。

```
<Field name='waveset.email'>
   <Display class='Text'>
      <Property name='size' value='60'/>
   </Display>
</Field>
```

waveset.email という文字列は、Identity Manager のリポジトリに格納される電子メールアドレスをターゲットとした、ユーザービューのパス式です。

例 :

このフィールド例は、特定のリソースアカウント用に定義されている電子メールアドレスを編集します。フィールド名は、アカウント内のリソースを参照します。

```
<Field name='accounts[Active Directory].email'>
   <Display class='Text'>
      <Property name='size' value='60'/>
   </Display>
</Field>
```

accounts[Active Directory].email という文字列は、指定されたリソースのアカウント属性に関する情報を保持するユーザービュー内の別の場所へのパス式です。この例では、リソースの名前は Active Directory です。

例 :

このフィールド例は、スキーママップの左側部分に email という属性を持つ、Identity Manager を含むすべてのリソースの電子メールアドレスを定義します。

```
<Field name='global.email'>
   <Display class='Text'>
      <Property name='size' value='60'/>
   </Display>
</Field>
```

参照するフィールドの指定

フィールドに名前を付けることで、別のフィールドからそのフィールドの値を参照できます。別のフィールドからフィールド値を参照するときは、<ref></ref> のタグセットを使用します。次の例は、firstname フィールドと lastname フィールドの値を取得し、lastname, firstname のように、フィールドの文字列、コンマ、および空白文字を連結して、fullname フィールドの値を導出します。文字列の指定には、<s> タグを使用します。

コード例 2-6

```
<Field name='global.firstname'>
   <Display class='Text'/>
</Field>
<Field name='global.lastname'>
   <Display class='Text'/>
</Field>
<Field name='global.fullname'>
   <Expansion>
      <concat>
         <ref>global.lastname</ref><s>, </s>
         <ref>global.firstname</ref>
      </concat>
   </Expansion>
</Field>
```

すべての Field 名がビューのパス式を表すわけではありません。一部のフィールドは、別のフィールドのコンテナを表すように定義され、ビューのその他の属性に対応しません。このような場合は、FieldRef 要素によって参照できるように、Field 名を使用してフィールドを特定します。フィールドを参照する必要がない場合は、名前を指定する必要はありません。

たとえば、フォームボタンはアクションを実行しますが、値を持たず、別のフォームから参照する必要がありません。このため、フィールド名の指定も必要ありません。

```
<Field>
   <Display class='Button'>
      <Property name='label' value=' 再計算 '/>
      <Property name='command' value=' 再計算 '/>
   </Display>
</Field>
```

ユーザービューの詳細については、「ユーザービューとフォーム」の節を参照してください。

フィールドの表示プロパティー

Display 要素は、すべての可視フォームフィールドに共通です。Display 要素には、ブラウザに表示されるフィールドの特性を定義する Property 要素が含まれます。フィールドに **true** と評価される Disable 要素が含まれる場合を除き、フォームの Display 要素を定義すると、その要素を画面に表示できます。フィールドを再計算するフォームが画面に表示されないように設定できる場合に、別のフィールドまたは値が設定されるまでフォームを表示する、という条件を指定することができます。「フィールドの無効化」の節を参照してください。

Display

可視フィールドのクラスとプロパティーを表現します。この要素は、インスタンス化するコンポーネントクラスと、そのインスタンスに割り当てるプロパティー値のセットを指定します。

```
<Display class='Text'>
   <Property name='size' value='20'/>
   <Property name='maxLength' value='100'/>
</Display>
```

Display 要素の class 属性は、Component クラスの名前である必要があります。デフォルトでは、これらのクラスは Applet、Button、DatePicker などを含み、com.waveset.ui.util.html パッケージに格納されています。デフォルトのすべてのクラスとその説明については、「HTML 表示コンポーネント」の「基本コンポーネン

トクラス」の節を参照してください。このパッケージに含まれないクラスを参照するときは、class 属性の完全修飾クラス名を指定します。このマニュアルで説明するすべてのクラスはデフォルトパッケージに含まれるので、修飾名を指定する必要はありません。

プロパティー

Display 要素内に指定されます。Property の値は、コンポーネントに割り当てられるプロパティーの名前と、その値を定義します。プロパティー名は、常に name 属性で指定されます。

Display 要素の Property 値の指定

Display 要素の Property の値は、次の方法で指定できます。

- value 属性
- XML オブジェクト言語
- 値を指定する式

ほとんどのプロパティーの値は、value 属性を使用して、値に適切なタイプを割り当てるようにシステムに強制することで設定できます。

value 属性の使用

プロパティーの値を設定するもっとも一般的な方法は、value 属性の使用です。value 属性の値は文字列として認識されますが、システムは必要に応じてコンポーネントに適したデータ型を強制します。前述の例では、size には整数値 20 が設定され、maxLength には整数値 100 が設定されています。

次の例では、SimpleTable を使用して、いくつかのサブフィールドを構成するフィールドを作成します。XML フォームでもっとも一般的に使用される Container コンポーネントは、SimpleTable と ButtonRow です。

コード例 2-7

```
<Field name='SelectionTable'>
   <Display class='SimpleTable'>
      <Property name='columns'>
         <List>
             <String>Account</String>
             <String>Description</String>
         </List>
      </Property>

   </Display>
<Field name='accounts[LDAP].selected'>
   <Display class='Checkbox'>
      <Property name='label' value='LDAP'/>
   </Display>
</Field>
<Field>
   <Display class='Label'>
      <Property name='text' value='Primary Corporate LDAP Server'/>
   </Display>
</Field>
<Field name='accounts[W2K].selected'>
   <Display class='Checkbox'>
      <Property name='label' value='Windows 2000'/>
   </Display>
</Field>
<Field>
   <Display class='Label'>
      <Property name='text' value='Primary Windows 2000 Server'/>
   </Display>
</Field>
</Field>
```

Display 要素には、ゼロ個以上の Property 要素が含まれます。これらの要素は、そのコンポーネントに割り当てられるプロパティーの名前と値を定義します。Property 要素の名前は、常に name 属性で指定されます。プロパティー値の設定には、多くの場合、value 属性が使用されます。value 属性の値は文字列として認識されますが、必要に応じて、コンポーネントに適したデータ型が強制されます。

XML オブジェクト言語の使用

XML オブジェクト言語を使用してプロパティー値を指定することもできます。これは、主にリストの値を指定するときに便利です。この言語の構文を使用することで、いくつかの標準 Java オブジェクトと、Identity Manager によって定義されるその他オブジェクトを表現できます。

もっとも一般的な Java XML オブジェクトは次のとおりです。

- `List`
- `Map`
- `MapEntry`
- `String`
- `Integer`
- `Boolean`
- `Object`

XML オブジェクト構文を使用してプロパティー値を指定するときは、要素が `Property` 要素内に配置されます。XML オブジェクト言語の詳細については、「XML オブジェクト言語」を参照してください。

コード例 2-8

```
<Property name='size'>
   <Integer>10</Integer>
</Property>

<Property name='title'>
   <String>New Password</String>
</Property>

<Property name='leftLabel'>
   <Boolean>true</Boolean>
</Property>

<Property name='allowedValues'>
   <List>
      <String>Texas</String>
      <String>Iowa</String>
      <String>Berkshire</String>
   </List>
</Property>
```

リストの値を認識するすべてのプロパティーは、List 要素を認識します。また、ほとんどの属性は、リストを指定するコンマ区切りのリスト構文を認識します。

値を計算する式の使用

Property の値を式で指定することもできます。これにより、固定のリテラル値と、ページプロセッサによって定義される変数の値を組み合わせるなどの方法で、実行時に値を計算することができます。例:

コード例 2-9

```
<Property name='title'>
   <concat>
      <s>Welcome </s>
      <ref>waveset.accountId</ref>
      <s>, select one of the following options.</s>
   </concat>
</Property>
```

この例では、waveset.accountId が変数を参照しています。システムがこのコンポーネントの HTML を生成するときに、ページプロセッサシステムが waveset.accountId 変数の値を供給します。参照できる変数の名前は、ページプロセッサによって定義されます。ほとんどの場合、これらの変数の定義には、XML フォームで使用される *view* が使用されます。フォームは、そのフォームを使用するビューと、そのビューによって定義される参照属性のみに注意するだけで設計できます。

Disable 要素

ブール型の値を計算します。計算結果が true となる場合は、現在のフォームの処理で、そのフィールドと、すべての入れ子フィールドが無視されます。

Disable 要素で時間のかかる可能性がある処理を作成しないでください。これらの式は、フォームが再計算されるたびに実行されます。代わりに、この計算を頻繁には実行しない別のフォーム要素を使用してください。

注	display.session および display.subject 変数は、Disable フォーム要素で使用できません。

例

この例は、<Disable> 要素内の式を使用してフィールドの可視性を制御するフィールド定義を示しています。accountInfo.typeNames は、そのユーザーが割り当てられているすべてのリソースのタイプの検出に使用されます。これらのタイプは、ユーザーのすべてのリソースタイプのリストとして返されます。返されるタイプ名のリストに Solaris が含まれるときは、そのフィールドは画面に表示されます。含まれない場合は、そのフィールドは無効化されます。

コード例 2-10

```
<Field name='HomeDirectory' prompt='Home Directory'>
   <Display class='Text'/>
      <Disable>
         <not>
            <contains>
               <ref>accountInfo.typeNames</ref>
               <s>Solaris</s>
            </contains>
          </not>
   </Disable>
</Field>
```

Disable 要素は一般的に、フォーム上のほかのフィールドの値を確認するために使用されます。別のフィールドが特定の値を持つ場合は無効化し、そうでない場合は無効化しません。NULL の場合は特殊なケースになります。多くの場合、参照している別のフィールドはほかの入力フィールドに基づいて計算されます。

コード例 2-11

```
<Field name='special value subfield'>
   <Comment>otherField の値が「special value」の場合のみ表示します
  </Comments>
  <Disable>
   <neq>
     <ref>otherField</ref>
     <s>special value</s>
   </neq>
  </Disable>
...
</Field>
```

コード例 2-12

```
<Field name='account correlation rule'>
  <Comment> リソースでの同期でアカウント相関規則がサポートされている場合は、その規則を選択できます。そうでない場合、フィールドは表示されません。処理規則が選択された場合は、相関規則が実行されないため、フィールドは表示されません。
   </Comments>
  <Disable>
   <or>
    <isnull>
     <ref>resourceAttributes[correlationRule].displayName</ref>
    </isnull>
    <notnull>
     <ref>resourceAttributes[processRule].value</ref>
    </notnull>
   </or>
  </Disable>
...
</Field>
```

Default 要素

NULL 以外の値がフィールドに設定されていない場合にのみ、このフィールドの値として使用される値を計算します。`Default` は基本的には `Derivation` と同じですが、現在値が NULL 以外の場合にのみ値が適用されます。Default 式は、次の場合に計算されます。

- フォームを初めて読み込む
- 1 つまたは複数のリソースからデータを取得する
- フィールドの値が NULL 以外の状態で、フォームを再計算または保存する。

例

この例は、文字列操作式を使用して、ユーザーの名の頭文字と姓から構成されるデフォルトのアカウント ID を返すフィールド定義を示しています。

コード例 2-13

```
<Field name='waveset.accountId'>
   <Display class='Text'>
      <Property name='title' value='AccountID'/>
   </Display>
   <Default>
      <concat>
         <substr>
            <ref>accounts[AD].firstname</ref>
               <i>0</i>
               <i>1</i>
            <ref>accounts[AD].lastname</ref>
```

コード例 2-13

```
         </substr>
      </concat>
   </Default>
</Field>
```

Derivation 要素

フィールドの値を無条件で計算します。Derivation 式が評価されると、フィールドの現在値は常に置き換えられます。

Derivation 式は、フォームが最初に読み込まれるとき、あるいは、1 つまたは複数のリソースからデータが返されるときに計算されます。

次の例は、条件ロジックを使用して、値のセットを別のセットにマップするフィールド定義を示しています。このフィールドを処理すると、<Derivation> 要素内の式が評価され、リソースから返される場所のコードに基づいて、このフィールドに表示される説明的な値が決定されます。

コード例 2-14

```
<Field name='location'>
   <Display class='Text'>
      <Property name='title' value='Location'/>
   </Display>
   <Derivation>
         <switch>
            <ref>accounts[Oracle].locCode</ref>
            <case>
               <s>AUS</s>
               <s>Austin</s>
            </case>
               <s>HOU</s>
               <s>Houston</s>
            </case>
            <case>
               <s>DAL</s>
               <s>Dallas</s>
            </case>
            <case default='true'>
               <s>unknown</s>
            </case>
         </switch>
   </Derivation>
</Field>
```

Expansion 要素

フィールドの値を無条件で計算します。この要素は、式の評価のタイミングが Derivation 要素とは異なります。

Expansion ステートメントは、次の場合に計算されます。

- ページが再計算される
- フォームが保存される

次の例は、条件ロジックを使用して、前述の例の `location` フィールドから得られる値を、Oracle リソースに格納される 3 文字の略号に変換するフィールド定義を示しています。フィールド名の違いに注意してください。location フィールドの値は、どのリソースにも格納されません。これは、別のフィールドの計算に使用されます。

コード例 2-15

```
<Field name='accounts[Oracle].locCode'>
   <Expansion>
      <switch>
         <ref>location</ref>
         <case>
            <s>Austin</s>
            <s>AUS</s>
         </case>
         <case>
            <s>Houston</s>
            <s>HOU</s>
         </case>
         <case>
            <s>Dallas</s>
            <s>DAL</s>
         </case>
      </switch>
   </Expansion>
</Field>
```

Validation 要素

フォームに入力された値が有効であるかどうかを検証します。Validation 規則は、フォームを送信するたびに評価されます。

この例では、ユーザーの郵便番号が 5 桁であるかどうかが Validation 規則によって確認されます。

コード例 2-16

```
<Validation>
   <cond>
      <and>
      <eq><length><ref>global.zipcode</ref></length>
            <i>5</i>
      </eq>
            <gt><ref>global.zipcode</ref><i>99999</i></gt>
      </and>
      <null/>
      <s>zip codes must be five digits long</s>
   </cond>
</Validation>
```

編集フィールドとコンテナフィールド

Field 要素に指定された Display 要素は、フィールドの表示に使用されるコンポーネントを表現します。これには、次の 2 種類のフィールドが使用されます。

- **編集フィールド**: 編集する特定の値に関連します。
- **コンテナフィールド**: 1 つまたは複数のフィールドを囲みます。

編集フィールドは必ず名前を持ち、常に Text や Checkbox など、いずれかの編集コンポーネントとともに使用されます。

編集フィールドの例

```
<Field name='waveset.email'>
   <Display class='Text'>
      <Property title='Email Address'/>
      <Property size='60'/>
      <Property maxLength='128'/>
   </Display>
 ...
</Field>
```

編集フィールドの名前は、多くの場合、フォームで使用されるビュー内のパス式です。前述の例で waveset.email は、Identity Manager リポジトリ内のユーザーオブジェクトに関連付けられている電子メールアドレスを参照します。

コンテナフィールドは名前を持たない場合があり、常に ButtonRow、SimpleTable、EditForm などのいずれかのコンテナコンポーネントとともに使用されます。

よく使用されるコンテナのタイプには、1 つの列にタイトル、別の列にコンポーネントを持つ HTML テーブルを作成する EditForm コンテナがあります。これらのタイトルは title プロパティーに定義され、そのフォームに関連付けられている Identity Manager ページに表示されます。

フィールドの無効化

フィールドを無効にすると、フィールドと、その入れ子フィールドはページに表示されなくなります。その値の式は評価されないか、またはどの global.* 属性にも組み込まれません。無効化されたフィールドの値がすでにビューに含まれる場合、その値は変更されません。

```
<Disable></Disable>
```

例

```
<Field name='waveset id'>
   <Display class='Text'>
      <Property title='accountId'>
   </Display>
   <Disable>
      <eq><ref>userExists</ref><s>true</s></eq>
   </Disable>
</Field>
```

注 Disable 式は、その他の式よりも頻繁に評価されます。このため、Disable 式は、比較的単純に記述してください。データベースのルックアップのように、負荷の大きい処理を実行する Java クラスは呼び出さないようにしてください。

Disable 式の規則でフィールドを参照するときは注意が必要です。誤って設定すると、コンテナ内のフィールドが無効化される場合があります。

フィールドの非表示

フィールドを非表示に設定すると、フィールドと、その入れ子フィールドはページに表示されなくなります。ただし、フォームの処理によって、フィールドの値は設定されます。

フィールドに Display クラスを割り当てないだけで、フィールドは簡単に非表示にできます。

```
<Field name='field A'/>
```

フィールド値の計算

フィールドの値は、別のフィールドの値や、任意の論理式の値から計算できます。たとえば、ユーザーの姓、ミドルネームのイニシャル、および名からフルネームを導出できます。

コード例 2-17

```
<Field name='global.fullname'>
   <Expansion>
      <concat>
         <ref>global.firstname</ref>
         <s> </s>
         <ref>global.middle</ref>
         <ref>global.lastname</ref>
         <s> </s>
      </concat>
   </Expansion>
</Field>
```

デフォルト値の設定

ユーザーの名の頭文字と、姓の最初の 7 文字を使用して、電子メールアドレスを設定することができます。この例では、それらを連結する前に値が設定されていることを確認するため、システムが追加チェックを行います。この追加チェックの目的は次のとおりです。

- アカウントの最初の作成時にのみ、電子メールアドレスを設定できるようにする。
- 姓、名の各フィールドに値が設定されていることを確認する。

コード例 2-18

```
<Field name='global.email'>
   <Default>
      <and>
         <notnull><ref>global.firstname</ref></notnull>
         <notnull><ref>global.lastname</ref></notnull>
      </and>
         <concat>
         <downcase>
            <substr>
            <ref>global.firstname</ref>
            <i>0</i>
            <i>1</i>
            </substr>
         </downcase>
         <downcase>
            <substr>
            <ref>global.lastname</ref>
            <i>0</i>
            <i>6</i>
            </substr>
         </downcase>
         <s>@waveset.com</s>
      </concat>
   </Default>
</Field>
```

フィールド値の導出

一部のフィールドは、別のフィールドを計算するときにだけ、フォーム上で使用されます。これらのフィールドは、ユーザーが属すどのリソースにも格納できません。ユーザーレコードを編集すると、それぞれのリソースが参照され、属性のフィールド値が取り込まれます。計算に使用されるフィールドの値を取り込むときは、Derivation 規則を記述します。

例

電話番号のフィールドは、1 つのテキストボックスとしてフォームに表示されます。しかし、より高度なフォームには、リソースに保存する電話番号の計算に使用される、市外局番と電話番号の 3 つのフィールドが含まれる場合があります。

電話番号を表す単純な例では、次のようなフォームフィールドを使用できます。

コード例 2-19

```
<Field name='P1'>
   <Display class='Text'>
      <Property name='title' value='Office Phone Number'/>
      <Property name='size' value='3'/>
      <Property name='maxLength' value='3'/>
   </Display>
</Field>
<Field name='P2'>
   <Display class='Text'>
      <Property name='title' value='-'/>
      <Property name='size' value='3'/>
      <Property name='maxLength' value='3'/>
   </Display>
</Field>
<Field name='P3'>
   <Display class='Text'>
      <Property name='title' value='-'/>
      <Property name='size' value='4'/>
      <Property name='maxLength' value='4'/>
   </Display>
</Field>
<Field name='global.OfficePhone'>
   <Expansion>
      <concat>
         <ref>P1</ref><s>-</s>
         <ref>P2</ref><s>-</s>
         <ref>P3</ref>
      </concat>
   </Expansion>
</Field>
```

例

次の例は、前述の例で定義されている P1 フィールドのフィールド定義を拡張します。これは、電話番号の属性をフォームに読み込む方法を定義し、それを 3 つのフィールドの表示に展開します。

コード例 2-20

```
<Field name='P1'>
   <Display class='Text'>
      <Property name='title' value='Office Number'/>
      <Property name='size' value='3'/>
      <Property name='maxlength' value='3'/>
   </Display>
</Field>
```

ユーザーが Identity Manager にデータを入力すると、データが適切に入力されかどうかがフォームによって確認されます。しかし、リソースに直接入力された場合は、リソースが同じ要件を満たしているかどうかについて Identity Manager が検証することはできません。たとえば、長年にわたって、管理者が電話番号を 123-4567 (8 文字) と入力したり、123-123-4567 (12 文字) または (123) 123-4567 (14 文字) のように入力したりしている可能性もあります。

例

`OfficePhone` フィールドの定義は、これまでの例と同様ですが、Derivation 規則を使用するには、3 つのフィールド (`P1`、`P2`、および `P3`) のそれぞれを更新する必要があります。この例は、`P1` フィールドの更新を表しています。

コード例 2-21

```
<defvar name='lenOfficePhone'>
   <length><ref>Office Phone</ref></length>
</defvar>
<Field name='P1'>
   <Display class='Text'>
      <Property name='title' value='Office Phone Number'/>
      <Property name='size' value='3'/>
      <Property name='maxLength' value='3'>
   </Display>
   <Derivation>
      <or>
       <cond><eq>
       <ref>lenOfficePhone</ref>
          <s>8</s></eq>
          <s> </s></eq>
       </cond>
       <cond><eq>
          <ref>lenOfficePhone</ref>
             <s>12</s></eq>
               <substr>
                <ref>Office Phone</ref>
                <i>0</i>
                <i>1</i>
                </substr>
       </cond>

       <cond><eq>
               <ref>Office Phone</ref>
               <i>0</i>
               <i>1</i>
            </substr>
      </cond>
      </or>
   </Derivation>
</Field>
```

コード例 2-21

```
          <ref>lenOfficePhone</ref>
          <s>14</s></eq>
            <substr>
               <ref>Office Phone</ref>
               <i>0</i>
               <i>1</i>
            </substr>
      </cond>
      </or>
   </Derivation>
</Field>
```

フィールドを計算するときは、データの現在の形式と、リソースの品質に注意してください。新規ユーザーを作成するときは、正しいフィールド値の確認は比較的容易です。リソースから読み込む場合に、既存のデータをフィールドに適合させることのほうがずっと困難です。どのフィールドにも Derivation 規則を適用し、読み込むときに属性の形式を確認することができます。

フィールドの再計算

ユーザーがフォームを操作するときに、システムは何度もフィールドを計算します。フィールドの計算は、フィールドを最初に表示したときにデフォルト値を設定するために行われ、フォームの計算は、ユーザーが「**保存**」をクリックしたときに行われます。これ以外に、「ユーザーの編集」ページの「**再計算**」をクリックすると、フォームが評価されます。また、action フィールドを使用してフォームを評価することもできます。

例

```
<Field>
   <Display class='Button'>
      <Property name='label' value=' 再計算 '/>
      <Property name='command' value=' 再計算 '/>
   </Display>
</Field>
```

システムがフィールドの値を再計算するように設定するときは、次のように、Display クラス要素の action を true に設定します。

```
<Display class='Select' action='true'>
```

この値は、ユーザーが選択またはクリックするフィールドのみに追加します。テストエリアやテキストエリアのフィールドに追加しないでください。フィールドに action=true が設定されると、ブラウザでフィールドを変更するたびにフォームが再計算されます。

例

```
<Field name='Region'>
   <Display class='Select' action='true'>
      <Property name='title' value='Geographic Region'/>
      <Property name='allowedvalues' value='North, South,
Central, Midwest'./>
<Property name='nullValue' value='Select a region'/>
   </Display>
</Field>
```

フォームの作成に関するガイドライン

新しいフォームの構造を作成したり、既存のフォームを編集したりするときは、次のガイドラインに注意してください。

- **ページに表示する順序で、フィールド要素を一覧にします**。ブラウザに表示される要素の順序は、フォーム内のフィールド要素の順序によって決定されます。
- **参照先のフィールドは、参照元のフィールドの前に配置します**。フィールドが別のフィールドの値を参照する式を持つ場合は、参照元のフィールドを参照先のフィールドのあとに配置します。
- **論理的に true と評価される場合、無効化されたフィールドは無視されます**。Disable 式を持つすべてのフィールドは評価されます。Disable 式の結果が論理的に true となる場合、フォームの評価時にそのフィールドは無視されます。

フォームフィールド内の式の最適化

フォームで実行される一部のアクティビティーは、Identity Manager の外部にあるリソースを呼び出すことができます。これらのリソースへのアクセスは、Identity Manager のパフォーマンスに影響します。特に、グループや電子メール配信のリストのコンパイルのように、結果が値の長いリストになる場合は大きく影響します。このような呼び出し実行中のパフォーマンスを改善する方法については、「フィールドデータを取得する Java クラスの使用」の節のガイドラインを参照してください。

シナリオの例

次に、式を最適化する例を示します。

Identity Manager に格納されていない情報や、リソースアカウント属性としてアクセスできない情報をデータベースに照会する場合の一般的な手順は次のとおりです。

1. データベースアクセスを実行する Java クラスを記述します。
2. Java クラスを呼び出す Default 式を使用するフォームフィールドを定義します。
3. 非表示の値を参照します。

フィールドデータを取得する Java クラスの使用

情報の取得時に呼び出されるメソッドを持つ Java クラスを記述します。次の節「非表示フォームフィールドの定義」で紹介する例は、カスタムメソッドである `getJobGrade` メソッドを使用します。このカスタムクラスは、`idm\WEB-INF\classes\com\waveset\custom` ディレクトリ構造に配置してください。システムにこれらのディレクトリが存在しない場合は、作成する必要があります。

このクラスを記述するときは、次のガイドラインに注意してください。

- データベース要求のように負荷の大きい操作を実行するメソッドは、非表示フォームフィールドの `Default` 式から呼び出すようにします。こうすることで、フォームを最初に読み込むときに値をビューに格納できます。格納された値は、データベースのオーバーヘッドなしで何度も参照できます。
- 呼び出されるメソッドが静的に宣言されていない場合は、次の例のように、新しい要素を使用して最初にクラスをインスタンス化します。

非表示フォームフィールドの定義

まず、Java クラスを呼び出す Default 式を使用する非表示フォームフィールドを、フィールド定義に `Display` クラスを含めずに定義します。

```
<Field name='jobGrade'>
   <Default>
      <invoke name='getJobGrade'
class='com.waveset.custom.DatabaseAccessor'>
         <ref>waveset.accountId</ref>
      </invoke>
   </Default>
</Field>
     </Derivation>
```

Default 式は、`jobGrade` 属性の値がビューに含まれない場合にのみ評価されます。一度 Default 式を実行すると、結果が `jobGrade` に格納され、再実行されません。

要素の「フォーム要素」ダイアログで次の操作を行います。

1. 「Display Class」メニューから「非表示」を選択します。

2. 「**OK**」をクリックします。

Hidden 表示クラスは、HTML の `<input type=hidden'/>` コンポーネントに相当します。このコンポーネントは、1 つの値をとるデータ型のみをサポートします。複数の値をとるデータ型を確実に直列化および直列化復元する方法を持たないためです。

List 要素を文字列として表示する場合は、次の例のように String 要素に明示的に変換する必要があります。

```
<Field name='testHiddenFieldList' >
   <Display class='Hidden'/ >
   <Derivation>
     <invoke name='toString'>
       <List>
         <String>aaaa</String> <String>bbbb</String>
       </List>
     </invoke>
   </Derivation>
</Field>
```

非表示属性の参照

非表示属性を定義すると、次のように別の式からその属性を参照できるようになります。

```
<Field name='secureKey'>
   <Disable><lt><ref>jobGrade</ref><i>10</i></lt></Disable>
     ...
</Field>
```

XPRESS の `defvar` 変数を使用して計算結果を保持することもできますが、通常は非表示フォームフィールドを使用する場合ほど効率的ではありません。

繰り返し時の変数の最適化に関する注意点

XPRESS 変数の値は、通常、フォームフィールドに対する 1 回の実行の間だけ持続します。このため、Expnasion 式に変数を使用することはできますが、それに続く Derivation 式では使用できません。計算結果の値を複数回の実行でも有効にする必要があるときは、代わりに非表示フォームフィールドを使用します。非表示フィールドの値はビューに格納され、変数セッションをキャンセルまたは保存するまで持続します。

新しいリソースとユーザーの自動リンク設定の無効化

Identity Manager には、ユーザーに新しいリソースを割り当てるときに、既存のアカウントとのリンクを制御する方法が用意されています。

ユーザーに新しいリソースを割り当てるときに、ID を割り当てられたアカウントがすでにリソースに存在する場合は、Identity Manager はデフォルトで、そのアカウントを自動的に Identity Manager ユーザーにリンクしてからプロビジョニングを継続します。反対に、この自動リンク設定を無効にし、ユーザーの新規アカウントを作成するときに別のアカウント ID を入力することもできます。

ユーザーに新規アカウントをどのようにリンクさせるかは、次の 2 つの方法で制御できます。

- ユーザーフォームで、この情報の手動リンク設定を有効にする
- プロビジョニング時の自動リンク設定を回避する

ユーザーフォームでの手動リンク設定の有効化

手動リンク設定を有効にする方法は次のとおりです。

1. 各ユーザーフォームにプロパティー定義を設定します
2. 標準フォームライブラリに含まれるフィールドを参照します

手順 1: プロパティー定義の設定

フォームの先頭に次のようなプロパティー定義を追加します。

```
<Form>
   <Properties>
      <Property name='InteractiveLinking' value='true'/>
   </Properties>
   ...
</Form>
```

手順 2: 標準フォームライブラリに含まれるフィールドの参照

フォーム内の任意の場所にフィールド参照を追加します。次に例を示します。

```
<FieldRef name='DiscoveredAccountFields'/>
```

このフィールドを参照するには、ユーザーフォームに次の Include ステートメントを設定してください。この Include 式は、通常、すべてのユーザーフォームにすでに存在します。

```
<Include>
   <ObjectRef type='UserForm' name='User Library'/>
</Include>
```

これらのフォーム変更を完了すると、Identity Manager はフォームの表示を更新するたびに、また、フォームを保存する前に、既存のアカウントを調べます。Identity Manager が既存のアカウントを検出すると、フォームの最上部に警告メッセージが表示され、検出されたアカウントごとに新しいフィールドが挿入されます。これらの新規フィールドには、リンクを設定するアカウントを手動で指定するためのチェックボックスがあります。

さらに、Identity Manager は、各属性のフィールドをリソースのアイデンティティーテンプレート内に生成します。このフィールドを使用して、アカウントに別のアイデンティティーを指定することができます。Identity Manager は既存アカウントの属性をフェッチし、それをビューに含めます。

これらの属性は、`MissingFields` 参照または独自のカスタムフィールドを使用して表示できます。存在しないアカウントには別のアイデンティティーを指定するか、フォームを保存する前に既存アカウントとのリンクを設定できるようにチェックボックスを選択してください。

プロビジョニング時の自動リンク設定の回避

対話的でない方法でワークフローからプロビジョニングを行うときは、Identity Manager にアカウントのリンク設定を自動的に行わせるかどうかも制御できます。`checkinView` の呼び出しに `NoLinking` ビューオプションを渡すことで、リンクの自動設定を回避できます。このオプションは、いくつかの方法で指定できます。

- 次のように、このオプションを引数として WorkflowServices メソッドに渡します。

```
<Action application='com.waveset.provision.WorkflowServices'>
   <Argument name='op' value='checkinView'/>
   <Argument name='view' value='$(user)'/>
   <Argument name='NoLinking' value='true'/>
</Action>
```

- このオプションをビューの属性として設定します。この場合のビュー属性の名前は、`viewOptions.NoLinking` となります。この属性をワークフローに設定するときは、次のように XPRESS ロジックを使用します。

```
<set name='user.viewOptions.NoLinking'>
  <s>true</s>
</set>
```

結果ページでのクリアテキストによる属性の表示の回避

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。

属性が結果ページにクリアテキストとして表示されないようにするには、その属性をシークレット属性として登録します。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>
```

フォームからのリソースメソッドの呼び出し

invoke メソッドを使用することで、フォームからリソースのメソッドを呼び出すことができます。

invoke メソッドを呼び出すには、クラス名とメソッド名を指定します。invoke タグ内部のメソッドには、次の例のように引数を渡すこともできます。

コード例 2-22

```
<Default>
   <block>
       <defvar name='vmsResName'>
          <index i='0'>
           <ref>accountInfo.accounts[type=vms].name</ref>
          </index>
      <defvar>
          <invoke name='callResourceMethod' class='com.waveset.ui.FormUtil'>
              <ref>display.session</ref>
              <ref>vmsResName</ref>
          <null/>
          </invoke>
</Default>
```

フィールドの「フォーム要素」ダイアログで次の操作を行います。

1. 「Display Class」メニューから「Javascript」を選択します。
2. 「**OK**」をクリックします。

別のフォームからのフォームの参照

<FormRef> 要素を使用することで、完全なフォームではなく、別のフォーム内の特定のフィールドを参照できます。

外部フォームからの別のフォームを含めるには、<FormRef> 要素を使用します。次の例は、MissingFields というフォームを呼び出します。

```
<FormRef name='MissingFields'/>

   <FieldRef name='AuthenticationAnswers'/>

   <FieldRef name='AccountInformation'/>

   <Field name='waveset.backgroundSave'>

      <Display class='Hidden'/>

   </Field>
```

別のフォームからのフィールドの参照

<FieldRef> 要素を使用することで、完全なフォームではなく、別のフォーム内の特定のフィールドを参照できます。

外部フォームからの特定のフィールドを含めるには、<FieldRef> 要素を使用します。このとき、次の項目を指定します。

- フィールドが存在するフォームの名前。このフォーム名は、<ObjectRef> 要素を使用して、フォームヘッダーの Include セクションのリストに含めてください。フォーム名 (UserForm) と一意の設定 ID は、プロパティータイプで指定します。name プロパティーは、後で参照するフィールドの名前を識別します。
- ページ上に表示する位置に対応したフォームセクションに挿入されるフィールド名。

```
<Include>
   <ObjectRef type='UserForm'
      id='#ID#04F5F14E01889DFE:2E5C94:F131DD723D:-7FE4'
      name='Password Library'/>
   <ObjectRef type='UserForm'
      id='#ID#04F5F14E01889DFE:2E5C94:F131DD723D:-7FE3'
       name='Account Summary Library'/>
   <ObjectRef type='UserForm'
       id='#ID#UserForm:UserFormLibrary'/>
    <ObjectRef type='UserForm' name='Global Attributes'/>
</Include>
```

次の例では、ページ上に表示する位置に対応したフォームセクションにフィールド名が挿入されています。

```
<Field name='global.fullname' hidden='true'>
    <Expansion>
      <cond>
        <and>
           <ref>global.firstname</ref>
           <ref>global.lastname</ref>
        </and>
        <concat>
           <ref>global.firstname</ref>
              <s> </s>
           <ref>global.lastname</ref>
        </concat>
      </cond>
    </Expansion>
</Field>
```

次の例の <FieldRef> 要素は、参照する属性の名前を識別します。

```
<Field>
    <Disable>
       <isnull>
           <ref>waveset.id</ref>
       </isnull>
    </Disable>
    <FieldRef name='DynamicChangePasswordFields'/>
</Field>
```

フォームの編集

フォームを編集することで表示特性を変更したり、論理処理を追加してフィールドやコンポーネントを選択したりできます。ここでは、次の 2 つのカテゴリに分けて、フォーム関連編集タスクについて説明します。

- **Display 要素の操作** : ここでは、Identity Manager のフォームを編集するときに、特にユーザーに表示される基本ページコンポーネントの表示特性の変更について説明します。このようなコンポーネントには、ボタン、ラジオボタン、チェックボックスなどがあります。
- **非表示コンポーネントの操作** : このセクションでは、バックグラウンド処理や、表示されるフォームへの論理処理の追加などの目的で Identity Manager のフォームに追加される、HTML 要素コンポーネントについて説明します。このような要素には、`<Disable>` コンポーネントや `<Expansion>` コンポーネント、FormUtil メソッドなどがあります。

タスク関連のセクションで説明される HTML コンポーネントは、「HTML 表示コンポーネント」でアルファベット順に説明されています。

Display 要素の操作

Identity Manager フォームでの変更や追加がもっとも多い Display 要素は、ボタン、フィールド、およびテキスト入力ボックスです。Display 要素には、これ以外にテーブルやセクションのヘッダーがあります。

クラスが指定されていない `Display` 要素は非表示となります。

ボタン

一般的な押しボタンを作成するときは、`<Button>` コンポーネントを使用します。

複数のボタンを水平に並べるときは、`<ButtonRow>` コンポーネントを使用します。

```
<Field>
   <Display class='Button'>
      <Property name='location' value='true'/>
      <Property name='label' value=' キャンセル '/>
      <Property name='command' value='Cancel'/>
   </Display>
</Field>
```

ボタンをボタン行に配置するときは、ボタンの定義に <Property name='location ' value=' button '/> というコードを追加します。この Property フィールドを設定しない場合、ボタンはフォームに記述されている順序で縦に表示されます。

ボタンラベルの割り当てと変更

ボタンを定義するとき、ラベルの値は、次のように label プロパティーの value の設定によって指定されます。

```
<Display class='Button'>
   <Property name='label' value=' キャンセル '/>
```

ブラウザは、このコードに指定されている「**キャンセル**」をボタンのラベルとして表示します。

デフォルトのボタン名の変更

Identity Manager フォームの下部には、通常は 2 つのボタンが表示されます。デフォルトでは、これらのボタンは「**保存**」と「**キャンセル**」という名前が付けられています。これらのボタン名を変更するときは、フォームを次のように変更します。

1. フォーム名を定義する行 (ヘッダー内) で、name フィールドの設定を変更します。

   ```
   <Form name='Anonymous User Menu'>
   ```

   上記の行を次のように変更します。

   ```
   <Form name='Anonymous User Menu' noDefaultButtons=true>
   ```

フォームの下部に次のような「**保存**」ボタンと「**キャンセル**」ボタンのフィールドを追加し、任意のラベル名に変更します。

```
<Field>
   <Display class='Button'>
      <Property name='label' value='Submit'/>
      <Property name='name' value='submitButton'/>
      <Property name='value' value='true'/>
      <Property name='command' value=' 保存 '/>
   </Display>
</Field>
<Field>
   <Display class='Button'>
      <Property name='label' value=' キャンセル '/>
      <Property name='command' value='Cancel'/>
      <Property name='location' value='true'/>
   </Display>
</Field>
```

command パラメータの値とボタン

注 ここで説明する内容は、Button オブジェクトを作成する場合にのみ重要です。XML フォームでコンポーネントを作成するときは、次の表の値が認識されることが前提になると判断できます。

Identity Manager のインタフェースのすべてのページでは、クリックされたフォーム送信ボタンを伝達するメカニズムとして、*command* というデータ送信パラメータが使用されてきました。この条件は、コンポーネントを使用するページプロセッサシステムには適用されませんが、特に Button などの一部のコンポーネントには、*command* パラメータ用の特別サポートが含まれます。

XML フォームを処理するシステムをはじめとする一部のページプロセッサシステムは、*command* パラメータの使用を前提としています。さらに、特定のアクションを示すために、いくつかの command パラメータの値が使用されています。次の表は、このような値について説明しています。

表 2-11 command パラメータに指定できる値

パラメータ	説明
Save	フォームコンテンツを保存すべきであることを示します。
Cancel	フォームコンテンツを破棄すべきであることを示します。
Recalculate	入力データに基づいてフォーム表示を更新すべきであることを示します。

command パラメータでは、どのような値も使用できますが、認識されない値がページの再表示につながることを覚えておいてください。

<ButtonRow> 要素によるボタンの整列

複数のボタンを 1 行に整列させるときは、ButtonRow 要素を使用します。

```
<Field name='OrganizeButtons'>
   <Display class='ButtonRow'>
      <Property name='title' value='Choose a Button'/>
   </Display>
<Field name='ChangePassword'>
   <Display class='Button'>
      <Property name='label' value='Change Password'/>
      <Property name='value' value='Recalculate'/>
   </Display>
```

```
</Field>
<Field name='ResetPassword'>
   <Display class='Button'>
      <Property name='label' value='Reset Password'/>
      <Property name='value' value='Recalculate'/>
   </Display>
</Field>
```

テキストフィールド

フォームには、単一行と複数行の両方のテキスト入力ボックスを含めることができます。単一行のテキスト入力フィールドを作成するときは、<Text> 要素を使用します。複数行のテキスト入力フィールドを作成するときは、<TextArea> 要素を使用します。

```
<Display class='Text'>

   <Property name='title' value='Zip Code'/>

   <Property name='size' value='10'/>

   <Property name='maxLength' value='10'/>

   <Property name='required' value='true'/>

</Display>
```

フィールドラベルの割り当てと変更

テキストフィールドまたはエリアを定義するとき、ラベルの値は、次のように label プロパティーの value プロパティーで設定されます。

```
<Display class='Text'>

   <Property name='label' value='入力'/>
```

ブラウザは、このコードに指定されている「入力」をテキスト入力フィールドのラベルとして表示します。

コンテナ

一部の Display 要素は、コンテナコンポーネントというコンポーネント内に配置されます。コンテナコンポーネントには、次のような用途があります。

- 複数のコンポーネントをまとめ、特定の方法で表示します。単純なコンポーネントは、コンポーネントを縦または横に並べることができます。より複雑なコンテナは、より複雑にコンポーネントを配置でき、コンポーネントの周囲に装飾を加えることもできます。
- フォームで非表示または無効にするコンポーネントをグループ化します。

コンテナクラスを作成すると、通常は HTML table タグが生成されます。

次の表は、代表的なコンテナコンポーネントについて説明しています。

表 2-12　代表的なコンテナコンポーネント

コンポーネント	説明
<SimpleTable>	コンポーネントをグリッドに配置します。オプションとして、上部に列のタイトル行も設定できます。
<ButtonRow>	ボタンを横方向で 1 行に配置します。このコンポーネントは、基本的には、水平レイアウトが事前に設定されたパネルです。
<BorderedPanel>	コンポーネントを、東西南北と中央の 5 つの領域に配置します。
<SortingTable>	ソートに対応した列を持つ、行の背景色が交互に青とベージュで示される表を表示します。

簡単な表の作成

Identity Manager フォームのコンテナコンポーネントでは、<SimpleTable> というコンポーネントが頻繁に使用されます。これは、コンポーネントをグリッドに配置します。オプションとして、上部に列のタイトル行も設定できます。この表示コンポーネントの唯一のプロパティーは、列にタイトルを割り当て、プロパティー文字列のリストで表の幅を定義する columns です。

次の例は、複数のサブフィールドの配置に SimpleTable を使用するフィールドを示しています。

コード例 2-23

```
<Field name='SelectionTable'>
   <Display class='SimpleTable'>
      <Property name='columns'>
         <List>
             <String>Account</String>
             <String>Description</String>
         </List>
      </Property>
   <Field>
      <Display class='Label'>
      <Property name='text' value='Primary Windows 2000 Server'/>
      </Display>
   </Field>

</Field>
```

コード例 2-23

```
   </Display>
   <Field name='accounts[LDAP].selected'>
      <Display class='Checkbox'>
         <Property name='label' value='LDAP'/>
      </Display>

   </Field>
   <Field>
      <Display class='Label'>
         <Property name='text' value='Primary Corporate LDAP Server'/>
      </Display>
   </Field>
   <Field name='accounts[W2K].selected'>
      <Display class='Checkbox'>
         <Property name='label' value='Windows 2000'/>
      </Display>
   </Field>
   <Field>
      <Display class='Label'>
      <Property name='text' value='Primary Windows 2000 Server'/>
      </Display>
   </Field>

</Field>
```

コンポーネントのグループ化

フォームの複数のコンポーネントをグループ化して非表示または無効にするときは、<SimpleTable> コンテナを次の例のように使用します。

コード例 2-24　　フォームのコンポーネントのグループ化

```
<Field>
   <Disable>
      <not>
         <contains>
             <ref>accountInfo.typeNames</ref>
             <s>Windows Active Directory</s>
         </contains>
      </not>
   </Disable>
      <Field name='accounts[AD].HomeDirectory'>
         <Display class='Text'>
            <Property name='title' value='Home Directory'>
         </Display>
      </Field>
</Field>
```

リストの操作

リストの作成に使用するコンポーネントは、リストの長さと、ユーザーが複数のオプションを同時に選択できるかどうかによって異なります。

多くのテキストボックスには、選択可能な複数のオプションを示すリストを備えています。これらのリストのデータを取り込むときは、`allowedValues` というプロパティーの内部でリストを指定して選択するか、リソースのメソッド呼び出し (FormUtil クラスメソッド) を利用して動的に取得します。リストのテキストエリアへの情報の取り込みについては、この章の「リストデータの取り込み」の節を参照してください。

次の表は、代表的なリストタイプと、その作成に使用される HTML Display コンポーネントについて説明しています。

表 2-13 代表的なリストタイプと関連する Display コンポーネント

リストタイプ	HTML コンポーネント
true と false のように、互いに排他的な値を持つオプションリスト	`<CheckBox>` 「チェックボックスの作成」の節を参照してください。
ユーザーが 1 つのオプションのみを選択できる複数オプションのリスト	`<RadioButton>` 「ラジオボタンの作成」の節を参照してください。
ユーザーが 1 つのオプションのみを選択できる複数オプションのリスト (オプション数が多いリスト)	`<Select>` 「単一選択リストの作成」の節を参照してください。
複数のオプションを同時に選択できる複数オプションのリスト	`<MultiSelect>` 「複数選択リストの作成」の節を参照してください。

チェックボックスの作成

チェックボックスの表示には、`<Checkbox>` コンポーネントを使用します。チェックボックスを選択すると、そのボックスの値は `true` となります。選択していないボックスの値は false です。`label` プロパティーの値を編集することで、チェックボックス名を変更できます。

例 1

```
<Field name='accounts[LDAP].selected'>
```

```
   <Display class='Checkbox'>
      <Property name='label' value='LDAP'/>
   </Display>
</Field>
```

例 2

```
<Field name='global.Password.Expired'>
   <Display class='CheckBox'>
      <Property name='title' value='User must change password at
      next login'/>
      <Property name='alignment' value='left'/>
   </Display>
</Field>
```

ラジオボタンの作成

1 つまたは複数のラジオボタンの水平リストを表示するには、<Radio> コンポーネントを使用します。ユーザーが一度に選択できるラジオボタンは 1 つだけです。コンポーネントの値が NULL の場合、または許可される値のいずれとも一致しない場合は、ボタンは選択されません。

```
<Field name='global.EmployeeType'>
   <Display class='Radio'>
      <Property name='title' value='EmployeeType'/>
      <Property name='labels' value='Employee, Contractor, Temporary, Part Time'/>
      <Property name='required' value='true'/>
   </Display>
</Field>
```

単一選択リストの作成

<MultiSelect> コンポーネント以外に、<Select> コンポーネントも、選択できる項目のリストを作成できます。選択できる値のリストが長くなる場合、ラジオボタンではフォームを占める面積が大きくなります。一方、select リストでは、選択できる値の長いリストからユーザーが値を選択できます。リストが整っている場合は、このリストは先行入力をサポートします。ユーザーが選択できる項目を指定するには、allowedValues プロパティーを使用します。

コード例 2-25

```
<Field name='global.title'>
   <Display class='Select'>
      <Property name='title' value='Title'/>
      <Property name='allowedValues'>
         <List>
          <String>Staff</String>
          <String>Manager</String>
          <String>Director</String>
          <String>VP</String>
         </List>
      </Property>
   </Display>
</Field>
```

複数選択リストの作成

複数のオプションを選択できるリストボックスの表示には、<MultiSelect> コンポーネントを使用します。このテキストボックスは、1 つのボックスに含まれる定義済みの値セットを選択ボックスに移動できる、2 つの部分から構成されるオブジェクトを表示します。リストボックスの値の指定には、allowedValues 要素を使用するか、getResources などのメソッド呼び出しを利用して動的に値を取得します。

<Select> コンポーネント以外に、<MultiSelect> コンポーネントも、選択できる項目リストを動的に作成できます。これらのリストのデータを取り込むときは、allowedValues というプロパティーの内部でリストを指定して選択するか、リソースのメソッド呼び出しを利用して動的に取得します。複数選択入力ボックスに表示するリストの取り込みについては、「リストデータの取り込み」の節を参照してください。

コード例 2-26

```
<Field name='waveset.roles'>
   <Display class='MultiSelect' action='true'>
      <Property name='title' value='Roles'/>
      <Property name='availableTitle' value='Available Roles'/>
      <Property name='selectedTitle' value='Current Roles'/>
      <Property name='allowedValues'>
            <invoke name='getObjectNames' class='com.waveset.ui.FormUtil'>

                <ref>display.session</ref>
                <s>Role</s>
                <ref>waveset.original.roles</ref>
            </invoke>
      </Property>
```

コード例 2-26

```
   </Display>
</Field>
```

選択リストの別の表示値セット

フィールドに実際に割り当てられている値セットとは異なる値セットを表示する選択リストを作成できます。このリストは、暗号化された複数の値の判読可能な名前を表示したり、フォームを国際化するために別の表示言語に対応したりする場合によく使用されます。このようなリストを作成するには、valueMap プロパティーを使用して、実際の値と表示値を関連付けます。次の例を参照してください。

コード例 2-27　　valueMap プロパティーによる選択リストの値の変更

```
<Field name='waveset.organization'>
   <Display class='Select'>
      <Property name='title' value='Add Account'/>
      <Property name='nullLabel' value='Select...'/>
      <Property name='valueMap'>
         <list>
            <s>Top</s>
            <s>Top Level</s>
            <s>Top:OrgB</s>
            <s>Ted's Organization</s>
            <s>Top:OrgC</s>
            <s>Super Secret Org</s>
         </list>
      </Property>
   </Display>
</Field>
```

前述の例では、値のマップが文字列ペアのリストとして指定されます。リスト項目の奇数行の文字列は、このフィールドに割り当てられる実際の値です。偶数行の文字列は、選択リストに表示される値です。たとえば、選択リストから `Ted's Organization` を選択すると、このフィールドの値は `Top:Orgb` となります。

リストデータの取り込み

リストには、ユーザーオブジェクトや外部リソースに格納されている情報から動的に計算されたオプションが取り込まれることがよくあります。このようなリストを作成するには、まず、リストデータを取り込む前に、HTML List コンポーネントを作成します。HTML テキストボックスコンポーネントの使用についての詳細は、「単一選択リストの作成」と「複数選択リストの作成」の節を参照してください。

ここで説明するメソッドを含め、これらのリストデータの取り込みには 2 つの方法があります。

- `allowedValues` プロパティーによるリストデータの取り込み
- `FormUtil` メソッドによる、単一選択リストまたは複数選択リストへの、外部リソースから動的に取得した情報の取り込み

特定のタスクの実行に、XPRESS ではなく XML オブジェクト言語を使用する利点については、「XML オブジェクト言語と XPRESS でのリストの表現」の節を参照してください。

リストへの選択可能値セットの取り込み

フォームで使用されるリストにデータを取り込むためのもっとも一般的な方法は、`allowedValues` プロパティーを使用することです。このプロパティーを使用することで、`<Select>` 要素と `<MultiSelect>` 要素で選択できる値のオプションリストを指定できます。このコンポーネントの値は常にリストであり、その値は通常、文字列です。

```
<Field name='department'>
   <Display class='Select' action='true'>
      <Property name='title' value='Department'/>
      <Property name='allowedValues'>
         <List>
         <String>Accounting</String>
         <String>Human Resources</String>
         <String>Sales</String>
         <String>Engineering</String>
         </List>
      </Property>
   </Display>
</Field>
```

グループの複数選択リストへの動的なデータ取り込み

複数選択リストは、通常、2 つの部分から構成されます。

- リストの左側には、選択できる項目が表示されます。値の定義には allowedValues プロパティーを使用します。このプロパティーには、文字列のリスト、XML オブジェクト文字列のリスト、または Java メソッドの呼び出しによって取得された文字列のリストを指定します。
- リストの右側には、現在選択されている項目が表示されます。これらの値を設定するには、左側の allowedValues リストから 1 つまたは複数の項目を選択し、それを右側の選択リストに移動します。フォームの読み込み時に現在の設定を取得し、それをリストの右側に取り込むこともできます。

グループの複数選択リストの追加

リソースから動的に取得されたデータを、グループの複数選択リストに追加する方法は次のとおりです。

- スキーママップの右側にグループを追加します。複数選択リストを表示するテキストエリアの右側部分に表示される値には、フィールド名によって特定される関連ビュー属性の現在値が取り込まれます。

- 次のテキストを任意のフォームに追加し、必要に応じて Field name、prompt、availabletitle、selectedtitle、およびリソースの name のみを変更してください。

注 次の例の display.session の直前の : (コロン) は、フォームのベースコンテキストを無視し、ワークフローコンテキストのルートからオブジェクトを参照できることを表します。

次の例の display.session の直前の : (コロン) は、フォームのベースコンテキストを無視し、ワークフローコンテキストのルートからオブジェクトを参照できることを表します。

```
<Field name='global.AD Groups'>

  <Display class='MultiSelect' action='true'>

    <Property name='title' value='AD Group Membership'/>

    <Property name='availableTitle' value='Available AD Groups'/>

    <Property name='selectedTitle' value='Selected AD Groups'/>

    <Property name='allowedValues'>

      <invoke class='com.waveset.ui.FormUtil'  name='listResourceObjects'>

        <!-- メソッドがユーザー権限の検証に使用するセッション情報を送信します -->

        <ref>:display.session</ref>

        <!-- リソースオブジェクトタイプ - これはリソースごとに異なりますが、アカウント、グループ、および " 配信リスト " のタイプは共通です -->

        <s>Group</s>

        <! -- 呼び出されるリソースの名前 -->

        <s>AD Resource Name</s>

        <!-- オプションマップ - 一部のリソースにはコンテキストのようなオプションがあり、グループのリストが含まれます。たとえば、Active Directory の場合は、複数のコンテナが含まれます。デフォルトでは、リソースに指定されているコンテナが使用されます。ここに指定する値は、デフォルト値に優先して適用されます。リソースがオプションをサポートしない場合は、値を <null/> にしてください -->

        <Map>

         <MapEntry key='context' value='ou=Austin,ou=Texas,dc=Sun,dc=com'/>

        </Map>

        <!-- cacheList - このリストをリソースオブジェクトのリストキャッシュに取り込むかどうかに応じて、true または false に設定します -->
```

```
        <s>true</s>
      </invoke>
    </Property>
  </Display>
</Field>
```

注 リソースがオプションをサポートしない場合は、options map の値を null にしてください。一部のリソースにはコンテキストのようなオプションがあり、グループのリストが含まれます。たとえば、Active Directory の場合は、複数のコンテナが含まれます。デフォルトでは、リソースに指定されているコンテナが使用されます。ここに指定する値は、デフォルト値に優先して適用されます。

cacheList の値は、このリストをリソースオブジェクトのリストキャッシュに格納するかどうかに応じて、true または false に設定します。これにより、メソッドは 1 回実行され、結果はサーバーに格納されます。

選択リストのテキスト入力フィールドの作成

項目を選択するだけでなく、ユーザーが値を入力できる選択リストが必要となる場合もあります。このようなリストを作成するには、次の例のように 3 つのフィールドを実装します。

- この例は、Other というテキスト文字列と、それに隣接するテキストボックスを持つ選択ボックスを作成します。ユーザーが選択ボックスから「Other」オプションを選択すると、ユーザーがカスタム情報を入力するためのフィールドがページに表示されます。
- ユーザーが所属業務を選択するための、業務リストを定義する変数を作成するには、defvar 要素を実装します。

注 フォームで複数回参照される変数を、規則に取り込むようにしてください。次の例では、選択項目のリストは変数 (この例では titleList) に格納されます。これにより、Derivation 規則はこの変数を検索できるようになります。

次の例には、説明的なテキストが組み込まれています。

```
<defvar name='titleList'>
  <list>
    <s>Manager</s>
    <s>Accountant</s>
    <s>Programmer</s>
    <s>Assistant</s>
    <s>Travel Agent</s>
    <s>Other</s>
  </list>
</defvar>
```

この例の次の部分には、title と otherTitle という 2 つの可視フィールドが含まれます。otherTitle フィールドは、ユーザーが選択リストから「Other」オプションを選択した場合にのみ表示されます。3 番目の非表示フィールドは global.Title です。これは、Title と otherTitle のいずれかから設定されます。

ユーザーが選択するためのメインフィールドは Title フィールドです。リストに必要な項目を見つけられなかった場合、ユーザーは「Other」オプションを選択できます。これは一時的なフィールドであり、ユーザーが「保存」をクリックしたときに、格納されたり、ワークフロープロセスに渡されることはありません。リソースからの値の送信と、値がリストに含まれるかどうかの判断には、Derivation 規則が使用されます。

注 次の例では、フォームフィールドに値が自動的に設定されるように、action が true に設定されています。

コード例 2-28

```
<Field name='Title'>
   <Display class='Select' action='true'>
      <Property name='title' value='Title'/>
      <Property name='allowedValues'>
      <Property name='nullLabel' value='Select ・/>
         <expression>
            <ref>titleList</ref>
         </expression>
      </Property>
   </Display>
   <Derivation>
      <cond>
          <isnull><ref>global.Title</ref></isnull>
         <null/>
      <cond>
         <eq>
            <contains>
                <ref>titleList</ref>
                <ref>global.Title</ref>
            </contains>
             <i>1</i>
         </eq>
          <ref>global.Title</ref>
              <s>Other</s>
      </cond>
      </cond>
   </Derivation>
</Field>
```

Other フィールドは、ユーザーが Title フィールドから「Other」を選択した場合にのみフォームに表示されます。Other フィールドの値は、フォームの読み込み時に設定されます。この値は、Title フィールドと global.title フィールドの値に基づきます。

コード例 2-29

```
<Field name='otherTitle'>
    <Display class='Text'>
      <Property name='title' value='Other Title'/>
      <Property name='rowHold' value='true'/>
      <Property name='noWrap' value='true'/>
      <Property name='size' value='15'/>
      <Property name='maxLength' value='25'/>
    </Display>
    <Disable>
      <neq>
         <ref>Title</ref>
         <s>Other</s>
      </neq>
    </Disable>
    <Derivation>
      <cond>
        <eq>
           <ref>Title</ref>
           <s>Other</s>
        </eq>
         <ref>global.Title</ref>
      </cond>
    </Derivation>
</Field>
```

Field の値は、Title フィールドの値に基づきます。このフィールドの値が Other に設定されると、フィールドの値は otherTitle フィールドの値となります。それ以外の値に設定された場合は、Title フィールドの値となります。

コード例 2-30

```
<Field name='Title'>
   <Expansion>
     <cond>
       <eq>
          <ref>global.fieldTitle</ref>
          <s>Other</s>
       </eq>
        <ref>otherTitle</ref>
        <ref>Title</ref>
     </cond>
   </Expansion>
</Field>
```

フォームに表示する前のリソースアカウントリストのフィルタリング処理

フォームに表示する前に、リソースアカウントのリストをフィルタリングできます。ユーザーに表示されるリストから無効化されたアカウントを除去するデフォルトのフィルタリング動作が維持されるユーザーインタフェースの「Change Password Form」を除き、デフォルトでは、フィルタは適用されません。

この除去フィルタは、Form プロパティーとして定義されます。このフィルタは、1 つまたは複数の属性条件のリストです。これは、指定されたリソースアカウントを表示リストから除外するかどうかを決定するときに評価されます。

この機能をサポートするフォーム

次のフォームは、Form プロパティーによる除去フィルタの指定をサポートします。

Change Password Form (ユーザーインタフェース)

管理者インタフェースのフォーム :

- ユーザーパスワード変更フォーム
- プロビジョニング解除フォーム
- 無効化フォーム
- 有効化フォーム
- Rename Form
- 再プロビジョニングフォーム
- ユーザーパスワードリセットフォーム
- ロック解除フォーム

排除プロパティーの形式

フォームの排除 (Exclude Form) プロパティーの形式は次のとおりです。

```
<Configuration wstype='UserForm' ...
   <Extension>
     <Form noDefaultButtons='true'>
  ...
  <Properties>
```

表示されるアカウントのリストに、無効化されたリソースアカウントを含めるときは、リストから無効化属性条件を削除します。

コード例 2-31

```
</Property>
<Property name='Exclude'>
 <list>
 <new class='com.waveset.object.AttributeCondition'>
 <s>disabled</s>
 <s>equals</s>
 </new>
 </list>
 </Property>
 </Properties>
 ...
</Form>
</Extension>
</Configuration>
```

有効なビュー属性

有効な属性名は、currentResourceAccounts オブジェクトの各インスタンスの、前述の各フォームと関連付けられているビューによって参照される名前です。次の属性は、有効な属性です。

- accountDisplayName (文字列)
- accountId (文字列)
- directlyAssigned (true/false)
- disabled (yes/no)
- exists (yes/no)
- id (文字列)
- lastPasswordUpdate (文字列)
- resource (文字列)
- selected (true/false)
- type (文字列)
- userPwdRequired (yes/no)

例 : リソースアカウントリストからの LDAP リソースタイプの排除

指定されたフォームリストから、直接割り当てられない、LDAP タイプのすべてのリソースアカウントを排除するには、排除プロパティーを次のように設定します。

コード例 2-32

```
<Property name='Exclude'>
 <list>
 <new class='com.waveset.object.AttributeCondition'>
 <s>type</s>
 <s>equals</s>
 <s>LDAP</s>
 <s>LDAP</s>
 </new>
 <new class='com.waveset.object.AttributeCondition'>
 <s>directlyAssigned</s>
 <s>equals</s>
 <s>false</s>
 </new>
 </list>
</Property>
```

allowedValues プロパティー内からの FormUtil メソッドの呼び出し

データベースのような Identity Manager の外部にあるリソースから情報を動的に取得し処理するための FormUtil メソッドは、`allowedValues` プロパティー内からも呼び出すことができます。

この例は、FormUtil メソッドを呼び出して、選択リストにデータを取り込む方法を示しています。次の例では、`allowedValues` プロパティーの内部からメソッドが呼び出されます。`getOrganizationsWithPrefixes` メソッド (または任意の FormUtil メソッド) は、式の内部から呼び出されます。

コード例 2-33

```
<Field name='waveset.organization'>
   <Display class='Select'>
      <Property name='title' value='Organization'/>
      <Property name='autoSelect' value='true'/>
      <Property name='allowedValues'>
            <expression>
            <invoke class='com.waveset.ui.FormUtil'
               name='getOrganizationsWithPrefixes'>
                <ref>:display.session</ref>
            </invoke>
            </expression>
         </Property>
   </Display>
</Field>
```

XPRESS は、リソースまたは ActiveSync アダプタ内から Java メソッドを呼び出す機能もサポートしています。呼び出しの結果として得られるデータは、複数選択リストまたは単一選択リストに取り込むことができます。式の内部からのメソッドの呼び出しについては、「XPRESS 言語」を参照してください。

ラベルフィールドの作成

ラベルは、読み取り専用フィールドの値を表示するときに便利なコンポーネントです。<Label> コンポーネントのプロパティーを使用することで、色、値 (文字列)、フォントスタイルなどの表示特性を定義できます。

```
<Field>
   <Display class='Label'>
      <Property name='text' value='Primary Corporate LDAP
         Server'/>
   </Display>
</Field>
```

value 属性の値は常に文字列です。

その他の Display 要素の操作

これまでに説明した要素以外に、次の Display 要素をフォームに組み込むことができます。

- セクションヘッダー
- カレンダアイコン
- バックリンク

フォームへのセクション見出しの追加

長いフォームを目立つラベルで分割するときは、セクション見出しが便利です。<SectionHead> 要素は、title (prompt) プロパティーの値によって定義される、新しいセクション見出しを表示します。これは Label クラスを拡張したもので、font プロパティーに大きな太字のテキストとなるスタイルを設定します。また、pad プロパティーはゼロに設定され、デフォルトの 2 文字分の隙間は取り除かれます。

```
<Field>
   <Display class='SectionHead'>
      <Property name='title' value ='Calculated Fields'/>
```

```
   </Display>
</Field>
```

フォームへのカレンダアイコンの追加

DatePicker 要素を使用することで、ページにカレンダアイコンを追加できます。ユーザーは、このアイコンをクリックして日付を選択し、ページフィールドに値を適用できます。たとえば、Identity Manager の「監査レポートの作成」ページは、開始日と終了日の選択に、このコンポーネントを使用しています。

DatePicker 要素は、日付オブジェクトを返します。DatePicker 要素を使用して設定するほとんどのリソース属性は、文字列形式の日付を必要とします。追加のテキストフィールドは、新しい日付オブジェクトから文字列への変換、または現在の設定の表示に使用されます。

次の表に示される、さまざまな形式の文字列を渡して dateToString メソッドを呼び出すことで、形式が異なるいずれかの日付を取得できます。

表 2-14 有効期限日の形式

有効期限日のフィールド	形式
AIX	MMddHHmmyy
HPUX	MM/dd/yy
Solaris	MM/dd/yyyy

```
<Field name='aix_account_expire'>
   <Display class='DatePicker'>
     <Property name='title' value='Set Password Expiration Date'/>
  </Display>
</Field>
```

次のフィールドは、/etc/security/user ファイルに指定されている、パスワードの有効期限日を表示します。このフィールドは、新しい日付を選択したあとに表示の更新や再計算を行なった場合、aix_account_expire フィールドによって選択される新しい日付も表示します。Identity Manager は、aix_account_expire 日付フィールドに DatePicker フィールドから NULL 以外の値が設定されたかどうかを確認します。

この日付フィールドが設定されている場合、Identity Manager は invoke メソッドを呼び出して、指定された形式 (MMddHHmmyy) で日付オブジェクトを文字列に変換します。

設定されていない場合は、AIX OS に設定されている現在の日付 (accounts[AIX].aix_expires) が表示されます。

コード例 2-34

```
<Field name='accounts[AIX].aix_expires'>
  <Display class='Text'>
    <Property name='title' value='Current Password Expiration Date'/>
    <Property name='noNewRow' value='true'/>
    <Property name='readOnly' value='true'/>
    <Property name='size' value='10'/>
  </Display>
  <Expansion>
    <cond>
      <notnull>
         <ref>aix_account_expire</ref>
      </notnull>
      <invoke name='dateToString' class='com.waveset.util.Util'>
      <!-- dateToString メソッドの最初の引数は日付オブジェクトです -->
         <ref>aix_account_expire</ref>
      <!-- 2 番目の引数は、変換後の日付 / 文字列の形式です -->
           <s>MMddHHmmyy</s>
      </invoke>
       <ref>accounts[AIX].aix_expires</ref>
    </cond>
  </Expansion>
</Field>
```

バックリンクの追加

ブラウザの「**戻る**」ボタンと同じように動作するコンポーネントを追加できます。このコンポーネントを使用することで、フォーム上の任意の場所にバックリンクを追加できます。

```
<Field name='back'>

   <Display class='BackLink'>

      <Property name='title' value='Back'/>

      <Property name='value' value='previous page'/>

   </Display>

</Field>
```

フォーム上のコンポーネントの位置

フォーム上でのコンポーネントの位置は、次の要因によって決定されます。

- **このフォームに関連付けられている Java Service Pages (JSP):** フォームのタイトルとサブタイトルは、ここに設定できます。
- **フォームにコンポーネントが表示される順序 :** フォームフィールドは、フォームに取り込まれる順序でブラウザに表示されます。
- **コンテナフォームの使用 :** たとえば、縦に並べたボタンを作成するには、`<ButtonRow>` コンテナコンポーネントを使用します。

非表示コンポーネントの使用

多くのフォームはユーザーに表示されず、リソースアダプタを通じて外部リソースから取得したデータを、Identity Manager に渡す前に処理するために使用されます。表示されるフォームでも、一部のコンポーネントは非表示に設定できます。非表示コンポーネントは、この入力データの処理と、判読可能な形式へのデータ変換にも使用されます。

フォーム内の一部の非表示処理は、FormUtil Java クラス内のメソッドによって実行されます。これは、フォーム内のリストの情報を、外部リソースから動的に取得する場合に頻繁に使用されます。

ここでは、データを処理し、フォームで行われるその処理をオプションとして非表示に設定するタスクについて説明します。代表的なタスクには、次のものがあります。

- Derivation 要素と Expansion 要素を使用して、XPRESS ロジックを含める
- メソッドを呼び出して、リストにデータを取り込む
  - DN 文字列を作成する
  - セッション所有者がアクセスできるオブジェクトタイプのリストを取得する
  - 組織のリストを取得する
  - 割り当てられていないリソースのリストを取得する
  - リソースオブジェクト名のリストを取得する
- コンポーネントを無効化する
- コンポーネントを非表示にする

Derivation 要素と Expansion 要素による XPRESS ロジックの追加

一般に、フィールドには `Derivation` 規則または `Expansion` 規則のいずれかが適用されます。フィールドに両方の規則が含まれる場合は、これらのフィールドが相互に競合しないことを確認してください。

XPRESS を使用してフォームフィールドの値を計算するときは、<Expansion> コンポーネントと <Derivation> コンポーネントを実装します。これらの式は、式が評価されるタイミングが異なるだけで、基本的には同じです。多くの場合、Derivation 規則は、フォームの読み込み時にフィールドの値を設定する場合に使用されます。Expansion 規則は、ページを再計算するとき、またはフォームを保存するときにフィールドの値を設定する場合に使用されます。

表 2-15 Derivation 式と Expansion 式

コンポーネント	説明	評価
<Derivation>	このフィールドの値として使用される任意の値を無条件で計算します。Derivation 式が評価されると、フィールドの現在値は常に置き換えられます。	Derivation 規則は、フォームが最初に読み込まれるとき、あるいは、1 つまたは複数のリソースからデータがフェッチされるときに実行されます。
<Expansion>	フィールドの値を無条件で計算します。	Expansion 規則は、ページを再計算するとき、またはフォームを保存するときに実行されます。 ユーザービュー以外のすべてのフォームでは、Expansion 規則は、ページを再計算するとき、またはフォームを保存するときに実行されます。ユーザービューでは、ユーザーフォームを最初に読み込むときにも <Expansion> タグが実行されます。
<Validation>	フォームに入力された値が有効であるかどうかを検証します。	Validation 規則は、フォームを送信するたびに評価されます。

<Derivation> ステートメントの例

次の 2 つの例は、Derivation 要素の使用例を示しています。

- 例 1: グローバルフィールドの信頼できるソースの指定
- 例 2: 値セットと別の値セットのマッピング

例 1:

最初の値が定義されている場合、次の例はその値を使用します。最初の値が定義されていない場合は、2 番目の値が使用されます。

```
<Derivation>
   <or>
```

```
            <ref>accounts[AD].fullname</ref>
            <ref>accounts[LDAP].fullname</ref>
         </or>
      </Derivation>
```

例 2:

次の <Derivation> 要素の使用例は、条件ロジックを使用して、値のセットを別のセットにマップするフィールド定義を示します。

この例では、リソースアカウント属性 accounts[Oracle].locCode が、まず、case 要素 AUS に対して評価されます。結果が true となる場合は、返される値が 2 番目の値となり、location フィールドに表示されます。いずれの case 要素とも一致しない場合は、case 式のデフォルト値が返されます。case 式に含まれる最初の式で一致する略号が見つかった場合は、case 式に含まれる 2 番目の式の値が、switch 式の結果として返されます。

コード例 2-35

```
<Field name='location' prompt='Location'>
   <Display class='Text'/>
      <Derivation>
      <switch>
            <ref>accounts[Oracle].locCode</ref>
            <case>
               <s>AUS</s>
               <s>Austin</s>
            </case>
            <case>
               <s>HOU</s>
               <s>Houston</s>
            </case>
            <case>
               <s>DAL</s>
               <s>Dallas</s>
            </case>
            <case default='true'>
               <s>unknown</s>
            </case>
      </switch>
      </Derivation>
</Field>
```

<Expansion> ステートメントの例

次の 2 つの例は、Expansion 要素の使用例を示しています。

- 例 1: フィールドに入力されるテキストの大文字と小文字を標準化する規則の実装

- 例 2: Expansion ロジックを非表示にする

例 1:

Expansion 規則は、フィールドに入力された情報を、リソース側で前提とされる形式、または規則によって指定される形式と一致する値に変換します。たとえば、ユーザーが名前を入力する自由形式のテキストボックスに Expansion 規則を含め、頭文字を大文字、残りの文字を小文字にそろえることができます。

フィールドでグローバル変数を使用すると、フォームの保存時に、この値を持つすべてのリソースが設定されます。このフォームを読み込むと、Identity Manager は各リソースから値を取り込みます (フィールドが無効化されている場合を除く)。最後のリソースの読み込みによって、フォームの値が設定されます。ユーザーがローカルに変更を加えた場合は、その変更は表示されない可能性があります。このため、フィールドの信頼できるソースとして 1 つまたは複数のリソースを指定する Derivation 規則を使用することで、正しい値を確実に属性に適用することができます。

コード例 2-36

```
<Field name='global.lastname'>
   <Display class='Text'>
       <Property name='title' value='Last Name'/>
       <Property name='size' value='32'/>
       <Property name='maxLength' value='128'/>
       <Property name='noNewRow' value='true'/>
       <Property name='required'>
         <Boolean>false</Boolean>
       </Property>
   </Display>
        <Expansion>
          <block>
            <defvar name='lname'>
                  <ref>global.lastname</ref>
               </length>
               <s>1</s>
             </sub>
            </defvar>
            <concat>
              <substr>
                <upcase>
                  <ref>global.lastname</ref>
                </upcase>
                <s>0</s>
                <s>1</s>
              </substr>
              <substr>
                <ref>lname</ref>
                <s>1</s>
                <ref>nlength</ref>
              </substr>
            </concat>
          </block>
</Field>
```

コード例 2-36

```
              <downcase>
                <ref>global.lastname</ref>
              </downcase>
            </defvar>
            <defvar name='nlength'>
              <sub>
                <length>
                  <ref>global.lastname</ref>
                </length>
                <s>1</s>
              </sub>
            </defvar>
            <concat>
              <substr>
                <upcase>
                  <ref>global.lastname</ref>
                </upcase>
                <s>0</s>
                <s>1</s>
              </substr>
              <substr>
                <ref>lname</ref>
                <s>1</s>
                <ref>nlength</ref>
              </substr>
            </concat>
          </block>
</Field>
```

この XPRESS ロジックが複数のフィールドに実装される場合は、それを規則に含めることを検討してください。

例 2:

次の例には Display クラス定義が含まれないため、このフィールドは非表示にも設定されます。Display クラス定義を取り除くことで、フィールドはフォームに表示されなくなります。ただし、フィールドは依然としてフォームのアクティブ部分と見なされ、<Expansion> 式を利用してリソース属性の値を生成します。

```
<Field name='accounts[Oracle].locCode'>
   <Expansion>
      <switch>
         <ref>location</ref>
            <case>
               <s>Dallas</s>
               <s>DAL</s>
            </case>
      </switch>
   </Expansion>
</Field>
```

```
<Field name='accounts[Oracle].locCode'>
            <case>
               <s>Austin</s>
               <s>AUS</s>
            </case>
            <case>
               <s>Houston</s>
               <s>HOU</s>
            </case>
            <case>
               <s>Dallas</s>
               <s>DAL</s>
            </case>
      </switch>
   </Expansion>
</Field>
```

この例では、location フィールドによるマッピングとは逆のマッピングが行われます。

<Validation> ステートメントの例

Validation 式を使用することで、フォームに入力した値が有効であるかどうかを判断するためのロジックを指定できます。

Validation 式は、成功を示す場合は NULL を返し、失敗を示す場合は判読可能なエラーメッセージを含む文字列を返します。Validation 式のエラーメッセージは、フォーム最上部に赤のテキストで表示されます。

次の例には、ユーザーがフィールドに入力した年齢の値が、0 より大きいことを確認するロジックが含まれます。入力された年齢が 0 より大きいか、または 0 である場合、この式は NULL を返します。

```
<Field name='age'>
   <Validation>
     <cond>
       <lt>
         <ref>age</ref>
         <i>0</i>
       </lt>
       <s>Age may not be less than zero.</s>
     </cond>
   </Validation>
</Field>
```

メソッドを呼び出して、リストにデータを取り込む

単一選択リストと複数選択リストのテキストボックスには、外部リソースから取得した情報が取り込まれることがよくあります。Sun でサポートされるいずれかの FormUtil メソッドを呼び出すことで、この情報を動的にリストに取り込むことができます。これらの一般的なメソッドは、次のタスクを実行できます。

- リソースオブジェクト名のリストを取得する
- マップオプションを持たないリソースオブジェクトのリストを取得する
- DN 文字列を作成する
- アクセスできるオブジェクトタイプのリストを取得する
- セッション所有者がアクセスできるオブジェクトタイプのリストを取得する
- プレフィックスを持つ組織のリストを取得する
- プレフィックスを持たない組織のリストを取得する
- プレフィックスを持つ組織の表示名のリストを取得する
- ユーザーに割り当てられていないアプリケーションのリストを取得する

`<Select>` コンポーネントと `<MultiSelect>` コンポーネント、および `allowedValues` プロパティーについては、「リストデータの取り込み」の節を参照してください。

リソースオブジェクトの名前について

リソース上の情報を検索または要求し、Identity Manager にインポートするには、Identity Manager がサポートするオブジェクト定義を使用する必要があります。

次の表は、Identity Manager でサポートされるオブジェクトタイプを示しています。

表 2-16

サポートされるオブジェクトタイプ	説明
account	ユーザーアカウント ID のリスト
Administrator_Groups	ユーザーが所属できる管理者グループの名前
Applications	アプリケーションのリスト
Distribution Lists	電子メール配信エイリアスのリスト
Entitlements	PKI 権利書のリスト
group	セキュリティーオブジェクトと配信リストグループオブジェクトのリスト
Group	セキュリティーグループ
Nodes	SP2 ノードのリスト

表 2-16　(続き)

サポートされるオブジェクトタイプ	説明
PostOffices	GroupWise ポストオフィスのリスト
profile	トップシークレットプロファイルのリスト
PROFILE	DBA_PROFILES テーブルからの Oracle プロファイルのリスト
ROLE	DBA_ROLES テーブルからの Oracle ロールのリスト
shell	使用できる UNIX シェルのリスト
Template	NDS テンプレートのリスト
USERS	DBA_USERS テーブルからの Oracle プロファイルのリスト
UnassignedTokens	使用できる未割り当てトークンのリスト
User_Properties	ユーザープロパティー定義 (ユーザーに設定できるプロパティー) のリスト

リソースオブジェクト名のリストの取得

使用している特定のリソースに定義されたオブジェクト名のリストを取得するときは、`listResourceObjects` メソッドを使用します。マップオプションを持つ、または持たないリストを取得できます。マップオプションは、完全なリストを返す代わりに、返される値をフィルタリングして 1 つのコンテナに集約できるディレクトリ構造を持つリソースのみで使用されます。

サーバーのキャッシュからではなく、必ずリソースからリソースオブジェクトリストを取得するには、まず、`clearResourceObjectListCache()` メソッドを呼び出すか、*cacheList* 引数を false に設定します。ただし、大規模なリストでは、キャッシュを使用するほうがパフォーマンスが向上します。リソースへのアクセスは 1 回のみで、結果はキャッシュに格納されます。このため、キャッシュを利用することをお勧めします。

また、オブジェクトリストの要求対象となるリソースを指定する、キーと文字列値の 1 つまたは複数のペアのセットを指定することもできます。

次の表は、各リソースでサポートされるオブジェクトタイプを示しています。

表 2-17　サポートされるオブジェクトタイプ

リソース	サポートされるオブジェクトタイプ
AIX	account、Group
ACF2	account
ClearTrust	account、Group、group、Administrator_Groups、Applications、Entitlements、User_Properties
Entrust	Group、Role
GroupWise	account、Distribution Lists、PostOffices
HP-UX	account、Group、shell
LDAP	account、Group
Oracle	USERS、ROLE、PROFILE
NDS	account、Group
PeopleSoft	account
RACF	account、Group
SAP	account、table、profiles、activitygroups
SecurID	UnassignedTokens
SP2	Nodes
Solaris	account、Group、shell
TopSecret	account
VMS	account
Windows Active Directory	account、Group Active Directory のどの有効オブジェクトクラス名も、オブジェクトタイプとして指定できます。オブジェクトクラス名のリストは Active Directory スキーマドキュメントに記載されています。返されるリストには、オブジェクトの識別名が含まれます。デフォルトでは、メソッドは、Container リソース属性によって指定されるコンテナを検索します。ただし、`listResourceObjects` の呼び出し時のオプションとして、コンテナを指定することができます。その名前は、コンテナの識別名としてください。この場合、そのコンテナに含まれるオブジェクトの名前のみがリストに取得されます。

マップオプションを持たないリソースオブジェクトのリストの取得

マップオプションを持たないリソースオブジェクトのリストを取得するには、リソースオブジェクトのタイプとリソース名を指定します。一部のリソースは、リストのサブセットでの機能をサポートすることに注意してください。これは、開始ディレクトリを指定することで実行できます。

次の例の詳細は次のとおりです。

- `<UnassignedTokens>` という文字列は、取得するリソースのオブジェクトタイプを表します。一般的なリソースオブジェクトタイプには、これ以外にも groups、distribution lists、および accounts があります。
- `<SecurID>` という文字列は、オブジェクトタイプの取得元リソースを表します。
- `null` という値は、マップオプションを含めないことを表します。
- `true` という値は、結果をキャッシュに格納することをサーバーに伝えます。

```
<invoke name='listResourceObjects'
   class='com.waveset.ui.FormUtil'>
      <ref>:display.session</ref>
      <s>UnassignedTokens</s>
      <s>SecurID</s>
      <null/>
      <s>false</s>
</invoke>
```

マップオプションを持つリソースオブジェクトのリストの取得

マップオプションを持つリソースオブジェクトのリストを取得するには、リソースオブジェクトのタイプ、リソース名、および検索を開始するディレクトリを表すマップオプションを指定します。リソースは、ディレクトリベースで指定してください。

たとえば、Software Access ディレクトリ内のすべての Active Directory グループのリストを取得するには、指定したディレクトリパス (`ou=Software Access, dc=mydomain, dc=com`) で検索を行うマップオプションを作成します。

例 :

次の例の詳細は次のとおりです。

- `Group` という文字列は、取得するリソースのオブジェクトタイプを表します。リソースオブジェクトのタイプを表す文字列は、Available Resource Object Types というテーブルに指定されます。
- `AD` という文字列は、オブジェクトタイプの取得元リソース名を表します。マップオプションは、リストの取得元ディレクトリを指定します。

- true という値は、結果をキャッシュに格納することをサーバーに伝えます。
- false という値は、結果をキャッシュに格納しないことをサーバーに伝えます。

```
<invoke name='listResourceObjects'
      class='com.waveset.ui.FormUtil'>
         <ref>:display.session</ref>
            <s>Group</s>
            <s>AD</s>
         <Map>
            // この設定により、指定したコンテナ / 組織の単位、
          およびその下部に含まれるグループのみのリストを
          取得できます
         <MapEntry key='container'
         value='LDAP://hostX.domainX.com/cn=Users,dc=domainX,dc=com'/>
         </Map>
            <s>false</s>
</invoke>
```

DN 文字列の作成

指定したユーザー ID とベースコンテキストから、識別名のリスト、または単一の識別名を動的に作成できます。このメソッドでは、リストは返されません。通常は、Expansion 規則で使用されます。

DN 文字列のリストの動的な作成

ユーザー ID とベースコンテキストを指定して、DN 文字列のリストを動的に作成することができます。

次の例は、ユーザー ID とベースコンテキストを使用して DN 文字列のリストを動的に作成する方法を示しています。

次のコードでは、まず、ユーザーに追加するベースコンテキストが定義されます。

```
<Field name='baseMemberContextContractor'>
   <Default>
      <s>ou=Contractors,dc=example,dc=com</s>
   </Default>
</Field>

<Field name='baseMemberContextEmployee'>

   <Default>
      <s>ou=Employees,dc=example,dc=com</s>
   </Default>
</Field>
```

このフォームのユーザーは、次のフィールドにデータを入力します。多くの場合、これは、動的に生成したユーザー ID のリストの格納場所を表します。

```
<Field name='userIds'>
   <Display class='TextArea'>
      <Property name='title' value='UserIds'/>
   </Display>
</Field>
```

次の非表示フィールドには、値を計算するロジックが含まれます。

```
<Field name='Members'>
   <Expansion>
      <switch>
      // ユーザーに割り当てられているロールを参照します
      <ref>waveset.role</ref>
      <case>
      // ユーザーに "Contractor ロール" が割り当てられている場合に、次のような DN を生成します:
      // ex: CN=jsmith,ou=Contractors,dc=example,dc=com
         <s>Contractor Role</s>
         <invoke name='buildDns' class='com.waveset.ui.FormUtil'>
            <ref>userId</ref>
            <ref>baseMemberContextContractor</ref>
         </invoke>
      </case>
      <case>
// また、ユーザーに "Employee ロール" が割り当てられている場合は、次のような DN を生成します:
         // ex: CN=jdoe,ou=Employees,dc=example,dc=com
         s>Employee Role</s>
         <invoke name='buildDns' class='com.waveset.ui.FormUtil'>
            <ref>userId</ref>
            <ref>baseMemberContextEmployee</ref>
         </invoke>
         </case>
      </switch>
   </Expansion>
</Field>
```

単一 DN 文字列の作成

1 つの DN を指定して、リストまたはテキストエリアにデータを取り込む buildDn メソッドを呼び出すことができます。例:

```
<invoke name='buildDn' class='com.waveset.ui.FormUtil'>
   <s>jdoe</s>
```

```
   <s>dc=example,dc=com</s>
</invoke>
```

この例は、CN=jdoe,dc=example,dc=com という文字列を返します。

未割り当てリソースのリストの取得

指定したユーザー ID には表示する権限はあるが、現時点では割り当てられていないすべてのリソースのリストを取得するときは、getUnassignedResources メソッドを使用します。

<ref> ステートメントは、指定したユーザーに関する情報を持つビュー属性を表します。例 :

```
<invoke name='getUnassignedResources'
class='com.waveset.ui.FormUtil'>
   <ref>:display.session</ref>
   <ref>waveset.role</ref>
   <ref>waveset.original.resources</ref>
</invoke>
```

アクセス可能なオブジェクトタイプのリストの取得

セッション所有者が現時点でアクセスできるオブジェクトタイプのリストを取得するときは、getObjectNames メソッドを使用します。

検索できるオブジェクトタイプは次のとおりです。

- Account
- Administrator
- Configuration
- EmailTemplate
- Resource
- Role
- System
- TaskInstance
- User
- UserForm

すべてのオブジェクトタイプのリストは、「デバッグ」ページの「List Objects」オプションで表示できます。例 :

```
<invoke name='getObjectNames' class='com.waveset.ui.FormUtil'>
   <ref>:display.session</ref>
   <s>UserForm</s>
</invoke>
```

セッション所有者がアクセスできるオブジェクトタイプのリストの取得

セッション所有者がアクセスできるオブジェクト名のリストを取得するときは、getObjectNames メソッドを使用します。例:

```
<invoke name='getObjectNames' class='com.waveset.ui.FormUtil'>
  <ref>:display.session</ref>
</invoke>
```

プレフィックスを持つ組織のリストの取得

プレフィックス (たとえば、TOP, TOP:IT, TOP:HR) を持つ組織のリストを取得するときは、getOrganizationsWithPrefixes メソッドを使用します。例:

```
<invoke name='getOrganizationsWithPrefixes'
 class='com.waveset.ui.FormUtil'>
   <ref>:display.session</ref>
</invoke>
```

プレフィックスを持たない組織のリストの取得

プレフィックス (たとえば、TOP, TOP, TOP) を持たない組織のリストを取得するときは、getOrganizations メソッドを使用します。例:

```
<invoke name='getOrganizations' class='com.waveset.ui.FormUtil'>
   <ref>:display.session</ref>
</invoke>
```

プレフィックスを持つ組織表示名のリストの取得

プレフィックスを持つ組織表示名のリストを取得するときは、getOrganizationsDisplayNamesWithPrefixes メソッドを使用します。

```
<invoke name='getOrganizationsDisplayNamesWithPrefixes'
 class='com.waveset.ui.FormUtil'>
   <ref>:display.session</ref>
</invoke>
```

ユーザーに割り当てられていないアプリケーションのリストの取得

現時点でユーザーに割り当てられていないアプリケーションのリストを取得するときは、getUnassignedApplication メソッドを使用します。例:

```
<invoke name='getUnassignedApplications' class='com.waveset.ui.FormUtil'>
   <ref>:display.session</ref>
   <ref>waveset.roles</ref>
   <ref>waveset.original.applications</ref>
</invoke>
```

ハッシュマップの作成

listResourceObjects メソッドと callResourceMethods メソッドは、ハッシュマップを受け付けます。ハッシュマップは、<Map> 要素を使用して作成できます。

次の例の <Map> 要素は、変更されない制的なマップを作成します。

```
  <Map>
     <MapEntry name='key1' value='value1'/>
     <MapEntry name='key2' value='value2'/>
  </Map>
```

<map> 要素を使用して、XPRESS 式でマップを作成することもできます。<map> 要素は、コンテンツが別の式によって定義されるマップを動的に作成することができます。

XPRESS 言語によるマップの作成については、「XPRESS 言語」を参照してください。

フィールドの無効化

フィールドを無効にすると、フィールドと、その入れ子フィールドはページに表示されなくなります。また、値の式も評価されません。無効化されたフィールドの値がすでにビューに含まれる場合、その値は変更されません。

```
<Disable></Disable>
```

注 global.* 属性は、有効なフィールドのみから取得されます。フォームがフィールドを非表示にするのではなく、動的に無効にする場合は、global.* 属性を使用して、そのフィールドの値を取得することはできません。

例:

```
<Disable>
```

```
    <eq><ref>userExists</ref><s>true</s></eq>
</Disable>
```

注	Disable 式は、その他の式よりも頻繁に評価されます。このため、Disable 式は、比較的単純に記述してください。データベースのルックアップのように、負荷の大きい処理を実行する Java クラスは呼び出さないようにしてください。

フィールドの非表示

フィールドを非表示に設定すると、フィールドと、その入れ子フィールドはページに表示されなくなります。ただし、フォームの処理によって、フィールドの値は設定されます。

フィールドを非表示にするには、表示しない特定のフィールドの Display プロパティーを定義しません。これは条件に依存しません。

```
<Field name='field A'/>
```

値の計算

フォーム内での動的な値の計算には、次の方法があります。

- フィールド値の生成
- フォーム内での規則の使用
- フォーム内での XPRESS ステートメントの使用

フィールド値の生成

一部のフォームでは、最初は抽象的な導出フィールドのセットをユーザーに表示し、その後のフォームの送信時に、リソースアカウントの実際の属性値のさまざまなセットを生成することが必要となる場合があります。これは、フォームの展開と呼ばれます。展開されたフィールドは、よく導出フィールドと組み合わせて使用されます。

フォーム内での規則の使用

フォームでは、通常、allowedValues 表示プロパティーを計算する規則を呼び出したり、<Disable> 式の中でフィールドの可視性を制御したりします。フォーム内で次の項目を格納および再利用するもっとも効率的なメカニズムは、規則の使用であると考えられます。

- 企業の部署名のリスト
- デフォルト値

- オフィスビルのリスト

規則の詳細については、『Identity Manager 配備ツール』の「規則」の章を参照してください。

XPRESS ステートメントの使用

XPRESS 言語は、式とスクリプトを記述するための XML ベースの言語です。この言語で記述されたステートメントは式と呼ばれ、フォームにデータ変換機能を追加したり、ワークフローやフォームなどの Identity Manager オブジェクトに状態遷移ロジックを組み込んだりするために、Identity Manager 全体で使用されます。

XPRESS は、XML に基づいた構文を使用する関数言語です。この言語で記述されるすべてのステートメントは、0 個以上の引数をとる関数を呼び出し、値を返します。製品には関数が組み込まれていますが、新しい関数を定義することもできます。
XPRESS は、任意の Java クラスでのメソッドの呼び出しと、式に含まれる Javascript の評価もサポートします。

XPRESS の機能の詳細については、「XPRESS 言語」を参照してください。

XPRESS を使用する理由

式は、主に次の Identity Manager タスクに使用されます。

- **エンドユーザーと管理者のフォームのカスタマイズ** : フォームは、フィールドの可視性の制御と、表示および保存できる形式へのデータの変換に XPRESS を使用します。
- **ワークフローでのコントロールフローの定義** : ワークフローは、ワークフロープロセスの実行手順の順序を決定する遷移条件の定義に XPRESS を使用します。
- **ワークフローアクションの実装** : ワークフローアクションは、XPRESS を使用して実装できます。アクションの式は、単純な計算を実行するだけでなく、Java クラスまたは JavaScript を呼び出して、複雑な処理を実行することもできます。

これらの要素に含まれる式は、Identity Manager 全体で使用できます。

式の例

次の例の `<add>` 要素は、`add` という XPRESS 関数の呼び出しを表します。

```
<add> <ref>counter</ref> <i>10</i> </add>
```

この関数には、2 つの引数が渡されます。

- 最初の引数 - `ref` という関数の呼び出しによって定義される値。`ref` 関数の引数は、変数名のリテラル文字列です。`ref` 関数から返される値は、*counter* という変数の現在値です。
- 2 番目の引数 -- `i` という関数の呼び出しによって定義される値。`i` 関数の引数は、整数のリテラル文字列です。`i` 関数から返される値は、10 という整数です。

add 関数から返される値は、*counter* 変数の現在値に整数 10 を加えた結果となります。すべての関数呼び出しは、次の処理で使用される値を返します。たとえば、ref の呼び出しによってカウンタの値が得られると、<i> 呼び出しは整数 10 を返し、<add> 呼び出しは 2 つの呼び出しの結果を合計します。

フォームに組み込まれた式の例

次の例は、Identity Manager フォームに組み込まれた XPRESS ロジックの使用例を示しています。XPRESS は、ブラウザに表示するロール関連の選択項目を生成するいずれかの FormUtil Java メソッドの呼び出しに使用されています。式全体が <expression> というタグに囲まれていることに注目してください。

```
<Field name='waveset.role'>
   <Display class='Select' action='true'>
      <Property name='title' value='Role'/>
      <Property name='nullLabel' value='None'/>
      <Property name='allowedValues'>
         <expression>
            <invoke class='com.waveset.ui.FormUtil' name='getRoles'>
            <ref>:display.session</ref>
        <ref>waveset.original.role</ref>
            </invoke>
         </expression>
      </Property>
   </Display>
 </Field>
```

ユーザー編集フォーム

Identity Manager では、リソースのスキーママップ内でその属性が必須かどうかが画面上でわかるようになっています。ユーザー編集フォームでは、これらの属性を * (アスタリスク) で示します。デフォルトの Identity Manager では、属性名に続くテキストフィールドの後ろにこのアスタリスクが表示されます。

アスタリスクの位置をカスタマイズする手順は次のとおりです。

1. Identity Manager IDE または任意の XML エディタを使用して、コンポーネントプロパティー設定オブジェクトを開きます。
2. EditForm.defaultRequiredAnnotationLocation=left を <SimpleProperties> タグに追加します。

   defaultRequiredAnnotationLocation で有効な値は、left、right、および none です。
3. 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。

フォームへのガイダンスヘルプの追加

Identity Manager には、次の 2 種類のオンラインヘルプが用意されています。

- タスク関連の情報を示すヘルプ。Identity Manager のマストヘッドにあるボタンからアクセスできます。このヘルプを編集することはできません。
- ガイダンス (ポップアップヘルプ)。フィールドレベルのヘルプです。フィールドまたはエリアの左側に表示されるガイダンスアイコン () からアクセスできます。

コンポーネントにガイダンスヘルプを指定する方法

ガイダンスヘルプのテキストは、どのコンポーネントにも関連付けることができます。ただし、現時点では、表示に `EditForm` コンテナが必要です。ガイダンステキストを指定するには、コンポーネントの `title` プロパティーとヘルプカタログ内のエントリを一致させることで、コンポーネントに割り当てます。「コンポーネントの title プロパティーとヘルプエントリの一致」の節を参照してください。

コンポーネントの title プロパティーとヘルプエントリの一致

コンポーネントタイトルにサフィックス _HELP が付加されたものをヘルプカタログ内のキーの値として使用することで、カタログエントリをコンポーネントに自動的に関連付けることができます。たとえば _FM_DELEGATEWORKITEMSFORM_SELECT_WORKITEM_TYPE キーのヘルプカタログエントリは、_FM_DELEGATEWORKITEMSFORM_SELECT_WORKITEM_TYPE_HELP です。

XML フォームの使用時は、`Property` 要素を使用して、コンポーネントのタイトルを明示的に指定できます。このプロパティーが設定されていない場合は、`Field` 要素に含まれる `prompt` 属性の値が適用されます。

ガイダンスヘルプの優先指定

カスタムメッセージカタログを使用して、ポップアップウィンドウに表示されるガイダンステキストを優先指定できます。カスタムメッセージカタログに defaultCustomCatalog という名前を付けると、Identity Manager はこれを認識し、このカタログを自動的に使用します。また、別のカタログ名を付け、customMessageCatalog という属性名のあとに、System Configuration オブジェクトの値としてそのカタログ名を指定することもできます。

たとえば、次のようにします。

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

次の例では、`_FM_DELEGATEWORKITEMSFORM_SELECT_WORKITEM_TYPE` というコンポーネントのカスタムガイダンスヘルプが設定されます。

```
<Waveset>
   <Configuration name="sampleCustomCatalog">
      <Extension>
         <CustomCatalog id="defaultCustomCatalog" enabled="true">
            <MessageSet language="en" country="US">
            <Msg id="_FM_DELEGATEWORKITEMSFORM_SELECT_WORKITEM_TYPE">作業項目のタイプ
を選択してください</Msg>
            <Msg id="_FM_DELEGATEWORKITEMSFORM_SELECT_WORKITEM_TYPE_HELP>作業項目のタ
イプ：作業項目のタイプをリストから選択してください。</Msg>
            </MessageSet>
         </CustomCatalog>
      </Extension>
   </Configuration>
</Waveset>
```

フォームに関連するその他のタスク

フォーム関連では、ほかに次のようなタスクがあります。

- FormUtil メソッドの呼び出し
- フォームへの Javascript の挿入
- ユーザーまたはオブジェクトの存在確認
- XPRESS フォームへのアラートメッセージの挿入

FormUtil メソッドの呼び出し

FormUtil クラスは、フォームオブジェクトを使用して XPRESS 式から呼び出せるユーティリティーメソッドの集合です。これは、許可される値のリストへの取り込みと、入力の検証に使用できます。一般に、FormUtil メソッドは、リストまたはフィールドの許可される値の定義を支援するために呼び出されます。

```
<invoke class='com.waveset.ui.FormUtil'

   name = 'listResourceObjects'>

</invoke>
```

この name フィールドは、メソッドの名前を表します。

フォーム内でのこれらのメソッドの使用例については、「非表示コンポーネントの使用」の節を参照してください。

フォームへの JavaScript の挿入

事前にフォーマットされた Javascript をフォームに挿入するときは、<script> コンポーネントを次のように使用します。

```
<Field>
  <Expansion>
    <script>
     ............
```

オブジェクトまたはユーザーの存在確認

アクションを実行する前に、オブジェクトの存在を確認できると便利かもしれません。たとえば、新規ユーザーを作成したり、フィールドに入力したマネージャー名が有効であることを確認したりする前に、ユーザー名が Identity Manager に存在するかどうかを確認できます。

オブジェクトの存在を確認するときは、testObject メソッドを使用します。このメソッドの使用時にオブジェクトタイプを指定するときは、161 ページの「アクセス可能なオブジェクトタイプのリストの取得」の節に記載されているオブジェクトタイプを使用してください。次の例では、<s>User</s> というユーザータイプが指定されています。2 番目の文字列は、オブジェクトタイプの値 (この例では jdoe) を指定します。

例 :

```
<invoke name='testObject' class='com.waveset.ui.FormUtil'>
   <ref>:display.session</ref>
   <s>User:</s>
   <s>jdoe</s>
</invoke>
```

オブジェクトが検出されると、testObject メソッドは true を返します。検出に失敗した場合は、NULL が返されます。

ユーザーの存在を確認するときは、testUser メソッドを使用します。検索するユーザーオブジェクトの名前は、<s> 要素で指定されます。例 :

```
<invoke name='testUser' class='com.waveset.ui.FormUtil'>
   <ref>:display.session</ref>
   <s>jdoe</s>
</invoke>
```

検出に成功した場合は、true が返されます。検出に失敗した場合は、NULL が返されます。

XPRESS フォームへのアラートメッセージの挿入

警告 (WARNING)、エラー (ERROR)、または情報 (OK) アラートメッセージを XPRESS フォームに挿入できます。

注 この例では、警告 ErrorMessage オブジェクトをフォームに挿入する方法を示しますが、別の重要度レベルを割り当てることもできます。

1. Identity Manager IDE を使用して、警告を追加するフォームを開きます。
2. `<Property name='messages'>` をメインの EditForm または HtmlPage 表示クラスに追加します。
3. 次のサンプルコードの `<defvar name='msgList'>` コードブロックを追加します。
4. コードサンプル文字列のメッセージキー (アラートボックスに表示されるメッセージテキストを識別するキー) を置き換えます。

   `<message name='UI_USER_REQUESTS_ACCOUNTID_NOT_FOUND_ALERT_VALUE >`
5. ファイルを保存して閉じます。

コード例 2-37

```
<Display class='EditForm'>
   <Property name='componentTableWidth' value='100%'/>
   <Property name='rowPolarity' value='false'/>
   <Property name='requiredMarkerLocation' value='left'/>
   <Property name='messages'>
     <ref>msgList</ref>
   </Property>
</Display>
<defvar name='msgList'>
  <cond>
    <and>
      <notnull>
        <ref>username</ref>
      </notnull>
      <isnull>
        <ref>userview</ref>
      </isnull>
    </and>
    <list>
      <new class='com.waveset.msgcat.ErrorMessage'>
        <invoke class='com.waveset.msgcat.Severity' name='fromString'>
          <s>warning</s>
        </invoke>
        <message name='UI_USER_REQUESTS_ACCOUNTID_NOT_FOUND_ALERT_VALUE'>
          <ref>username</ref>
        </message>
      </new>
    </list>
  </cond>
</defvar>
```

警告以外の重要度レベルを表示するには、前述の例の `<s>warning</s>` を次の 2 つの値のいずれかに置き換えます。

- `error` -- Identity Manager の InlineAlert が赤色のエラーアイコンで表示されます。
- `ok` -- 成功またはその他の重要性が低いと考えられるメッセージについて、InlineAlert が青色の情報アイコンで表示されます。

次の場合、Identity Manager の InlineAlert は警告アイコンで表示されます。

```
<invoke class='com.waveset.ui.FormUtil' name='getRoles'>

<s>warning</s>

</invoke>
```

前述の `warning` は、`error` または `ok` にすることができます。

ウィザードフォームとタブ付きフォーム

ウィザードフォームとタブ付きフォームはどちらも、扱いにくい単一ページのフォームを、管理がより容易な、複数ペインのフォームに変えるメカニズムです。どちらのフォームも、論理セクションまたはページの間にセパレータを持ちます。これらのページセパレータは、Tabbed User Form のようにフォームの上部に配置されるタブである場合と、前後のページに移動するためのナビゲーションボタンを利用してページを切り替えながらユーザーに操作を促すウィザードである場合があります。

デフォルトの Tabbed User Form の XML バージョンについては、この章の「Tabbed User Form」を参照してください。

ウィザードフォームとは

次のような場合は、タスクから複数のフォームを呼び出すよりも、ウィザードフォームを使用するほうが便利です。

- ページ間の遷移ロジックが単純である
- ページ間で特権システム呼び出しが必要とされる

ウィザードフォームには、次に説明する 2 つのボタン行があります。

表 2-18 最初のボタン行

ボタン行	説明
上の行	フォームペインを切り替えるための「次へ」および「戻る」ボタン
次の行	次の表に示される、標準のユーザーフォームボタン。この行は、`noDefaultButtons` オプションを **true** に設定して、独自のボタンを実装することで制御できます。

2 番目のボタン行は、次のように異なります。

表 2-19 2 番目のボタン行

ウィザードのページ	デフォルトボタン
最初のページ	「次へ」、「キャンセル」
中間のページ	「戻る」、「次へ」、「キャンセル」
最後のページ	「戻る」、「OK」、「キャンセル」

ウィザードフォームの実装

ウィザードフォームの構文は、Tabbed User Form の構造によく似ています。ウィザードフォームを作成する方法は次のとおりです。

1. 最上位レベルのコンテナに、TabbedPanel の代わりに WizardPanel 表示クラスを割り当てます。
2. noCancel プロパティーを true に設定します。
3. ウィザードのページを格納した、1 つまたは複数の EditForm フィールドを定義します。

次の例には、説明のためのコメントが含まれます。

```
<Form>
   <Display class="HtmlPage"/> ----- これを設定しないと、インデントと色に問題が生じます
   <Field name='MainTabs'> -- タブページを格納した最上位コンテナの名前
      <Display class='TabPanel'/> -- 最上位コンテナの表示クラス、TabPanel または
WizardPanel のいずれか
   <Field name='Identity'> -- タブのラベル
      <Display class='EditForm'> -- 各 " ページ " は編集フォームである必要があります
         <Property name='helpKey' value='Identity and Password Fields'/>
      </Display>
   <Field name='waveset.accountId'>
      <Display class='Text'>
         <Property name='title' value='_FM_ACCOUNT_ID'/>
      </Display>
<Disable> <ref>waveset.id</ref> </Disable>

   </Field>
   </Field>

</Field>
```

ヒントおよび対策

- 検証エラーは、属性が表示されるページではなく、ユーザーが最後に表示したページに表示されます。これに対応するには、適切なページに戻るように促すように、検証メッセージに説明を追加します。
- 複雑なウィザードでは、プロセス全体のどの段階にいるかを視覚的にユーザーに表示してください。各ページの上部に表示されるラベルまたはセクション見出しを使用して、「**ページ 1**」のようなテキストを表示します。
- ウィザードフォームでは、条件付きナビゲーションを使用しないでください。この機能の実装が必要なときは、WizardPanel の直下にある各子要素の Disable 式を使用します。たとえば、次のようにします。

```
<Field name='Page2'>
```

```
    <Display class='EditForm'/>
    <Disable><neq><ref>showPage2</ref><s>true</s></neq></Disable>
...
</Field>
```

- フィールドやボタンは、ゲート変数を設定する、直前のページに配置します。無効化されたページは、遷移ロジックから自動的に削除されます。

デフォルトのユーザー作成フォームとユーザー編集フォームの代替フォーム

管理者がデフォルトのユーザーフォームを使用してユーザーを編集する場合、管理者がユーザーアカウントの編集を開始するのと同時に、そのユーザーが所有するすべてのリソースがフェッチされます。ユーザーが多くのリソースにアカウントを持つ場合、この処理に時間がかかる可能性があり、パフォーマンスを低下させかねません。このような環境に Identity Manager を配備したときは、デフォルトのユーザー作成インタフェースやユーザー編集インタフェースの代わりに、スケーラブルフォームを使用してみてください。

スケーラブルフォームの概要

スケーラブルフォームは、ユーザー数とリソース数が多い環境で、Identity Manager のユーザー作成インタフェースとユーザー編集インタフェースのパフォーマンスを改善するようにカスタマイズされたフォームです。パフォーマンスの改善は、次のような機能によるものです。

- リソースの差分フェッチ
- ユーザーリソースの選択的な参照
- 複数リソースの編集

Identity Manager には、デフォルトのユーザー編集フォームとユーザー作成フォームのスケーラブルバージョンが用意されています。

リソースの差分フェッチ

リソースの差分フェッチは、ネットワーク接続などの方法で、Identity Manager サーバーがリソースから直接、情報を取得する手段の 1 つです。管理者がデフォルトのユーザーフォームを使用してユーザーを編集する場合、通常は、管理者がユーザーアカウントの編集を開始するのと同時に、そのユーザーが所有するすべてのリソースがフェッチされます。これに対し、スケーラブルフォームの設計に込められた目的は、管理者が表示または変更するリソースのみをフェッチすることで、フェッチに制限をかけることです。

選択的な参照

選択的な参照は、スケーラブルフォームに組み込まれたもう 1 つの機能です。管理者は、所有ロール、リソースタイプ、またはリソースリストに基づいて、リソースを差分表示できます。

複数リソースの編集

複数リソースの編集により、管理者はリソースのサブセットを選択してリソース属性を編集できます。管理者は、ロール、リソースタイプ、またはリソースリストに基づいてサブセットを選択できます。

スケーラブルフォームの使用

次のようなケースでは、スケーラブルフォームの使用を検討してください。

- **管理者が、多数のリソースアカウントを持つユーザーを手動で編集する場合**。これらの状況でスケーラブルフォームを実装することで、管理者は、ユーザーのすべてのリソースアカウントのデータをフェッチするオーバーヘッドを回避し、特定のリソースアカウントを選択的に編集できます。このメカニズムは、ユーザーに関連付けられている特定のリソースタイプが、その他のリソースタイプと比較して応答が極端に遅い場合に特に便利です。
- ActiveSync などのカスタムプロビジョニングプロセスが、特定のリソースの更新のみを目的としている場合

注 別のリソースを参照する属性がフォームロジックに含まれる場合は、スケーラブルフォームを使用しないでください。この設定では、対象リソースをまとめてフェッチしない限り、このような相互参照属性に最新データが取り込まれません。

別のリソースを参照する属性がフォームロジックに含まれる場合は、スケーラブルフォームを使用しないでください。この設定では、対象リソースをまとめてフェッチしない限り、このような相互参照属性に最新データが取り込まれません。

新規ユーザーは、最初はリソースを持たないため、ユーザー作成フォームのスケーラブルバージョンには、限定的ではありますが、標準のデフォルトバージョンに対してメリットがあります。

使用できるスケーラブルフォーム

Identity Manager には、次の 2 つのスケーラブルユーザーフォームが用意されています。各フォームについては後述します。

- Dynamic Tabbed User Form。デフォルトの Tabbed User Form の代替フォームです。
- リソーステーブルユーザーフォーム。デフォルトの Tabbed User Form の代替フォームです。

Dynamic Tabbed User Form

管理者が編集を開始するのと同時にすべてのリソースをフェッチする、デフォルトの Tabbed User Form の代替フォームです。Dynamic Tabbed User Form には、差分フェッチ機能と、リソースタイプに基づいて複数のリソースを編集する機能が用意されています。

注 実装の詳細については、`WSHOME/samples/form_name.xml` の各ユーザーフォームに関するコメントを参照してください。

フォームのインポートとマッピング

Dynamic Tabbed User Form によるデフォルトの Tabbed User Form の代用には、次の 3 つのフォームが関連します。

表 2-20 Dynamic Tabbed User Form と関連するフォーム

フォーム	説明
Dynamic Tabbed User Form	デフォルトの Tabbed User Form の機能を持ちますが、リソースタイプごとに 1 つのタブを動的に作成します。
動的ユーザーフォーム	ユーザーフォームのリソースタイプタブを作成するためのフィールドを持ちます。
動的フォームの規則ライブラリ	指定されたユーザーフォームを持たないリソースの属性を動的に出力するための規則ライブラリを持ちます。
動的リソースフォーム	現時点で Dynamic Tabbed User Form との互換性を持つすべてのフォームを持ちます。ユーザーは、このリストをカスタマイズできます。

Dynamic Tabbed User Form のインストールは、フォームのインポートと、フォームマッピングの変更という 2 段階で行われます。

手順 1: フォームのインポート

1. Identity Manager のメニューバーから、「**設定**」>「**交換ファイルのインポート**」を選択します。
2. ファイル名 (`dynamicformsinit.xml`) を入力するか、「**参照**」をクリックして、./sample ディレクトリ内の `dynamicformsinit.xml` ファイルを探します。
3. 「**インポート**」をクリックします。Identity Manager にインポートに成功したことを示すメッセージが表示されます。

手順 2: フォームマッピングの変更

ユーザーフォームをエンドユーザーに割り当てるには、2 つの方法があります。これらのフォームマッピングを編集する方法は、実際の環境で管理者がこれらのフォームをどのように使用するかによって異なります。次の方法があります。

- **すべての管理者のデフォルトユーザーフォームとしてスケーラブルユーザーフォームを割り当てる。**この方法で割り当てるときは、「デフォルトユーザーフォームとしてのスケーラブルユーザーフォームの割り当て」を参照してください。Identity Manager の管理者は、すべての管理者が使用する 1 つのフォームを割り当てることができます。
- **特定の管理者に、個別にスケーラブルユーザーフォームを割り当てる。**この方法で割り当てるときは、「管理者ごとのスケーラブルユーザーフォームの割り当て」を参照してください。

デフォルトユーザーフォームとしてのスケーラブルユーザーフォームの割り当て

1. メニューバーから、「**設定**」>「**フォームおよびプロセスマッピングの設定**」を選択します。
2. 「フォームマッピング」セクションで、「フォームタイプ」列から `userForm` を探します。
3. 「マップされるフォーム名」列の下の入力ボックスで、Dynamic Tabbed User Form を指定します。

管理者ごとのスケーラブルユーザーフォームの割り当て

1. メニューバーで、「**アカウント**」>「**ユーザーの編集**」を選択します。
2. ユーザーを、次のいずれかの方法で選択します。
   - ユーザー名をクリックしてから「**編集**」をクリックします。

     または

- ユーザー名を右クリックしてポップアップメニューを表示させ、「**編集**」メニューオプションを選択します。

3. デフォルトのユーザー編集フォームが表示されたら、「セキュリティー」タブをクリックします。
4. 「ユーザーフォーム」フィールドから「Dynamic Tabbed User Form」を選択します。
5. 「**保存**」をクリックして、設定を保存します。

リソーステーブルユーザーフォーム

リソーステーブルユーザーフォームには、ユーザー編集フォームのスケーラブルバージョンのほとんどの駆動ロジックが含まれます。このフォームは、差分フェッチ機能と、リソースタイプに基づいて複数リソースを編集する機能を実装します。

実装の詳細については、`WSHOME/samples/resourcetableformsinit.xml` に記載されているコメントを参照してください。

フォームのインポートとマッピング

リソーステーブルユーザーフォームによるデフォルトの Tabbed User Form の代用には、次の 5 つフォームが関連します。

表 2-21 リソーステーブルユーザーフォームと関連するフォーム

フォーム	説明
リソーステーブルユーザーフォーム	ナビゲーション、差分フェッチ、およびフォームのレイアウトに使用される、グローバルに利用できるすべてのフィールドを持ちます。リソース関連のその他すべてのスケーラブルフォームは、このメインフォームによって駆動されます。
リソーステーブルユーザーフォームライブラリ	リソーステーブルユーザーフォームのプライマリフィールドを持ちます。ブレッドクラムフィールドとナビゲーションフィールドが含まれます。
リソーステーブルアカウント情報フォーム	リソーステーブルフォームのアカウント情報セクションのフィールドを持ちます。
リソーステーブル規則ライブラリ	ユーザーのリソースの取得、カウント、分析に適用される規則ライブラリを持ちます。これは、主にユーザーフォームライブラリで使用され、ロールとリソースに基づいてテーブルデータを生成します。
リソーステーブルユーティリティーライブラリ	たとえば、ロールごと、またはタイプごとにリソースを取得する規則のように、リソーステーブルフォームでの選択プロセスで使用される規則を持ちます。

リソーステーブルユーザーフォームのインストールは、フォームのインポートと、フォームマッピングの変更という 2 段階で行われます。

手順 1: フォームのインポート

1. Identity Manager のメニューバーから、「**設定**」>「**交換ファイルのインポート**」を選択します。
2. ファイル名を入力するか、「**参照**」をクリックして、`WSHOME/sample/resourcetableforms.xml` を探します。このファイルをインポートすると、次の内容もインポートされます。

手順 2: フォームマッピングの変更

1. メニューバーから、「**設定**」>「**フォームおよびプロセスマッピングの設定**」を選択します。
2. 「フォームマッピング」セクションで、「フォームタイプ」列から `userForm` を探します。
3. 「マップされるフォーム名」列の下の入力ボックスで、リソーステーブルユーザーフォームを指定します。

スケーラブルフォームのカスタマイズ

スケーラブルユーザーフォームのインポートとマッピングが完了したら、それをカスタマイズしてください。差分フェッチを有効にするには、次の情報を指定しなければなりません。

- **最初にフェッチするリソースアカウント**。フェッチの対象とするリソース名を指定するには、`TargetResources` フォームプロパティーを使用します。
- 最終的な保存の段階で**更新される操作**。

動的なユーザーフォームとリソーステーブルユーザーフォームはどちらも、ユーザーのリソースに固有の属性を表示するのに、リソース固有のフォームを使用します。次のユーザーフォームは、スケーラブルフォームで使用するために WSHOME/sample/forms ディレクトリに追加されました。

- ./ACF2UserForm.xml
- ./ActivCardUserForm.xml
- ./ADUserForm.xml
- ./AIXUserForm.xml
- ./BlackberryUserForm.xml
- ./ClearTrustUserForm.xml
- ./HP-UXUserForm.xml

- ./NDSUserForm.xml
- ./OS400UserForm.xml
- ./PeopleSoftCompIntfcUserForm.xml
- ./RACFUserForm.xml
- ./SAPPortalUserForm.xml
- ./SolarisUserForm.xml
- ./SunISUserForm.xml
- ./TopSecretUserForm.xml

これらのフォームは、Dynamic Tabbed User Form とリソーステーブルユーザーフォームの両方とともに自動的にインポートされます。

配備したシステムが上記以外のリソースタイプを使用している場合は、スケーラブルフォームには、スキーママッピングに指定されているすべての属性名と値を示すデフォルトのユーザーフォームが表示されます。上記以外の既存のカスタムリソースユーザーフォームを使用するときは、スケーラブルフォームとの互換性を維持するために、特定の修正を加えてください。次に、互換性の維持に必要な手順の一部について説明します。

注 この修正の例として、上記リストのいずれかのフォームを参照してください。

スケーラブルユーザーフォームとの互換性を維持するためのリソースフォームのカスタマイズ

独自にカスタマイズしたリソースフォームを追加して、Dynamic Tabbed User Form またはリソーステーブルユーザーフォームと併用する場合は、次の一般的な手順に従ってください。

手順 1: 動的リソースフォームの修正

独自のリソースフォームを追加する方法については、`dynamicformsinit.xml` ファイルを参照してください。このファイルで動的リソースフォームを検索し、フォームに記載されている手順を実行します。

注 フォームには手順の説明がコメントとして記載されていますが、フォームのインポート後は表示されません。

手順 2: リソースフォームの修正

前述のリストに含まれないフォームを使用している場合は、互換性を維持できるようにリソースフォームに修正を加えます。前述のリストに含まれるいずれかのファイルを例として参照してください。方法は、各リソースフォームの冒頭に記載されています。

Tabbed User Form のカスタマイズ : 属性エリアへのパスワードフィールドの移動

同時に異なるパスワードを持つ 2 つのリソースを更新するには、割り当てられているリソースごとにパスワードフィールドを作成します。たとえば、「アカウント」ページの AD リソースの属性エリアに「AD」パスワードフィールドを設けることによっても、その他のリソースとは別に設定できるパスワードポリシーを適用できます。

デフォルトパスワードポリシーの表示

デフォルトでは、パスワードポリシーの情報は、「**アカウント**」>「**ID**」タブに次のように表示されます。

パスワードフィールドを、デフォルトの位置である ID エリアから属性エリアに移動するには、次の 3 段階の手順を実行して、Identity Manager のデフォルトのパスワード同期メカニズムを無効にしなければなりません。

1. `manualPasswordSynchronization` チェックアウトプロパティーを設定します。
2. Tabbed User Form に `Field` コンポーネントと `FieldLoop` コンポーネントを追加します。
3. Tabbed User Form に、リソースに固有のパスワードフィールドを追加します。

次に、これらの手順について詳しく説明します。

手順 1: manualPasswordSynchronization チェックアウトプロパティーの設定

フォームに次のプロパティーを追加して、`manualPasswordSynchronization` ビューチェックアウトオプションを設定します。

```
<Form>
   <Properties>
     <Property name='manualPasswordSynchronization' value='true'/>
...
   </Properties>
```

```
...
</Form>
```

manualPasswordSynchronization を true に設定すると、Identity Manager は、パスワードシンクロナイザを使用する代わりに、リソースごとのパスワードフィールドを表示します。

手順 2: パスワードの同期の無効化

「パスワード」ビューの下の selectAll フラグをオフにすることで、パスワードの同期を無効にできます。このためには、デフォルトフォームに次のフィールドを追加します。

```
<Field name='password.selectAll'>
   <Comments>
    同期を行わないように、selectAll フラグを強制的にオフにします。
    ビューハンドラによって true に設定されることがあるため、この無効化が必要です。
   </Comments>
     <Expansion><s>false</s></Expansion>
</Field>
   <FieldLoop for='res'>
     <expression>
       <remove>
          <ref>password.targets</ref>
          <s>Lighthouse</s>
       </remove>
     </expression>
   <Comments>
   また、新規アカウントではビューハンドラによってデフォルトで true に設定されるため、
   個々の選択フラグも強制的に false に設定し、リソースごとに
   パスワードプロンプトを表示する必要があります。
   </Comments>
   <Field name='password.accounts[$(res)].selected'>
   <Expansion><s>false</s></Expansion>
   </Field>
</FieldLoop>
```

手順 3: 「属性」ページへのリソース固有のパスワードフィールドの追加

各リソースに固有のパスワードフィールドを次のように記述します。

```
<Field name='accounts[resname].password'>
```

ポリシー検証の無効化

次のフィールドをユーザーフォームに追加することで、フォームでのポリシー検証を無効にすることができます。

```
<Field name='viewOptions.CallViewValidators'>
  <Display class='Hidden'/>
    <Expansion>
        <s>false</s>
    </Expansion>
</Field>
```

このフィールドは、`modify.jsp` の OP_CALL_VIEW_VALIDATORS フィールドの値より優先されます。

ユーザーパスワード履歴の追跡

Identity Manager では、管理者が行うユーザーパスワードの変更はデフォルトでは追跡されません。次の方法を利用すれば、管理者はこのデフォルトの動作を変更できます。配備に合わせて最適なオプションを選択してください。

オプション 1: フォームへのビューオプションの追加

次の方法でビューオプションをターゲットフォームに追加できます。このビューオプションは、あらゆるシステム設定より優先されることに注意してください。つまり、このビューオプションを `true` に設定し、関連するシステム設定属性が `false` の場合は、Identity Manager はこのビューオプションに従ってシステム設定を無視します。

ActiveSync 処理に関係のないターゲットフォームを操作している場合は、ターゲットフォーム (一般にユーザーフォーム) の `savePasswordHistory` 属性を次のように設定します。

コード例 2-38

```
<Field name='savePasswordHistory'>
   <Default>
      <Boolean>true</Boolean>
   </Default>
</Field>
```

Active Sync が設定されているときにパスワード変更を記録する場合は、別の方法で savePasswordHistory ビューオプションを設定する必要があります。Synchronize User Password TaskDefinition は、次のアクションを SetPasswordView アクティビティーに追加することで変更できます。

コード例 2-39

```
<Activity id='5' name='SetPasswordView'>
   <Action id='0'>
     <expression>
          <set name='PasswordView.resourceAccounts.password'>
             <ref>password</ref>
          </set>
     </expression>
   </Action>
<!-- アクションをここに追加 -->
   <Action id='1'>
    <expression>
         <set name='PasswordView.savePasswordHistory'>
             <Boolean>true</Boolean>
        </set>
    </expression>
   </Action>
   <!-- 終わり -->
   <Action id='2'>
     <expression>
       <dolist name='account'>
        <ref>PasswordView.resourceAccounts.currentResourceAccounts</ref>
```

オプション 2: システム設定オブジェクトの設定の変更

関連するシステム設定オブジェクトの設定を編集することもできます。ログインアプリケーションの savePasswordHistory オプションを設定できます。

1. システム設定オブジェクトで、次のパスを探します。

   ```
   security.admin.changePassword.[login interface]
   ```

2. 該当するインタフェースの savePasswordHistory の値を false から true に切り替えます (次の例を参照)。デフォルトでは、これらの値は false です。

コード例 2-40　　システム設定オブジェクトの設定の変更

```
<Attribute name='security'>
  <Object>
    <Attribute name='admin'>
      <Object>
         <Attribute name='changePassword'>
            <Object>
               <Attribute name='Administrator Interface'>
                  <Object>
                     <Attribute name='savePasswordHistory'>
                        <Boolean>false</Boolean>
                     </Attribute>
                  </Object>
               </Attribute>
               <Attribute name='Command Line Interface'>
                  <Object>
                     <Attribute name='savePasswordHistory'>
                        <Boolean>false</Boolean>
                     </Attribute>
                  </Object>
               </Attribute>
               <Attribute name='IVR Interface'>
                  <Object>
                     <Attribute name='savePasswordHistory'>
                        <Boolean>false</Boolean>
                     </Attribute>
                  </Object>
               </Attribute>
               <Attribute name='SOAP Interface'>
                  <Object>
                     <Attribute name='savePasswordHistory'>
                        <Boolean>false</Boolean>
                     </Attribute>
                  </Object>
               </Attribute>
               <Attribute name='User Interface'>
                  <Object>
                     <Attribute name='savePasswordHistory'>
                        <Boolean>false</Boolean>
                     </Attribute>
                  </Object>
               </Attribute>
            </Object>
         </Attribute>
    </Object>
  </Attribute>
  <Attribute name='authn'>
     <Object> ..
```

SPML インタフェースを使用してパスワード履歴の記録を許可するには、システム設定オブジェクトで次のように設定する必要があります。

```
security.admin.changePassword.Command Line Interface
```

カスタマイズしたフォームの検証

実行時環境に実装する前に、編集したフォームに関する情報を、次の方法で収集できます。

- エラーロギングを使用して、フォームフィールド内の式ステートメントのエラーを確認します。
- フォームエディタを使用して、個々の式ステートメントの構文を検証します。フォームエディタは、パーサーから返された構文エラーメッセージをポップアップウィンドウに表示します。フォームエディタの使用方法については、フォームエディタに関連するオンラインヘルプを参照してください。

エラーロギングの有効化と無効化

Identity Manager のエラーロギングユーティリティーは、フォームに含まれる式の構文に関する問題を標準出力に出力します。XPRESS のトレースを有効にすると、`<block>` タグを使用して、フォームのサブセットの XPRESS ステートメントにログメッセージを限定できます。XPRESS ステートメントの処理の詳細情報を取得するには、`waveset.properties` ファイルの設定オプション、`xpress.trace` を true に設定します。このオプションを true に設定すると、XPRESS ステートメントのすべての評価が、トレースメッセージとしてコンソールに出力されます。これは、XPRESS API によるトレースを有効にするためにコードを変更できない実行中アプリケーションで評価される、ステートメントのデバッグに使用できます。

すべての XPRESS フィールドの XPRESS トレースは、コマンド行または Identity Manager の管理者インタフェースから有効にできます。この方法で有効にしたトレースは、すべてのフィールドに影響します。ログメッセージをフォームのサブセットに限定するときは、`<block>` タグセットを使用して、エラートレースの対象を `<block></block>` タグ内のコードのみに制限します。

Identity Manager のすべての式の評価に関するエラーロギングをコマンド行から有効にする方法は次のとおりです。

1. 編集する `config/waveset.properties` ファイルを開きます。
2. `xpress.trace=false` という行を探します。
3. `false` という値を `true` に変更します。
4. ファイルを保存します。

5. アプリケーションサーバーを再起動します。

これとは別に、Identity Manager の管理者インタフェースからエラーロギングの有効化と無効化を切り替えることもできます。

1. Configurator として Identity Manager にログインします。
2. 「**デバッグ**」を選択して「Debug」ページを開きます。
3. 「デバッグ」ページで、「**Reload Properties**」を選択します。

XPRESS のトレースを無効にするときは、`xpress.trace` の値を `false` に変更し、`waveset.properties` ファイルを読み込みなおします。

フォームとフォームフィールドのサンプル

ここでは、製品に付属するデフォルトフォームのサンプルを示します。また、環境にサンプルフォームを組み込む方法についても説明します。

注 Identity Manager の特定バージョンに付属するフォームのバージョンによっては、これらのサンプルに若干の相違がある場合があります。

- Tabbed User Form
- エンドユーザーメニューフォーム
- Anonymous User Menu Form

ユーザーフォームライブラリ

フォームは、フォーム自体としてではなく、フィールドの集合を格納したコンテナとして使用できます。Identity Manager は、ユーザーフォームライブラリというオブジェクトを利用して、フォームのこの用途をサポートしています。このライブラリには、パスワードの変更などに使用される、リソースの詳細な選択に関連する複雑なフィールドが含まれます。

次のリストは、ユーザーライブラリに割り当てられている各ライブラリの概要を示しています。

ユーザーライブラリ ユーザーフォームのプライマリライブラリ。このライブラリは、このリストに示されるその他のライブラリを含み、秘密質問の回答を表示、編集するための **AuthenticationAnswers** フィールドを定義します。

パスワードライブラリ	パスワードの仕様と同期に関するフィールドを持ちます。
アカウント概要ライブラリ	ユーザーに関連付けられているアカウントに関する、読み取り専用の概要情報を表示するフィールドを持ちます。
アカウントリンクライブラリ	アカウントのリンクと、リソースごとの複数のアカウントに関するフィールドを持ちます。
ユーザーセキュリティーライブラリ	機能、フォームの割り当て、承認の転送など、ユーザーセキュリティーに関するフィールドを持ちます。

ユーザーフォームライブラリ

このライブラリには、次のような、リソースアカウントビューに関連するフィールドのみが含まれます。

- ChangeUserPassword
- Deprovision
- Disable
- Enable
- Password
- RenameUser
- ResetPassword
- ResetUserPassword
- ResourceAccounts

このライブラリは主に、Identity Manager ユーザーに関連付けられているリソースアカウントの情報を表示したテーブルから構成され、各種操作で使用されるこれらの情報を選択できます。

サンプルフォームライブラリの使用

`<FormRef>` 要素を使用することで、Identity Manager に付属するサンプルフォームは、カスタマイズしているどのフォームにも含めることができます。

環境にサンプルフォームを追加する一般的な手順は次のとおりです。

手順 1: 規則のインポート

手順 2: フォームのインポート

手順 3: デフォルトフォームからの新規フォームの作成 (Include 参照の追加と、フォーム参照の追加)

手順 1: 規則のインポート

Identity Manager の管理者インタフェースを使用して、サンプル規則を読み込みます。次の手順で実行します。

1. Identity Manager のメニューバーから、「**設定**」>「**交換ファイルのインポート**」を選択します。
2. サンプルファイル名を入力するか、「**参照**」をクリックして、`idm\sample\rules` ディレクトリからファイルを探します。

   サンプル規則の共通ファイルの名前は次のとおりです。

   - sample\rulesListGroups.xml
   - sample\rules\NamingRules.xml
   - sample\rules\RegionalConstants.xml

   サンプル規則のリソースファイルの名前は次のとおりです。

   - sample\rules\ADRules.xml
   - sample\rules\NDSRules.xml
   - sample\rules\NTRules.xml
   - sample\rules\OS400UserFormRules.xml
   - sample\rules\RACFUserFormRules.xml
   - sample\rules\TopSecretUserFormRules.xml
3. 「**インポート**」をクリックします。Identity Manager にインポートに成功したことを示すメッセージが表示されます。

手順 2: フォームのインポート

Identity Manager の管理者インタフェースを使用して、サンプルフォームを読み込みます。次の手順で実行します。

1. Identity Manager のメニューバーから、「**設定**」>「**交換ファイルのインポート**」を選択します。
2. サンプルファイル名を入力するか、「**参照**」をクリックして、`idm\sample\forms` ディレクトリからファイルを探します。サンプルフォームのファイル名は次のとおりです。

   - sample\forms\ACF2UserForm.xml
   - sample\forms\AIXUserForm.xml

- sample\forms\HP-UXUserForm.xml
- sample\forms\NDSUserForm.xml
- sample\forms\NTform.xml
- sample\forms\OS400UserForm.xml
- sample\forms\SecurIDUserForm.xml
- sample\forms\SolarisUserForm.xml
- sample\forms\TopSecretUserForm.xml
- sample\forms\vitalStatform.xml

3. 「**インポート**」をクリックします。Identity Manager にインポートに成功したことを示すメッセージが表示されます。

手順 3: Tabbed User Form の更新 (Include 参照の追加)

Tabbed User Form または作成するメインフォームから、サンプルフォームに Include 参照を追加します。次の手順で実行します。

1. Tabbed User Form をコピーし、名前を変更します (たとえば、<*CompanyName*>tabbedUserForm)。
2. Web ブラウザのアドレス行にこの URL を入力し、**Enter** キーを押します。

   http://*ApplicationServerHost:Port*/idm/debug
3. 認証が完了すると、「システム設定」ページが表示されます。
4. 「タイプ」リストから UserForm オプションを選択し、「**List Objects**」をクリックします。
5. <*CompanyName*>tabbedUserForm または、作成したメインフォームの横の「**編集**」をクリックします。
6. フォームの Include エリアを変更し、次の例に太字で表示されている各サンプルフォームを追加します。

```
<Include>
  <ObjectRef type='UserForm' id='#ID#UserForm:UserformLibrary'
name='UserForm Library'/>
  <ObjectRef type='UserForm' name='UserFormName'/>
<Include>
```

   UserFormName に設定できる値は次のとおりです。

   - ACF2 User Form
   - AIX User Form
   - HP-UX User Form

- LDAP Active Sync User Form
- Netegrity Siteminder Admin Form
- Netegrity Siteminder LDAP User Form
- Netegrity Siteminder ExampleTable User Form
- NDS User Form
- NT User Form
- Open Networks User Form
- OS400 User Form
- Oracle ERP User Form
- RACF User Form
- RSA ClearTrust User Form
- SecurID User Form
- Skeleton Database Active Sync User Form
- Solaris User Form
- Tivoli Access Manager
- Top Secret User Form
- グローバル属性 (vitalStatform.xml)

フォームを保存する前に、継続して次の手順を実行します。

手順 4: Tabbed User Form の更新 (フォームの追加)

各サンプルフォームの FormRef を追加するには、それをメインフォームに追加します。

1. メインフォーム内の適切な場所に、サンプルフォームごとに次の行を追加します。

   ```
   <FormRef name='UserFormName'/>
   ```

2. 次の行を削除します。

```
<FormRef name='MissingFields'/>
```

3. 「**保存**」をクリックして、フォームの変更を保存します。

コンプライアンス関連のフォーム

表 2-22　コンプライアンス関連のフォーム

フォーム名	一般的な目的
アクセス承認リスト	アテステーション作業項目のリストを表示します
アクセスレビュー削除確認	アクセスレビューの削除を確認します
アクセスレビューアボート確認	アクセスレビューの終了を確認します
アクセスレビューダッシュボード	すべてのアクセスレビューのリストを表示します
アクセスレビュー概要	特定のアクセスビューの詳細を表示します
アクセススキャンフォーム	アクセススキャンを表示または編集します
アクセススキャンリスト	すべてのアクセススキャンのリストを表示します
アクセススキャン削除確認	アクセススキャンの削除を確認します
UserEntitlementForm ユーザーエンタイトルメント概要フォーム	ユーザーエンタイトルメントの内容を表示します
違反詳細フォーム	コンプライアンス違反の詳細を表示します
是正リスト	是正作業項目のリストを表示します
監査ポリシーリスト詳細	監査ポリシーのリストを表示します
監査ポリシー削除確認フォーム	監査ポリシーの削除を確認します
相反違反詳細フォーム コンプライアンス違反概要フォーム	SOD 違反マトリックスを表示します

FormUtil メソッドの使用

ここでは、このクラスのメソッドに関する検討事項の概要と使用方法のヒントについて説明します。FormUtil メソッドは、ビューを変換するときにフォームが使用するユーティリティーです。各メソッドについて具体的な情報は、FormUtil の Javadoc に記載されています。

各メソッドについては、<*distribution*>\REF\javadoc (<*distribution*> はインストールディレクトリ) を参照してください。

FormUtil クラスメソッド

FormUtil クラスは、フォームオブジェクト内の XPRESS 式から呼び出せるユーティリティーメソッドの集合です。FormUtil メソッドは、通常、Select フィールドと MultiSelect フィールドの valueMap 内で、使用できる値のリストを制限するために使用されます。また Identity Manager には、日付やディレクトリ DN などの文字列値をフォーマットするための追加メソッドも用意されています。

メソッドのコンテキストについて

FormUtil メソッドから Identity Manager リポジトリにアクセスする必要のあるときは、コンテキストオブジェクトが必要になります。Lighthouse コンテキストは認証済みセッションを表現しているので、可視性およびアクションの制限を適用するための認証チェックが必要となります。

コンテキストのフェッチ

ほとんどの FormUtil メソッドでは、ビュー属性 display.session を参照し、最初の引数として LighthouseContext または Session オブジェクトを渡す必要があります。フォームは、ベースコンテキストプレフィックスとともに使用されることが多いので、ベースコンテキストプレフィックスを削除できるように、display.session を参照する際には先頭に常にコロンを付けて (:display.session) 参照することをお勧めします。

メソッドの呼び出し

フォーム内から FormUtil メソッドを呼び出すときは、次の構文を使用します。

```
<invoke class='com.waveset.ui.FormUtil'
   name = 'method_name'>
      <ref>:display.session</ref>
      <s>arg2</s>
</invoke>
```

この name フィールドは、メソッドの名前を表します。

コンテキストが不明な状態での FormUtil メソッドの呼び出し

select は、どの変数に Lighthouse コンテキストが指定されているかがわからなくても使用できます。これにより、メソッド呼び出しを再使用することが簡単になります。

```
<select>
   <ref>:display.session</ref>
   <ref>context</ref>
</select>
```

フォームでは、<ref>:display.session</ref> を使ってコンテキストを指定します。ただしワークフローでは、同じ FormUtil 呼び出しで <ref>context</ref> を代わりに使用します。

コンテキストをフェッチするための 3 番目の方法は、WF_CONTEXT オブジェクトを使用して getLighthouseContext メソッドの呼び出しを行う方法です。3 つすべての手法を 1 つのルールにまとめたものを次に示します。これは後で使用できます。

コード例 2-41

```
<rule name='Get Context'>
<select>
   <ref>:display.session</ref>
   <ref>context</ref>
   <invoke name='getLighthouseContext'>
      <ref>WF_CONTEXT</ref>
   </invoke>
</select>
</rule>
```

ベストプラクティス

簡単な <select> ステートメントを含む、Get Context のような名前の規則を作成できます。この規則は、次の例のように、必要とする FormUtil メソッドを呼び出すときに任意のフォームまたはワークフローから呼び出すことができます。

コード例 2-42

```
<invoke name='getObject'>
 <!-- 通常は、:display.session などをここに記述します。代わりに前述の規則を呼び出します -->
<rule name='Get Context'/>
 <s>User:</s
<s>SamUser</s>
</invoke>
```

この呼び出しをより大きなユーティリティー規則の中で行えば、フォームとワークフローの両方で使用できます。

よく呼び出されるメソッド

もっともよく使用される FormUtil メソッドの簡単な説明を次の表に示します。

メソッド	説明
callResourceMethod	引数を指定して、リソース上の指定メソッドを呼び出します。
buildDn、buildDns	名前 (1 つまたは複数) とその名前に付加するベースコンテキストを受け取ります。このメソッドは、完全修飾識別 (DN) 名を返します。たとえば、group1 と dc=example,dc=com を渡すと、cn=group1, dc=example, dc=com という文字列が返されます。
checkStringQualityPolicy	文字列ポリシーに対して、指定された文字列の値を検証します。
controlsAtLeastOneOrganization	現在認証されているユーザーが、1 つまたは複数の組織 (ObjectGroup) 名のリストに含まれるいずれかの組織を制御できるかどうかを返します。サポートされる組織リストには、ObjectGroup タイプのすべてのオブジェクトのリスト表示で返される組織が含まれます。

メソッド	説明
getObject	リポジトリからオブジェクトを取得します (認証されていることが前提)。
getObjectNames	指定されたタイプのオブジェクトでセッションの所有者 (または現在のログインユーザー) がアクセスできるオブジェクトの名前のリストを返します。オプションマップに追加パラメータを指定し、返される名前のリストを制御できます。 オブジェクト名のリストを返すときは、`session.getObjects()` を実行するよりも、このメソッドを使用することをお勧めします。メソッドは、まず、ObjectCache を検索し、必要に応じて次にリポジトリを検索します。
getOrganizationsDisplayNames	現在の管理者がアクセスできる、組織ハンドルのリストを返します。組織の選択リストおよび複数選択リストを使用するフォームでは、このメソッドを使用する必要があります。
getResources	特定のリソース属性値 (type=LDAP など) と一致するリソースの名前リストを作成します。現在のリストを渡すと、リストはマージされます。
getResourceObjects	各オブジェクトが、タイプ、名前、ID (DN または完全修飾名)、および要求される `searchAttrsToGet` の値の属性セットを持つオブジェクトのリストを返します。返される値は、GenericObjects のリストです。それぞれの GenericObject には、マップへのアクセスと同様の方法でアクセスできます。各オブジェクトで `get` メソッドを呼び出します。このとき、属性名が渡され、属性値が返されます。
getRoles	現在の管理者がアクセスできる、ロール名のリストを返します。現在の値、または現在のリストを指定すると、返されるロール名にリスト内のロール名が追加されます。
geUnassignedApplications	ユーザーのプライベートアプリケーションに適したアプリケーション名のリストを作成します (プライベートアプリケーションは、ユーザーに直接割り当てられているアプリケーション)。これは、ロールを通じてすでにユーザーに割り当てられているアプリケーションの名前を除く、すべての利用可能なアプリケーションのリストです。 作成されるリストは、フォームでのプライベートアプリケーションの割り当てに便利です。
getSubordinates	ユーザーの、指定された管理対象部下のリストを取得します。

メソッド	説明
getUnassignedResources	ユーザーのプライベートリソースに適したリソース名のリストを作成します (プライベートリソースは、ユーザーに直接割り当てられているリソース)。これは、ロールを通じてすでにユーザーに割り当てられているリソースの名前を除く、すべての利用可能なリソースのリストです。 作成されるリストは、フォームでのプライベートリソースの割り当てに便利です。
getUsers	このメソッドの最初の変数は、すべてのユーザーを返します。2 番目の変数は、デフォルトではすべてのユーザーを返しますが、リストをさらにフィルタリングするためのオプションのマップを指定できます。
listResourceIObjects	指定したタイプ (たとえば、`group`) のリソースオブジェクトのリストを取得します。このメソッドの最初の試行では、サーバーの `resourceObjectListCache` からリストが取得されます。見つかった場合は、リストが返されます。 リストが見つからなかった場合は、結果リストのマージ、ソート、および重複削除を行う前に、メソッドは各リソースで `listResourceObjects` メソッドを呼び出します。最後に、同じリソースからの同じリソースオブジェクトタイプを求める次回以後の要求に備えて、この新しいリストをサーバーの `resourceObjectListCache` にキャッシュします。 このメソッドは、現在認証されている管理者 (たとえば、`subject`) として実行されます。変数は、単一のリソース ID、または `subject` 文字列と既存のセッションを値としてとります。 このメソッドは、次の条件に応じて、異なる複数の変数をとります。 メソッドが、単一のリソースを返すか、リソースリストを返すか。 キャッシュをクリアする必要があるかどうか。 メソッドが、セッション ID (ユーザーがすでに認証されている場合に実装される) を送信するか、主体を表す文字列 (`subjectString`) を送信するか。通常は、`Session` を使用します。
testObject	主体がオブジェクトの表示を認証されていない場合でも、指定したオブジェクトが存在するかどうかを確認します。ツリー全体を表示できない管理者がユーザーの新規作成プロセスを開始するときは、このメソッドを使用して、重複オブジェクトの作成を防止してください。

メソッド	説明
testUser	主体がオブジェクトの表示を認証されていない場合でも、指定したユーザーが存在するかどうかを確認します。ツリー全体を表示できない管理者がユーザーの新規作成プロセスを開始するときは、このメソッドを使用して、重複オブジェクトの作成を防止してください。
hasCapability、hasCapabilities	ユーザーが指定された 1 つまたは複数の機能 (文字列) を持つかどうかを確認します。このメソッドは、直接割り当てられている機能、または AdminGroups、AdminRoles、またはその両方によって間接的に割り当てられている機能を調べます。`session` の値が必要です。

FormUtil を使用するときのヒント

一般的な FormUtil メソッドを実装するときには、オブジェクトを取得するメソッドで注意が必要になります。特に、次のメソッドは注意して使用する必要があります。

- `getObject`
- `getResourceObject`
- `getUnassigned*`

ここでは、これらのメソッドを使用する際の基本的なヒントを紹介します。

getObject を使用してオブジェクトをリポジトリからフェッチ

`getObject` メソッドは、リポジトリ内からオブジェクトを取得するためにもっともよく使用されるメソッドです。このメソッドを使用するときは、実際の (Java) オブジェクトを取得していることに注意してください。このオブジェクトの属性にアクセスするには、オブジェクトを取得するメソッドの呼び出し元にオブジェクトをラップする必要があります。各オブジェクトでは固有の取得操作が使用されるため、詳細については個々のオブジェクトの Javadoc を参照してください。

WSUser オブジェクトを操作するときは、WSUser オブジェクトの `toHashMap()` メソッドを使用できます。このメソッドは、次の例のように、オブジェクトを Java の HashMap と同等の GenericObject に変換します。

コード例 2-43

```
<set name='wsUserObj'>
   <invoke name='getObject'>
<!-- 通常は、:display.session などの値をここに記述します。代わりに前述の規則を呼び出
します -->
       <rule name='Get Context'/>
       <s>User:</s>
       <s>SamUser</s>
   </invoke>
</set>
<set name='wsGenericObj'>
   <invoke name='toHashMap'>
   <ref>wsUserObj</ref>
   </invoke>
</set>
```

WSUser オブジェクトから accountId を取得するときは、次を使用します。

```
<invoke name='getAccountId'>

 <ref>wsUserObj</ref>

</invoke>
```

ただし、ユーザーを操作しているときは、このメソッドが面倒なことがあります。WSUser オブジェクトを使った処理が次のように GenericObject に変換されています。

```
<ref>wsGenericObj.waveset.accountId</ref>
```

基本的に、ビューなどの GenericObject は WSUser オブジェクトよりも操作が大幅に簡単です。オブジェクトを直接操作することがわずらわしいときは、関連するオブジェクトのビューをチェックアウトすることを検討してください。

注 処理によっては、基本となる PersistentObject がリポジトリによってロックされます。一般に、checkout という語句で始まるメソッドはオブジェクトをロックし (checkin はロック解除)、get で始まるメソッドはオブジェクトをロックしません。

getResourceObject を使用してオブジェクトをリソースからフェッチ

getResourceObject 処理は、指定されたリソースから指定されたオブジェクトを GenericObject として返します。フェッチできるオブジェクトおよびリストでサポートされる属性については、『リソースリファレンス』を参照してください。このメソッドにより、対応するリソースアダプタが呼び出されます。

getUnassigned* メソッドの使用

getUnassigned* メソッドは、lh コンテキストから渡されたユーザーに現在割り当てられていない機能を取得します。これらのメソッドは、現在は割り当てられていない機能を選択できる (それらの機能へのアクセスを要求できる) ユーザーフォームを作成する必要があるときに便利です。たとえば、次のようにします。

コード例 2-44

```
<Field name='waveset.resources'>
   <Display class='MultiSelect' action='true'>
     <Property name='title' value='_FM_PRIVATE_RESOURCES'/>
     <Property name='availableTitle' value='_FM_AVAILABLE_RESOURCES'/>
     <Property name='selectedTitle' value='_FM_SELECTED_RESOURCES'/>
     <Property name='allowedValues'>
       <invoke name='getUnassignedResources' class='com.waveset.ui.FormUtil'>
           <ref>:display.session</ref>
           <map>
             <s>current</s>
             <ref>waveset.resources</ref>
           </map>
       </invoke>
    </Property>
   </Display>
</Field>
```

listResourceObject メソッドの使用

listResourceObject メソッドには 10 個のバージョンがあります。バージョンによっては、1 つのリソース ID を指定したり、リソースのリストを指定したり、キャッシュまたはその他のキャッシュ詳細をクリアするためのブール型を指定する必要があります。また、メソッドを別のユーザーとして実行するように指定できるバージョンもあります。

フォームまたは規則内でこのメソッドを実装するときは、このメソッドのどのバージョンを使用しているのかについてコメントで明らかにしてください。次に例を示します。

```
Lists the Groups on the AD-Austin resource starting at the OurGroups
OU. It will leverage the server cache should this list be found
there.

Additional details in the Identity Manager formUtil javadocs under:

public static java.util.List listResourceObjects(java.lang.String
subjectString,

java.lang.String objectType,
```

```
java.lang.String resourceId,

java.util.Map options,

java.lang.String cacheList)
```

FormUtil メソッドを使用するシナリオで注意が必要なもの

ほとんどの FormUtil 処理ではビューが使用されます。このため、フォームで使用されるビュー関連のメソッドでもっともよく使用されるのは、checkoutView メソッドと checkinView メソッドになります。

一般的なチェックアウト処理は次のようになります。

コード例 2-45

```
<Action id='-1' application='com.waveset.session.WorkflowServices'>
   <Argument name='op' value='checkoutView'/>
   <Argument name='type' value='User'/>
   <Argument name='id' value='mfairfield'/>
   <Variable name='view'/>
   <Return from='view' to='user'/>
</Action>
```

チェックアウトおよびチェックイン呼び出しとオプションのマップの使用

チェックアウトおよびチェックイン呼び出しのオプション引数として、どのオプションを使用できるかを判断するのが難しいことがあります。これらのオプション引数は、UserViewConstants クラスの一部として定義されます。Javadoc には、次の形式でオプションが記載されています。

```
OP_TARGETS

OP_RAW

OP_SKIP_ROLE_ATTRS
```

オプションを確認するときにこれらのリテラル文字列をコードにハードコードする代わりに、コードベース全体で使用できるように文字列を表現する定数を定義します。これはよいコーディング手法ですが、XPRESS やワークフローでは OP_TARGET_RESOURCES などの静的フィールドを参照できません。

正しい値を渡すことができる有効な文字列であることを調べるために、正しい文字列であることを確認するためのテスト規則を作成できます。たとえば、次の規則は TargetResources を返します。

コード例 2-46

```
<block>
   <set name='wf_srv'>
      <new class='com.waveset.provision.WorkflowServices'>
   </set>

   <script>
      var wfSvc = env.get( 'wf_srv' );
      var constant = wfSvc.OP_TARGETS;
      constant;
   </script>
</block>
```

この規則は文字列を調べるのに便利ですが、すべての呼び出しに同じ文字列を返すため、本稼働配備には適していません。

有効な文字列であることがわかったら、ビューのチェックアウト呼び出しを次のように更新できます。このコードでは、Identity Manager および AD に変更が反映されるだけのビューがチェックアウトされます。

コード例 2-47

```
<Action id='-1' application='com.waveset.session.WorkflowServices'>
   <Argument name='op' value='checkoutView'/>
   <Argument name='type' value='User'/>
   <Argument name='id' value='mfairfield'/>
   <Argument name='options'>
      <map>
       <s>TargetResources</s>
       <list>
          <s>Lighthouse</s>
          <s>AD</s>
       </list>
     </map>
   </Argument>
<Variable name='view'/>
<Return from='view' to='user'/>
</Action>
```

ベストプラクティス

パフォーマンス上の観点から、可能な場合はユーザービューのサイズを制限することをお勧めします。ビューを小さくすると、リソースから取得されてネットワークに送信されるデータが少なくなります。たとえば、ユーザーが特定のリソースへのアクセスを要求するカスタムワークフローを顧客が実装することにした場合、そのワークフローは、(適切な承認が得られるまでの間) 変更を送信できるようにユーザービューをチェックアウトするはずです。この例では、取得する必要がある情報はビューの Identity Manager ユーザー部分だけで、`waveset.resources` リストおよび `accountInfo` オブジェクトがそれに合わせて更新されるようです。このような場合は次のようにして、ユーザービューをチェックアウトするときに TargetResources オプションを使用して、ユーザービューの Identity Manager ユーザー部分とオプションマップだけがチェックアウトされるようにします。

コード例 2-48

```
<map>
   <s>TargetResources</s>
   <list>
      <s>Lighthouse</s>
   </list>
</map>
```

追加オプション

FormUtil メソッドのサブセットでは、次のオプションが使用されます。

- *scopingOrg*
- *conditions*
- *current*

scopingOrg

このオプションは、ユーザーに複数の AdminRole が割り当てられているときに使用します。このオプションの値には、組織名を指定するようにしてください。この値を指定することによって、AdminRole によって制御される組織が利用できる名前だけが返されるようになるはずです。AdminRole は、scopingOrg 組織を制御する必要があり、ログイン中のユーザーに割り当てられます。

このオプションは、通常、ユーザーが別のユーザーを作成または編集するときに割り当てる名前 (たとえば、Resourcenames) が、必ず編集対象ユーザーが属す組織に適した名前となるようにするために使用されます。

scopingOrg パラメータの使用

次のような状況では、この属性を設定します。

- 指定されるユーザーに、複数の AdminRole が割り当てられている
- 管理者がユーザーを作成または編集するときに割り当てるタイプのオブジェクトの名前が、必ず作成 / 編集対象ユーザーが属す組織に適した名前となるようにする

次に例を示します。

- 管理者に Engineering AdminRole と Marketing AdminRole が割り当てられている
- その管理者が Engineering 組織に属しているユーザーを編集する

このような場合、そのユーザーに割り当てることができるリソースは、Engineering AdminRole によって制御される組織が利用できるリソースに制限されます。

scopingOrg 属性の実装

前述の動作を実装するには、ユーザーフォームの `waveset.resources` フィールドに `scopingOrg` 属性を追加します。

現在の組織で使用できる値を参照する方法は次のとおりです。

コード例 2-49

```
<Field name='waveset.resources'>
   <Display class='MultiSelect'>
      <Property name='title' value='_FM_PRIVATE_RESOURCES'/>
      <Property name='availableTitle'
          value='_FM_AVAILABLE_RESOURCES'/>
      <Property name='selectedTitle' value='_FM_SELECTED_RESOURCES'/>
      <Property name='allowedValues'>
         <invoke class='com.waveset.ui.FormUtil'
           name='getUnassignedResources'>
             <ref>:display.session</ref>
               <map>
                  <s>currentRoles</s>
                  <ref>waveset.roles</ref>
                  <s>currentResourceGroups</s>
                  <ref>waveset.applications</ref>
                  <s>current</s>
                  <ref>waveset.original.resources</ref>
                  <s>scopingOrg</s>
                  <ref>waveset.organization</ref>
               </map>
         </invoke>
      </Property>
   </Display>
</Field>
```

current

返される値とマージされる名前のリストを指定します。たとえば、選択されたすべての名前が必ず MultiSelect のリストから選択できるようにするときは、通常、MultiSelect フィールドで選択される名前のリストを指定します。

conditions

この値には、特定の属性、それらの期待値、および比較演算子が指定された AttributeCondition の組み合わせを設定します。AttribueCondition は、3 つの方法で指定できます。

表 2-23　conditions 属性の値

値の形式	説明
Map	複数の <MapEntry> 要素で構成されるマップ。各 <MapEntry> 要素には、*key* に対応する属性名と、*value* に対応する値を指定します。演算子は「equals」であるとみなされます。属性名と値の複数のペアを指定した場合は、それらは論理 and で連結されます。 例 <Map> <MapEntry key='memberObjectGroups' value='Top'/> </Map>
map	<MapEntry> を含まないマップ。最初のエントリは、このオブジェクトタイプがサポートする、クエリー可能な属性の名前です。2 番目のエントリは、関連付けられているクエリー可能属性を取得するために必要な、このタイプのオブジェクトの値です (演算子は「equals」と見なされる)。 属性名と値の複数のペアを指定した場合は、それらは論理 and で連結されます。 例 <map> <s>memberObjectGroups</s> <ref>waveset.organizations</ref> </map>

表 2-23　conditions 属性の値 (続き)

値の形式	説明
list	AttributeCondition オブジェクトのリスト。複数の AttributeCondition を指定した場合は、それらは論理 and で連結されます。「equals」以外の演算子を指定する必要がある場合は、この形式を使用する必要があります。 例 <list> <newclass= 'com.waveset.object.AttributeCondition'> <s>MemberObjectGroups</s> <s>equals</s> <ref>waveset.organization</ref> </new> </list>

conditions 属性の使用

オブジェクトタイプに固有の、1 つまたは複数のクエリー属性条件のリストを指定して、特定の FormUtil メソッドから返される名前のリストをフィルタリングできます。これらのメソッドには、引数として options マップを受け付けるメソッドが含まれます。これらのクエリー属性条件を、conditions をキーとしたクエリーオプションとして指定することができます。オプションの値にはマップ、または AttributeConditions のリストを指定できます。

例 : condition 属性による名前のフィルタリング

次の例は、conditions 属性を使用して、オプションマップを引数として受け付ける FormUtil メソッドから返される名前のリストに、追加のフィルタを適用する方法を示しています。この例では、conditions を使用して、タイプ LDAP のコンテナオブジェクトグループをサポートするリソースだけが返されるように指定します。

コード例 2-50　conditions 属性の使用の例 1

```
<Field name='waveset.resources'>
   <Display class='MultiSelect' action='true'>
 ...
     <Property name='allowedValues'>
        <invoke class='com.waveset.ui.FormUtil'
         name='getUnassignedResources'>
```

コード例 2-50　　conditions 属性の使用の例 1

```
            <ref>:display.session</ref>
                <map>
                <s>currentRoles</s>
                <ref>waveset.roles</ref>
                <s>currentResourceGroups</s>
                <ref>waveset.applications</ref>
                <s>current</s>
                <ref>waveset.original.resources</ref>
                <s>conditions</s>
                  <map>
                <s>supportsContainerObjectTypes</s>
                <s>true</s>
                <s>type</s>
                <s>LDAP</s>
                  </map>
                </map>
        </invoke>
      </Property>
   </Display>
</Field>
```

この 2 番目の例では、名前が *ldap* で始まるリソースのうち、コンテナオブジェクトをサポートするリソースを要求します。クエリー可能属性の値を比較するときは、大文字と小文字が区別されます。

コード例 2-51　　conditions 属性の使用の例 2

```
<Field name='orgResource'>
   <Display class='Select' action='true'>
       ...
      <Property name='allowedValues'>
        <invoke class='com.waveset.ui.FormUtil'
           name='getResourcesSupportingContainerObjectTypes'>
           <ref>:display.session</ref>
             <map>
                <s>conditions</s>
                <list>
                   <new class='com.waveset.object.AttributeCondition'>
                      <s>name</s>
                      <s>starts with</s>
                      <s>ldap</s>
                   </new>
                </list>
             </map>
        </invoke>
      </Property>
   </Display>
</Field>
```

コード例 2-52　　conditions 属性の使用の例 3

```
<Field name='accounts[Lighthouse].capabilities'>
   <Display class='MultiSelect'>
         ...
     <Property name='allowedValues'>
        <invoke class='com.waveset.ui.FormUtil'
           name='getUnassignedCapabilities'>
           <ref>:display.session</ref>
           <ref>waveset.original.capabilities</ref>
             <map>
             <s>conditions</s>
               <list>
                  <new class='com.waveset.object.AttributeCondition'>
                    <s>name</s>
                    <s>starts with</s>
                    <s>bulk</s>
                  </new>
               </list>
             </map>
        </invoke>
     </Property>
   </Display>
</Field>
```

サポートされるクエリー可能属性の名前

サポートされるクエリー可能属性の名前は、オブジェクトタイプごとに次のように分類されます。

上記以外のクエリー可能属性の名前は Identity Manager のスキーマ設定設定オブジェクトに定義されます (たとえば、`firstname` と `lastname`)。

サポートされる演算子

Identity Manager では、クエリー可能属性のすべての比較が大文字と小文字を区別する方式で実行されます。また、Identity Manager では、文字列を比較する方式を使用して比較が実行されるため、1000<999 になります。これは、文字列比較では文字単位で比較されるので、9 が 1 よりも大きくなるためです。

- `equals` または `is equal`
- `notEquals` または `is 'not equal'`
- `greaterThan` または `'greater than'`
- `greaterThanOrEqualTo` または `'not less than'`
- `lessThan` または `'less than'`

- lessThanOrEqualTo または‘not greater than’
- startsWith または‘starts with’
- endsWith または‘ends with’
- contains または‘contains'
- isPresent または exists
- ‘notPresent'
- isOneOf または is one of'
- containsAll

第 3 章

Identity Manager のビュー

この章では、Identity Manager で使用されるデータ構造、Sun™ Identity Manager ビューについて説明します。Identity Manager のワークフロー、フォーム、および参照情報にビューを実装する方法など、ビューに関する背景情報を提供します。

Identity Manager IDE を使用すれば、Identity Manager ビューやその他の汎用オブジェクトについてさらに詳しく調べることができます。Identity Manager IDE をインストールおよび設定する手順については、https://identitymanageride.dev.java.net を参照してください。

この章の内容

この章は、次の各節で構成されています。

- IDM ビューについて
- ユーザービューについて
- 代表的なビュー
- ビューオプション
- deferred 属性
- ビューの拡張

Identity Manager ビューについて

Identity Manager ビューは、Identity Manager によって管理される 1 つまたは複数のオブジェクトから構成される属性の集合です。ビューは一時的、かつ動的なもので、リポジトリには保存されません。ビューの表示を更新すると、ビューに含まれるデータは、新しいロールやリソースの割り当てに応じて変更されます。

Identity Manager の使用中は、主にフォームやワークフローにビューが表示されます。Identity Manager フォームは、編集時にビュー属性をどのようにブラウザに表示するかを表現するオブジェクトです。また、フォームには、表示対象属性から非表示属性をどのように算出するかを決める規則も設定されています。ワークフロープロセスは、論理的で反復可能な一連のアクティビティーで、ある関与者から別の関与者にドキュメント、情報、またはタスクが渡されると、一連の手順規則に従ってアクションを実行します。

ビューを操作するときは、まず、次の内容を理解してください。

- ビューの一般的な概念
- Identity Manager でのビューの用途
- ビューのカスタマイズ頻度

ビューとは

もっとも重要なビューは、Identity Manager に格納されているユーザー属性と、Identity Manager が管理するアカウントから読み込まれた属性が含まれるユーザービューです。ユーザービューの一部の属性は、Identity Manager のユーザーインタフェースと管理者インタフェースで使用されるフォームに表示されます。それ以外の属性は、非表示であるか、読み取り専用です。通常、非表示の属性は、別の表示対象属性の導出や、フィールド値の計算に使用されます。

たとえば、ユーザーを作成する際 (ユーザービューとして表される)、管理者は、「ユーザーの作成」ページの適切なフォームフィールドに姓、名を入力します。管理者がフォームを保存すると、システムは姓と名を連結し、非表示フィールドにユーザーのフルネームを導出できます。このフルネームは、Identity Manager を含め、1 つまたは複数のリソースに保存できるようになります。承認を受けると (承認が必須とされる場合)、システムはユーザービューを Identity Manager リポジトリ内の 1 つまたは複数のオブジェクトに戻し、ユーザーに割り当てられているリソースにビューを送信して、ユーザーのリソースアカウントを作成または更新します。

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。

ブール型のすべての属性は、ビューから削除されます。値が削除された属性は、論理的に false と見なされます。

ビューハンドラとは

ビューハンドラは、ビューの作成と指定アクションの実行に必要なロジックを含む Java クラスで、ビューの属性設定によって指定されます。ビューハンドラには、対話的なフォームの利便性を向上させるための情報が含まれる場合もあります。ビューがチェックインされると、ビューハンドラはビュー属性を読み取り、それをリポジトリオブジェクトに対する操作に変換します。承認やプロビジョニングなど、より複雑なタスクの実行では、ビューハンドラはワークフローを頻繁に呼び出します。ビューハンドラが、あるユーザーを対象に処理を実行する場合、そのユーザーに対してすでに実行中のワークフローが存在するときは、ほとんどのビューハンドラはビューへのチェックインを回避します。

ビューとフォーム

Identity Manager フォームには、ビューのデータを変換する規則と、ブラウザでビュー属性をどのように表示、変更するかを指定する規則が含まれます。Identity Manager のユーザーインタフェースは、ビューとフォームを処理して HTML フォームを生成します。ユーザーが HTML フォームを送信すると、Identity Manager は、送信された値をビューにマージし、ビューの更新をビューハンドラに要求します。対話的な編集セッションでは、ビューの表示が数回更新されることがあり、フォームに設定されているロジックに基づいて、異なる HTML フィールドを生成できます。ユーザーが対話的な操作を完了すると、ビューはチェックインされ、多くの場合、ワークフロープロセスへの入力としてそのビューが渡されます。

ビューとワークフロー

多くの場合、ビューをチェックインすると、ビューに指定された変更を完了するために、新しいワークフロープロセスが呼び出されます。ワークフローは、時間がかかるタスクをバックグラウンドで実行したり、承認プロセスを呼び出したりすることができます。また、リソースをクエリーしたり、必要に応じて適切なタスクを実行したりすることもできます。承認時は、管理者はビューの内容を確認し、必要に応じて変更

を加えることができます。承認されたビューの属性は、1 つまたは複数のリポジトリオブジェクトの変更に変換されます。ユーザーに関連するビューでは、選択されたリソースアカウントに変更を反映させるためにプロビジョニングが行われる場合があります。

アカウントタイプとユーザー用ビュー

アカウントタイプをユーザーに割り当てると、Identity Manager でそのアカウントタイプと `accountId` を使用できるようになります。User、Enable、Disable、Deprovision などのユーザー用のビューを操作するときは、対処方法を示した次のガイドラインに従ってください。

- デフォルトタイプのアカウントを示すときは、NULL 値を使用します。デフォルトタイプのアカウントを参照するときは、accounts[corp-ad] などのリソース名を使用します
- 特定のタイプのアカウントを参照するときは、リソース名の代わりにタイプ修飾した名前を使用します。タイプ修飾したリソース名の形式は次のとおりです。

  < リソース名 >|< アカウントのタイプ >

  リソース corp-ad の `Admin` タイプのアカウントのアカウントデータを参照するには、`accounts[corp-ad|Admin]` と指定します。

代表的なビュー

カスタマイズされたフォームとワークフローの両方でよく使われるビューは次のとおりです。

`User`	Identity Manager ユーザーの操作とリソースアカウントのプロビジョニングに使用されます。
AccountCorrelation (アカウント相関)	指定されたアカウント (またはアカウント属性) と相互に関連するユーザーの検索に使用されます。
AdminRole	ユーザーに管理者ロールを割り当てるときに使用されます。
Enable	無効化するリソースアカウントのリストの表示と選択に使用されます。
Deprovision	プロビジョニング解除するリソースリストの表示と選択に使用されます。
Disable	有効化するリソースアカウントのリストの表示と選択に使用されます。

ChangeUserAnswers (ユーザー回答変更)	ユーザーの秘密質問に対する回答の変更に使用されます。
ChangeUserCapabilities (ユーザー機能変更)	Identity Manager ユーザーの機能を変更するときに使用されます。
List	Identity Manager ユーザーインタフェースの作業項目とプロセスのリストの生成に使用されます。
Org (組織)	作成する組織のタイプと、それを処理するオプションのタイプの指定に使用されます。
Password	Identity Manager ユーザーのパスワードの変更に使用されます。また、オプションとして、パスワードをリソースアカウントに反映させます。
Process (プロセス)	ワークフローやレポートなどのタスクの呼び出しに使用されます。
Reconcile (調整)	調整処理の要求またはキャンセルに使用されます。
ReconcileStatus (調整状態)	最後に要求された調整処理の状態を取得するときに使用されます。
RenameUser	Identity Manager ユーザーとリソースアカウントアイデンティティーの名前の変更に使用されます。
Reprovision (再プロビジョニング)	再プロビジョニングするリソースのリストの表示と選択に使用されます。
ResetUserPassword	管理者がパスワードをランダム生成のパスワードにリセットするときに使用されます。また、オプションとして、新しいパスワードをリソースアカウントに反映させます。
Resource	リソースの操作に使用されます。
`ResourceObject` （リソースオブジェクト）	たとえば、グループやメーリングリストのように、リソースによってサポートされる任意のオブジェクトの操作に使用されるビューのファミリーです。
Role	作成する Identity Manager ロールのタイプの指定に使用されます。
TaskSchedule (タスクスケジュール)	TaskSchedule オブジェクトの作成と変更に使用されます。
Unlock (ロック解除)	リソース側の機能としてアカウントのロックがサポートされている場合、それらのリソースに対するアカウントのロック解除に使用されます。
WorkItem	ワークフロー承認フォームの記述に使用されます。
WorkItemList (作業項目リスト)	リポジトリ内の作業項目の集合に関する情報の表示と、複数の作業項目に対する処理の同時実行に使用されます。

ユーザービューについて

ユーザービューは、Identity Manager ユーザーに関する次のような情報を含む属性の集合です。

- Identity Manager の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、組織など、ほかのソースから取得される情報

フォームでもっとも頻繁に使用されるユーザービューは、ユーザーを作成または編集するページで使用するように設計されています。これらのページは、変更されたユーザービューを格納したワークフロープロセスを呼び出し、必要時には、変更されたビューの情報を Identity Manager と関連リソースに戻します。ユーザービューがワークフロープロセスに格納されている間、ワークフロープロセスはワークフローアクションによって属性値を操作できます。ワークフローは、手動アクションや承認フォームによるユーザー入力のために、属性値を表示することもできます。

ユーザービューとフォームの統合

多くの場合、ユーザービューはフォームと組み合わせて使用されます。フォームには、HTML フィールドによってデータをどのように表示するかを制御する規則と、フォームを表示する HTML ページの送信後にデータをどのように処理するかを制御する規則が含まれます。フォーム定義とビューは、フォームジェネレータ (フォーム生成ツール) というシステムコンポーネントによって組み合わされ、ブラウザに表示される HTML ページが生成されます。

フォーム内の HTML コンポーネントに割り当てることで、ビュー属性の値を表示できます (ビュー属性の表示方法の詳細については、第 6 章「HTML 表示コンポーネント」を参照)。

ビューは、`GenericObject` クラスのインスタンスとして実装されます。このクラスは、名前と値のペアを表示するメカニズムと、パス式によってオブジェクトの複雑な階層を切り替えるユーティリティーを提供します。パス式は、オブジェクト階層を切り替えて属性の値を取得または割り当てるために、実行時に解釈される文字列です。

有効なフォームフィールド名を割り当てるには、パス式の記述方法を理解する必要があります。パス式の使用の詳細については、「パス式」の節を参照してください。

ユーザービューとワークフローの統合

ユーザービューが含まれるワークフロープロセスの多くは、そのビューを user というワークフロー変数に格納します。ユーザービューのパスに user というプレフィックスを付けることで、ワークフローの式でビューを参照できます。たとえば、user.waveset.accountId のように指定します。この waveset という文字列は、それ自体がユーザービューオブジェクトに属する waveset というオブジェクトを指し、そのオブジェクトに属する属性の accountId を識別します。

ビュー用に記述される承認フォームは、WorkItem (作業項目) ビューと呼ばれます。作業項目ビューでは、すべてのワークフロー変数が、デフォルトで variables という属性に格納されます。ユーザービューを含むワークフロー用に記述された承認フォームでユーザービューの属性を参照するには、variables.user. というプレフィックスを使用します。たとえば、variables.user.waveset.roles のように指定します。詳細については、この章の「作業項目ビュー」を参照してください。

汎用オブジェクトクラス

高次では、オブジェクトは、名前と値のペアから構成される、名前が付けられた単なる属性の集合に過ぎません。属性の値は、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として使用できます。ほとんどすべてのオブジェクトは、Map、List、String の各 Java クラスを使用して抽象的に表現できます。

Identity Manager システムでは、任意のオブジェクトと集合を表現する単純なメモリーモデルとして GenericObject クラスが提供されます。これには、オブジェクト階層を簡単に移動して、属性値を表示、変更するための機能も含まれます。

GenericObject クラスは、java.util.Map インタフェースを実装し、java.util.HashMap を内部的に使用して、名前と値のペアの集合を管理します。このマップのエントリは、属性と呼ばれます。属性の値には、XML として直列化できる任意の Java オブジェクトを指定できます。GenericObject のもっとも一般的な属性値は次のとおりです。

次のクラスのインスタンスは次のとおりです。

- String
- Integer
- Boolean
- EncryptedData
- List
- Date

- GenericObject
- X509cert

属性値として List または GenericObject を割り当てることで、オブジェクトの複雑な階層を構築できます。属性値の割り当てが完了したら、その階層を切り替えて、属性の値にアクセスします。

パス式

パス式は、オブジェクト階層を切り替えて属性の値を取得または割り当てるために、GenericObject クラスによって実行時に解釈される文字列です。Identity Manager は、ドットと括弧を組み合わせて、階層内のオブジェクトと属性を表現します。

フォーム (たとえば、<Field name='user.waveset.roles'/>) をカスタマイズするときは、フォームフィールドの name 属性の値としてパス式を指定します。

オブジェクトをたどる

次の例は、2 つの属性が指定された簡単な GenericObject を示しています。

- name (String)
- address (GenericObject) address オブジェクトには、street という文字列の属性があります。

address オブジェクトの street 属性を示すパス式を作成するときは、address.street と指定します。

パス式では、あるオブジェクトから別のオブジェクトへの切り替えをドット文字 (.) で表します。これは、Java で使用されるドット、または C で使用される「->」演算子と似ています。次の例のように、パスが長くなることもあります。

```
user.role.approver.department.name
```

リストの切り替え

パス式を使用して、リストの値を切り替えることもできます。java.util.List という値が指定された children という属性を持つオブジェクトを考えてみましょう。リストを構成する各オブジェクトは、それ自体が name 属性と age 属性を持つ GenericObject です。最初の子の名前を示すパスは、次のように記述します。

```
children[#0].name
```

パス式では、角括弧を使用してリストのインデックスを表します。括弧内のトークンは、インデックス式と呼ばれます。単純な用例では、要素の位置によってリストのインデックスを表す正の整数が指定されます。

通常は、リスト内のオブジェクトの位置は任意です。インデックス式では、簡単な検索条件を指定して、リスト内のオブジェクトを特定することもできます。リスト内のオブジェクトは、多くの場合、name 属性を持ちます。これは、リスト内のオブジェクトを一意に識別する属性です。パス式は、インデックス式に指定された、オブジェクトを示す name 属性の暗黙的な参照をサポートします。

次に例を示します。

```
children[hannah].age
```

このパス式は、children 属性に格納されたオブジェクトのリストを取得します。このリストの検索は、hannah という値を持つ name 属性のオブジェクトが見つかるまで継続されます。一致する属性が見つかると、Identity Manager は age 属性の値を返します。

例 : = 演算子の使用

```
<ref>accountInfo.accounts[type=vms].name</ref>
```

accountInfo.accounts[type=vms].name は、VMS リソースの名前のリストを返します。要素が 1 つしか存在しない場合は、1 つの要素だけのリストが返されます。

== 演算子の使用

children[hannah].age は children[name==hannah].age と等価です。たとえば、type=LDAP を使用して検索すると、LDAP リソースの名前のリストが返されます。しかし、== 演算子を使用した場合、1 つのオブジェクトだけが返されます。たとえば、children[parent=hannah].occupation は hannah のすべての子の職業を返しますが、children[parent==hannah].occupation はリストではなく最初に見つかった子の職業のみを返します。

例

```
<index i='0'>

<    ref>accountInfo.accounts[type=vms].name</ref>

</index>
```

これは、次の式と等価です。

```
<ref>accountInfo.accounts[type==vms].name</ref>
```

タイプが vms のアカウントが複数存在する場合は、どちらの例でも最初に見つかったアカウントが返されます。特定の順番のものが返される保証はありません。

リストの計算

オブジェクトに格納されていない List 値を計算するパス式も記述できます。たとえば、次のようにします。

```
accounts[*].name
```

インデックス式に指定されたアスタリスクは、リスト内の各要素に対する繰り返しを表します。式の結果は、リスト内の各要素に残りのパス式を適用した結果のリストとなります。前述の例では、結果は `String` オブジェクトのリストとなります。文字列は、`accounts` リスト内の各オブジェクトの `name` 属性から取得されます。

アスタリスク (*) を持つパス式は、フィールドの集合を複製するフォームで、`FieldLoop` 構成体と組み合わせて使用されます。

アカウントタイプとユーザー用ビュー

アカウントタイプをユーザーに割り当てると、Identity Manager でそのアカウントタイプと `accountId` を使用できるようになります。User、Enable、Disable、Deprovision などのユーザー用のビューを操作するときは、対処方法を示した次のガイドラインに従ってください。

- デフォルトタイプのアカウントを示すときは、NULL 値を使用します。デフォルトタイプのアカウントを参照するときは、accounts[corp-ad] などのリソース名を使用します
- 特定のタイプのアカウントを参照するときは、リソース名の代わりにタイプ修飾した名前を使用します。タイプ修飾したリソース名の形式は次のとおりです。

  `< リソース名 >|< アカウントのタイプ >`

  リソース corp-ad の `Admin` タイプのアカウントのアカウントデータを参照するには、`accounts[corp-ad|Admin]` と指定します。

ユーザービューの属性

Web ブラウザからユーザーアカウントを作成または変更するときは、常にユーザービューを間接的に操作していることになります。ユーザーのアカウント情報の変更という点では、これは Identity Manager システムでもっとも重要なビューであるともいえます。

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。

その他のビューと同様に、ユーザービューも属性セットを持つ GenericObject として実装されます。ルートオブジェクト内の属性の値は、GenericObjects 自体です。属性は、入れ子構造にすることができます。

ユーザービューは、次の表に示される属性を持ちます。各属性については、後の節で詳しく説明します。

表 3-1 ユーザービューの最上位属性

属性	説明
waveset	Identity Manager リポジトリ (WSUser オブジェクト) に格納されている情報を持ちます。このビューは、基本ビューとも呼ばれます。
accounts	リソースからフェッチされたすべてのリソースアカウント属性の値を持ちます。通常、これらの値はフォームで編集される値です。
accountInfo	ユーザーに関連付けられているリソースとアカウントに関する読み取り専用の情報を持ちます。
display	インタフェースの実行時状態に関する読み取り専用の情報を持ちます。これは、ユーザーの対話的な編集のみに使用されます。display.session は、ログイン情報とアクセス情報を表します。display.subject は、ユーザーがログインに使用しているアカウントを識別します。display.eventType は、ユーザービューが作成処理と更新処理のどちらに使用されているかを表します。
global	すべてのリソースアカウントの間で同期される属性を持ちます。
password	ユーザーのパスワード、パスワードの有効期限、およびターゲットシステムに固有の属性値を持ちます。

フォームを設計するとき、通常は、ユーザービューオブジェクトである waveset、global、および account 属性 (たとえば、global.firstname) へのパスがフィールド名となります。

適切な変数ネームスペースの選択

アカウント関連の情報を導出するために、ユーザービューにはいくつかのネームスペースがあります。次の表は、変数ネームスペースの概要を示しています。

表 3-2　アカウント関連のユーザービュー属性

アカウント関連のネームスペース	説明
waveset.accounts	チェックイン処理の実行中に差異を検出するために、内部的に使用されます。これには、すべてのアカウント属性の開始値が含まれます。この値を変更しないでください。
accountInfo.accounts	ユーザー、およびその関連リソースへのリンクを持つアカウントに関する、導出された読み取り専用の情報。この属性はフォームで使用できますが、変更は行わないでください。
accounts	アカウント属性の読み取り / 書き込みコピーを保持します。更新可能なフィールドは、このネームスペースを指すようにします。
global	グローバル属性のコピーを保持します。このエリアの値は、フォームがグローバルフィールドを定義する場合、または特別な MissingFields 参照を使用している場合にのみ表示されます (グローバル属性をどのように処理するかは、フォームによって決定される)。 ワークフローにグローバル属性を設定するときは、フォームにグローバルフィールドも定義してください。ビューにグローバルな値を蓄積するだけでは不十分です。

属性の参照

フォームでは、次の 2 つの方法で属性を参照できます。

- Field 要素の名前属性を使用して、属性の完全パス名を指定します。次に例を示します。

```
<Field name='waveset.accountId'>
```

フォームフィールドへの Field 要素の名前属性の設定の詳細については、「Identity Manager フォーム」の章を参照してください。

- 別のフィールドから属性を参照します。

```
<Expansion>
   <concat>
       <ref>global.firstname</ref><s> </s>
      <ref>global.lastname</ref>
```

```
 <Expansion>
    </concat>
 </Expansion>
```

ワークフローでは、プロセス変数 (ワークフローエンジンが認識できる変数)、またはアクションと遷移の XPRESS ステートメントとして Field 属性を参照できます。ワークフローでこれらの属性を参照するときは、ビューが格納されているワークフロービューの名前をプレフィックスとしてパスに追加します (たとえば、user.waveset.accountId)。

一時的な値を持つ属性

ユーザービューの最上位に、一時的な値を格納するフィールドを定義できます。これらのフィールドの値は、メモリー内でユーザービューが有効な間 (通常はプロセス終了までの間) は存続しますが、Identity Manager リポジトリに格納されることも、リソースアカウントに反映されることもありません。

たとえば、電話番号の値は、3 つのフォームフィールドの値を連結した結果です。次の例では、p1 は市外局番、p2 と p3 は電話番号の残りの部分をそれぞれ参照します。これらの値は連結され、global.workPhone というフィールドに格納されます。リソースに反映する値は連結された電話番号のみであるため、このフィールドにのみ global というプレフィックスが追加されます。

一般に、最上位フィールドの構文は次の場合に使用されます。

- フィールドの値を Identity Manager またはその他のリソースに渡さない
- フィールドが、電子メール通知、または別のフィールドの計算のみに使用される

次のレベルに渡されるすべてのフィールドには、前述の「ユーザービューの属性」の表で説明した、いずれかのパスプレフィックスを付ける必要があります。

```
Field name='p1' required='true'>
   <Display class='Text'>
      <Property name='title' value='Work Phone Number'/>
      <Property name='size' value='3'/>
      <Property name='maxLength' value='3'/>
   </Display>
</Field>
<Field name='p2' display='true' required='true'>
   <Display class='Text'>
      <Property name='rowHold' value='true'/>
      <Property name='noNewRow' value='true'/>
      <Property name='size' value='3'/>
      <Property name='maxLength' value='3'/>
   </Display>
</Field>
<Field name='p3' display='true' required='true'>
   <Display class='Text'>
      <Property name='rowHold' value='true'/>
      <Property name='noNewRow' value='true'/>
      <Property name='size' value='4'/>
      <Property name='maxLength' value='4'/>
   </Display>
</Field>
<Field name='global.workPhone' required='true' hidden='true'>
   <Expansion>
      <concat>
         <ref>p1</ref>
         <s>-</s>
         <ref>p2</ref>
         <s>-</s>
         <ref>p3</ref>
      </concat>
   </Expansion>
</Field>
```

waveset 属性

waveset 属性セットは、Identity Manager リポジトリ内の WSUser オブジェクトに格納されている情報を持ちます。この属性セットの下に入れ子になった一部の属性は、フォーム内での直接的な操作には使用されず、ビュー内の WSUser オブジェクトのすべての情報を Identity Manager が完全に表現できるように指定されます。

頻繁に使用される属性

新規ユーザーを作成するときに、すべての属性が必要となるわけではありません。次のリストは、作成または編集でよく使用される waveset 属性を示しています。一部の属性は読み取り専用ですが、その値は、別の属性の値の計算に使用されます。この表の後の節で、すべての waveset 属性について説明します。

表 3-3　頻繁に使用される waveset 属性の属性 (ユーザービュー)

属性	編集の可能性	データ型
waveset.accountId	読み取り / 書き込み	String
waveset.applications	読み取り / 書き込み	String
waveset.correlationKey	読み取り / 書き込み	String
waveset.creator	読み取り専用	String
waveset.createDate	読み取り専用	String
waveset.disabled	読み取り / 書き込み	String
waveset.email	読み取り / 書き込み	String
waveset.exclusions	読み取り / 書き込み	List
waveset.id	読み取り	String
waveset.lastModDate	読み取り	String
waveset.lastModifier	読み取り	String
waveset.locked	読み取り	String
waveset.lockExpiry	読み取り / 書き込み	String
waveset.organization	読み取り / 書き込み	String
waveset.questions	読み取り / 書き込み	List
waveset.resources	読み取り / 書き込み	List
waveset.resourceAssignments	読み取り / 書き込み	List
waveset.roleInfos	読み取り / 書き込み	List
waveset.roles	読み取り / 書き込み	String
waveset.serverId	読み取り / 書き込み	String

waveset.accountId

Identity Manager ユーザーオブジェクトの表示名を指定します。この属性は、ユーザー作成時に設定します。ユーザーの作成後にこの属性の設定を変更すると、Identity Manager アカウントの名前変更が開始されます。

ユーザー名の変更については、『Identity Manager 管理ガイド』を参照してください。

waveset.applications

ユーザーに直接割り当てられている各アプリケーションの名前のリスト (Identity Manager のユーザーインタフェースではリソースグループ) を持ちます。ロールを通じてユーザーに割り当てられるアプリケーションは、これに含まれません。

waveset.attributes

Identity Manager リポジトリ内の `WSUser` オブジェクトに格納されている任意の属性の集合。`waveset.attributes` 属性の値は NULL か、または別のオブジェクトです。このオブジェクトに格納されている属性の名前は、*Extended User Attributes* というシステム設定オブジェクトによって定義されます。拡張された属性の例としては、`firstname`、`lastname`、`fullname` が一般的です。これらの属性は、次の方法で参照できます。

```
waveset.attributes.fullname
```

または

```
accounts[Lighthouse].fullname
```

通常は、`waveset.attributes` 属性の値を変更することはありません。その代わりに、`accounts[Lighthouse]` 属性の値を変更します。属性が格納されると、`accounts[Lighthouse]` 内の値は格納前に `waveset.attributes` にコピーされます。`waveset.attributes` は、属性の元の値の記録に使用されます。システムは、ここに記録されている値と `accounts[Lighthouse]` に格納されている値を比較し、更新された概要レポートを生成します。ユーザー属性の拡張方法を示す例については、`account[Lighthouse]` 属性の説明を参照してください。

waveset.correlationKey

調整時およびユーザー検出時にユーザーを特定するために使用される、相互関係の値を持ちます。この属性は、通常は公開されませんが、直接編集できます。

waveset.creator

このユーザーを作成した管理者の名前を持ちます。

この属性は読み取り専用です。

waveset.createDate

このアカウントが作成された日時を持ちます。日時は、`MM/dd/yy HH:mm:ss z` という形式で表示されます。

例

```
05/21/02 14:34:30 CST
```

この属性は一度だけ設定され、それ以後は読み取り専用です。

waveset.disabled

Identity Manager ユーザーの無効化状態を持ちます。アカウントが無効化されると、この属性は論理 true に設定されます。メモリーモデルでは、ブール型のオブジェクトであるか、`true` または `false` の文字列です。フォームからアクセスするときは、文字列と見なされます。

この属性の設定を変更することで、Identity Manager ユーザーを有効または無効にすることができます。ただし、`global.disable` を使用するほうが一般的です。システムが、変数を認識するすべてのリソース (Identity Manager を含む) に変数の値を必ず適用するようにするときは、変数名に `global.` というプレフィックスを追加します。

この値が true になると、ユーザーは Identity Manager のユーザーインタフェースにログインできなくなります。

waveset.email

Identity Manager リポジトリに格納される、ユーザーの電子メールアドレスを指定します。これは、通常は、リソースアカウントに反映される電子メールアドレスと同じです。

この属性の変更は、Identity Manager リポジトリのみに適用されます。電子メールの各種値をリソース間で同期させるときは、`global.email` 属性を使用します。

この属性の設定は変更可能です。

waveset.exclusions

ロール、リソースグループ、またはディレクトリを通じてユーザーにリソースが割り当てられる場合でも、プロビジョニングから除外されるリソースの名前を指定します。

waveset.id

Identity Manager ユーザーオブジェクトのリポジトリ ID を特定します。Identity Manager にユーザーを作成すると、この属性は NULL 以外の値になります。この値を調べることで、ユーザーが作成中であるか、編集中であるかを確認できます。この属性は、フォーム内のロジックによって調べられます。この値を使用して、新規ユーザーの作成中であるか (`waveset.id` の値が NULL)、既存ユーザーアカウントの編集中であるか (`waveset.id` の値が NULL 以外) に応じて表示されるフィールドをカスタマイズできます。

例

次の例は、`waveset.id` の値が NULL であるかどうかを調べる XPRESS ステートメントを示しています。

```
<isnull><ref>waveset.id</ref></isnull>
```

waveset.lastModDate

最後に加えられた変更の日時を持ちます。この日時は、1970 年 1 月 (GMT) の深夜零時から経過したミリ秒数で表されます。この属性は、ユーザーアカウントを変更するたびに更新されます。

この属性は読み取り専用です。

waveset.lastModifier

このユーザーアカウントを最後に変更した管理者またはユーザーの名前を持ちます。

この属性は読み取り専用です。

waveset.locked

ユーザーがロックされているかどうかを示します。値が `true` の場合は、ユーザーはロックされています。

waveset.lockExpiry

ユーザーの Lighthouse アカウントポリシーに、ロックされたアカウントの有効期限日としてゼロ以外の値が設定されている場合に、ユーザーロックの有効期限がいつ切れるかを指定します。この属性の値は、判読可能な日時です。

waveset.organization

ユーザーが所属する組織 (または `ObjectGroup`) の名前を持ちます。新しい組織に関する十分な権限を持っている管理者は、この属性の設定を変更できます。

組織の変更は重大なイベントであるため、組織の元の値も `waveset.original` 属性に格納されます。格納された値は、後で実行する比較に使用できます。

waveset.original

`waveset` 属性のいくつかの重要な属性の変更前の値に関する情報を持ちます。ビューが作成されると、システムがこの値を設定し、その後も変更されません。システムは、概要レポートと監査ログレコードの作成時に、この情報を使用します。

`waveset` 属性の変更前のすべての属性値がここに保存されるわけではありません。変更の追跡が現在定義されている属性は次のとおりです。

- password
- role
- organization

これらの属性を参照するときは、属性名のプレフィックスとして waveset.original. を追加します (たとえば、waveset.original.role)。

password

Identity Manager ユーザーのパスワードを指定します。ビューが新規に作成された直後は、この属性は暗号化されたユーザーパスワードを持ちません。その代わりに、ランダムに生成された文字列が設定されます。

password 属性セットは、次の表に示される属性を持ちます。

表 3-4 password 属性の属性 (ユーザービュー)

属性	説明
password	設定するパスワードを指定します。
confirmPassword	設定するパスワードを確認します。パスワードは、password.password の値と一致する必要があります。
targets	パスワードを変更できるリソースのリストを指定します。
selectAll	パスワードをすべてのリソースに反映させることを表す、ブール型のフラグを指定します。
accounts[]	各リソースに関する情報を持つオブジェクトのリストを指定します。この属性は、次に説明する 2 つの属性を持ちます。
accounts[<resource>]. selected	ブール型。設定した場合は、そのリソースでパスワードの変更が必要であることを示します。
accounts[<resource>]. expire	ブール型。設定した場合は、パスワードの有効期限が切れることを示します。 ユーザーが自身のパスワードを変更する場合は、この属性は false に設定されます。ただし、管理者が別のユーザーのパスワードを変更する場合は、このフラグは true に設定されます。 管理者またはユーザー以外のプロキシアカウントがアカウントのパスワードを変更する場合に、パスワードの有効期限が切れないようにするには、次のように設定します。 accounts [<resource>].expire = <s>false</s> この設定は、次のように機能します。 パスワードの有効期限は切れません Identity Manager はパスワードの変更をユーザーに再強制しません

waveset.passwordExpiry

Identity Manager パスワードの有効期限が切れる日付を持ちます。ビューの新規作成時は、メモリー表現は java.util.Date オブジェクトとなります。ビューがフォームで処理されると、この属性の値は mm/dd/yy という形式の日付のテキスト表現を持つ Date オブジェクト、または String オブジェクトのいずれかとなります。

waveset.passwordExpiryWarning

ユーザーが Identity Manager のユーザーインタフェースにログインするときに、警告メッセージを表示し始める日付を持ちます。通常は、waveset.passwordExpiry の日付より前の、同じ形式 (mm/dd/yy) の日付を持ちます。

waveset.questions

そのユーザーに割り当てられる秘密の質問と、その回答に関する情報を持ちます。この属性の値は、waveset.questions 属性を要素として持つ List です。

waveset.questions 属性セットは、次の表に示される属性を持ちます。

表 3-5 waveset.questions の属性 (ユーザービュー)

属性	編集の可能性	説明
answer	読み取り / 書き込み	暗号化された、秘密の質問に対する回答
id	読み取り	秘密の質問に割り当てられる、システム生成の ID
name	読み取り	この質問を識別する名前
question	読み取り	秘密の質問の内容 (テキスト)

name 属性は格納されません。システムは、id を変換して名前を生成します。このようなプロセスが必要となるのは、質問 ID は通常は番号であり、パス式で配列のインデックスに使用される番号がオブジェクト名としてではなく、絶対インデックスと見なされるためです。

たとえば、waveset.questions[#1].question という式は、質問リストの 2 番目の要素を示します (リストインデックスの開始番号はゼロ)。ただし、リストに含まれる質問が 1 つのみで、その ID 番号が 1 である場合、ID はリストインデックスには適さない場合があります。リストの要素を確実に指定するために、システムは、Q という文字と、それに続く ID という構成の名前 (たとえば、Q1) をそれぞれの質問に付けます。これにより、waveset.questions[Q1].question というパスは、常に正しい質問を示すようになります。

waveset.resources

ユーザーに直接割り当てられる各リソースの名前のリストを持ちます。ロールやアプリケーションを通じてユーザーに割り当てられるリソースは、このリストには含まれません。この属性に追加できるのは、非修飾リソース名のみです。ユーザーに割り当てられているすべてのリソースを確認する方法については、accountInfo 属性の説明を参照してください。

waveset.resourceAssignments

割り当てられているリソースのリストを修飾します。この属性は、既存の waveset.resources 属性と並立します。この属性に含まれるすべてのリソースは、waveset.resources 属性には非修飾の状態で格納されます。ユーザーにデフォルト以外のタイプのアカウントのみが割り当てられている場合でも、リソースは waveset.resources に格納されます。

waveset.resource または waveset.resourceAssignments に新しい割り当てを追加することができ、ビューの表示を更新すると、リストは自動的に再同期されます。これにより、デフォルトタイプのアカウントの割り当てが追加されます。waveset.resourceAssignments には、修飾と非修飾の両方の形式のリソース名を追加できます。これにより、修飾子に基づいて指定されるタイプのアカウントが追加されます。

waveset.roleInfos

このユーザーに割り当てられるロールについての情報が含まれるオブジェクトのリストを持ちます。

表 3-6

属性	説明
approvalRequired	(ブール型) このオプションのロールに承認が必要かどうかを指定します。directlyAssigned の値が false、assignmentType が optional の場合は、このオプションのロールに対して承認が必要かどうかがこの値によって決まります。
assignedBy	ユーザーに割り当てられるロールのうち、どのロールがこのロールの親であるかを指定します。directlyAssigned が false の場合、この値は、直接割り当てられる 1 つ以上のロールのうち、このロールが割り当てられる原因になったロールの名前になります。
assignmentType	間接ロールがどのように割り当てられるかを指定します。directlyAssigned が false の場合、この値は required、conditional、または optional のいずれかになります。
directlyAssigned	(ブール型) このロールがユーザーに直接割り当てられるかどうかを指定します。

表 3-6　(続き)

属性	説明
events	このロールの name/date エントリをマップします。イベントがいつ処理されるかを定義します (アクティブ化する日付と非アクティブ化する日付など)。 • name - 許可される値は activate および deactivate です。activate はこのロールをプロビジョニングする日付、deactivate はこのロールをプロビジョニング解除する日付であることを示します。 • date - 関連付けられたイベントの日付。
info	(オブジェクト型) ユーザーに割り当てるロールの変更を決定するときに、表示するべきでないロール情報が入ります。このオブジェクトには次の属性を指定できます。 * typeDisplayName - ロールタイプの表示名 / メッセージキー * description - ユーザーが入力するロールの説明
name	ロール名を指定します。
type	ロール設定オブジェクトに定義されているロールタイプを指定します。有効なタイプは BusinessRole、ITRole、Application Role、Asset Role です。
state	ロールの割り当て状態を指定します。有効な値は assigned または pendingActivationDate です。追加のカスタム状態を定義することもできます。

waveset.roles

このユーザーに割り当てられるロールの名前を持ちます。新しいロールに関する十分な権限を持っている管理者は、この属性の設定を変更できます。

ロールの変更は重大なイベントであるため、ロール属性の元の値も元のビューに格納されます。格納された値は、後で実行する比較に使用できます。

waveset.serverId

1 つの物理サーバー上の 1 つのリポジトリを指す複数の Identity Manager インスタンスが配備に含まれる場合に、一意のサーバー名を設定するために使用されます。詳細については、『Identity Manager インストール』を参照してください。

accounts 属性

accounts 属性は、Identity Manager ユーザーにリンクされた各アカウントのオブジェクトのリストを持ちます。各アカウントオブジェクトは、リソースから取得したアカウント属性の値を持ちます。

各アカウントオブジェクトの名前は、通常は、関連付けられているリソースの名前です。特定のリソースに複数のアカウントが関連付けられている場合は、オブジェクト名に |n という形式のサフィックスが付けられます。この *n* は整数を表します。リソースの最初のアカウントには、サフィックスは付けられません。2 番目のアカウントのサフィックスは |2 となります。リソースの 3 番目のアカウントには |3 というサフィックスが付けられ、以後、同様に続きます。

たとえば、Profile というアカウント属性を定義する、Active Directory というリソース名がある場合、この属性を示すビューパスは次のようになります。

```
accounts[Active Directory].Profile
```

このビューパスをフォームフィールドで使用した場合は、global.Profile 属性の値は Active Directory アカウントに反映されなくなります。

注 すべてのリソースへの値の反映を回避するために、グローバル属性の代わりに、アカウントに固有の属性をフォームで使用することが必要となる場合もあります。

リソース属性のオーバーライド

アカウント属性を設定する以外に、各アカウントのリソース属性のオーバーライドも指定できます。リソース属性は、Identity Manager のリソース定義用に定義され、したがって、リソースタイプ用に定義される属性です。これらは、個々のアカウントに関連付けられた属性ではありません。リソース属性の例には、サーバーのホスト名や、ディレクトリ内のベースコンテキストなどがあります。

リソースにアカウントを作成するが、1 つのリソース属性で別の値を使用する場合を考えてみましょう。これは、リソースを複製し、値を変更することで解決できます。しかし、リソースの過度な複製は混乱を生じる可能性があります。これに代わる方法として、ビューでアカウントごとにリソース属性をオーバーライドすることができます。

リソース属性のオーバーライドは、resourceAttributes という属性の下の属性オブジェクトに格納されます。たとえば、リソースが host という属性を定義した場合は、次のパスを記述することでビューに指定できます。

```
accounts[Active Directory].resourceAttributes.host
```

注　リソース属性のオーバーライドはあまりお勧めできませんが、それを避けられない場合もあります。同じ物理リソースを示すが、1 つの属性が異なる複製リソースの作成を回避するために、リソースの上書きを選択する場合もあります。たとえば、複数の Active Directory サーバーを使用する環境であれば、新しいリソースを作成するよりも、フォームでリソース属性 `host` をオーバーライドするほうが賢明かもしれません。詳細については、Identity Manager のご購入先までお問い合わせください。

accounts[Lighthouse]

Identity Manager リポジトリに格納されている属性のみの値を設定します。新規作成したビューは、`waveset.attributes` 属性セットに含まれる属性のコピーを持ちます。ビューを保存するときに、システムは `accounts[Lighthouse]` の内容と `waveset.attributes` の内容を比較し、更新レポートと監査ログエントリを生成します。この属性は Identity Manager のリポジトリに格納されますが、この属性の変更はリソースには自動的に反映されません。

Extended User Attributes 設定オブジェクトは、このビューで使用できる属性を定義します。システムは、この属性セットの中で、設定オブジェクトに登録されていない名前を無視します。

次のコードは、*Extended User Attributes* 設定オブジェクトの例を示しています。このオブジェクトは、`waveset.attribute` 属性セットによって管理される属性のリストを保持します。

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE Configuration PUBLIC 'waveset.dtd' 'waveset.dtd'>
<!--  id="#ID#Configuration:UserExtendedAttributes" name="User
Extended Attributes"-->
 <Configuration id='#ID#Configuration:UserExtendedAttributes'
name='User Extended Attributes' creator='Configurator'
createDate='1019603369733' lastMod='2' counter='0'>
  <Extension>
    <List>
      <String>firstname</String>
      <String>lastname</String>
      <String>fullname</String>
<!- 文字列の値をここに追加 - - >
      <String>SSN</String>
    </List>
  </Extension>
  <MemberObjectGroups>
    <ObjectRef type='ObjectGroup' id='#ID#Top' name='Top'/>
  </MemberObjectGroups>
</Configuration>
```

オブジェクトを修正し、デフォルトの firstname、lastname、および fullname 属性からリストを拡張できます。この例では、SSN という属性が追加されています。

accounts[Lighthouse].delegates

委任オブジェクトを workItemType に基づいて指定します。オブジェクトごとに作業項目タイプ固有の委任情報を指定します。

- delegateApproversTo の値が delegatedApproversRule である場合は、選択されている規則を指定します。
- delegateApproversTo の値が manager である場合は、この属性に値は指定しません。

この属性の属性については、「accounts[Lighthouse].delegate* 属性の属性」表を参照してください。

accounts[Lighthouse].delegatesHistory

委任オブジェクトを 0 ～ *n* で指定します。*n* は、委任履歴オブジェクトの現在の数 (最大数は委任履歴の深さ) です。この属性の属性については、「accounts[Lighthouse].delegate* 属性の属性」表を参照してください。

accounts[Lighthouse].delegatesOrginal

workItemType に基づく委任オブジェクトの元の一覧。get 操作または checkoutView 操作に従います。この属性の属性については、次の表を参照してください。

表 3-7 accounts[Lighthouse].delegate* 属性の属性

accounts[Lighthouse].delegate* 属性の属性	説明
workItemType	委任する workItem のタイプを識別します。有効な workItem タイプの一覧については、委任オブジェクトモデルの説明を参照してください。
workItemTypeObjects	名前を指定したロール、リソース、または組織について、ユーザーは今後の workItem 承認要求を委任します。この属性は、workItemType の値が roleApproval、resourceApproval、または organizationApproval のときに有効です。 指定しない場合は、このユーザーが承認者であるすべてのロール、リソース、または組織について今後の workItem 要求を委任することが、この属性の値になります (デフォルト)。
toType	委任するユーザーのタイプ。有効な値は次のとおりです。 • manager • delegateWorkItemsRule • selectedUsers

表 3-7　accounts[Lighthouse].delegate* 属性の属性 (続き)

accounts[Lighthouse].delegate* 属性の属性	説明
toUsers	委任するユーザーの名前を指定します (toType が selectedUsers の場合)。
toRule	規則を評価することによって委任するユーザーを決定するときに、規則の名前を指定します (toType が delegateWorkItemsRule の場合)。
startDate	委任が開始する日付を指定します。
endDate	委任が終了する日付を指定します。

accounts[Lighthouse].properties

この属性の値は、ユーザーが定義したプロパティーに対応する属性名を持つオブジェクトです。ユーザープロパティーを設定することで、任意のカスタムデータをユーザーとともに Identity Manager リポジトリに格納できます。格納されたプロパティーは、フォームやワークフローで使用できます。プロパティーは、拡張ユーザー属性と似ていますが、文字列や整数のような基本的なデータ型に限定されません。

Identity Manager は tasks システムプロパティーを定義します。このプロパティーは、将来の特定の日にワークフロータスクを実行するために、延期タスクスキャナで使用されます。tasks プロパティーの値は、オブジェクトのリストです。次の表は、リストに含まれるオブジェクトの属性を示しています。

表 3-8

属性	説明
name	実行する TaskDefinition オブジェクトの名前を指定します。
date	タスクを実行する日付を指定します。
taskName	作成される TaskInstance を指定します。指定しない場合は、Identity Manager によってランダムな名前が生成されます。
owner	タスクの所有者と見なされる Identity Manager 管理者を指定します。指定しない場合は、デフォルト値の Configurator が適用されます。
organization	TaskInstance の配置先となる Identity Manager 組織を指定します。指定しない場合は、タスク所有者によって制御される組織がランダムに選択されます。
description	作成時に TaskInstance に格納される説明文。このテキストは、Identity Manager の管理者インタフェースのタスク状態ページに表示されます。

用例

accounts[Lighthouse].properties の値を使用することで、ユーザーに割り当てられている延期タスクの表を表示できます。このリストは、sample/formlib.xml 内の Default User Library というフォームライブラリに追加されます。

延期タスクの表を表示するフィールドには、Deferred Tasks という名前が付けられています。waveset.properties 属性を変更すると、デフォルトの Tabbed User Form で延期タスクテーブルを参照できるようになります。延期タスクが存在する場合は、「ID」タブパネルの下部に表が表示されます。

accounts[Lighthouse].viewUserForm

表示専用ユーザーフォームの表示に使用されます。この表示専用フォームには、フィールド情報が Labels として表示され、管理者はそのユーザー情報の値を変更することはできませんが、一覧、表示、および検索を行うことは可能です。管理者がユーザーの詳細を表示するときは、アカウントリストからユーザーを選択し、「**表示**」をクリックします。

accounts[<resource>].properties

Identity Manager リポジトリへのアカウントプロパティーの格納に使用されます。この属性は、アカウントに関する何らかの情報 (たとえば、アカウントの作成日) をネイティブアカウント属性としてリソースに格納できない場合に使用されます。

accounts[<resource>].waveset.forceUpdate

この属性は、ユーザーが変更されたあとに、属性値が変更前の状態でリソースアクションに残っている場合に、更新のためにリソースに常に送信されるリソースアカウント属性のリストを指定するときに使用されます。この属性は、ユーザーがリソースから割り当てられていない場合のリソースアクションの実行で必要となります。

ユーザーフォームからの次のフィールド定義は、Solaris リソースを使用しています (<resource> は具体的なリソース名に置き換えられている)。

```
<Field name='accounts[waterloo].waveset.forceUpdate'>
   <Default>
      <List>
          <String>delete after action</String>
          <String>Home directory</String>
      </List>
   </Default>
</Field
```

前述のコードにより、Identity Manager はプロビジョニングツールとリソースアダプタに、delete after action 属性と Home directory 属性を送信します。

global 属性

ユーザービューの global 属性セットを使用することで、多数のリソースアカウント (Identity Manager を含む) に簡単に属性を割り当てることができます。global 属性の値は、global 属性として参照される属性を持つオブジェクトです。ビューを保存すると、システムは、スキーママップに global 属性名を定義するすべてのリソースアカウントに、各 global 属性の値を割り当てます。同じ名前の拡張属性が存在する場合、これらの値は Identity Manager のリポジトリにも反映されます。

たとえば、*R1* と *R2* という 2 つのリソースが、fullname という属性を定義するとします。ビューに global.fullname 属性が格納されると、その値は accounts[R1].fullname 属性と accounts[R2].fullname 属性に自動的にコピーされます。

global 属性を使用して、Identity Manager リポジトリに格納される拡張属性を割り当てることもできます。global 属性が Identity Manager の拡張属性としても宣言されている場合、これは accounts[Lighthouse] にコピーされます。

注 アカウントの作成時は、global.accountId を使用しないでください。アカウント ID は、リソースの DN テンプレートによって作成されます。global.accountId はこれに優先して適用されるため、問題が生じる可能性があります。

異なる 2 つの fullname 属性の参照

global 属性は、同じ名前を持つ account 属性と組み合わせて使用できます。たとえば、Active Directory リソースでは、fullname は lastname と firstname から構成されます。しかし、それ以外のすべてのリソースの fullname は、firstname lastname を使用します。

次の例は、フォームでこれら 2 つのフィールドを参照する方法を示しています。

```
<Field name='global.fullname'>
 <Expansion>
       <concat>
         <ref>global.firstname</ref><s> </s>
         <ref>global.lastname</ref>
       </concat>
 </Expansion>
</Field>
<Field name='accounts[ActiveDir].fullname'>
 <Expansion>
```

```
<Field name='global.fullname'>
       <concat>
          <ref>global.lastname</ref><s>, </s>
          <ref>global.firstname</ref>
       </concat>
   </Expansion>
</Field>
```

前述の例では、新規ユーザーの作成は、想定どおりに行われます。ただし、そのユーザーを読み込むときは、Active Directory リソースからの fullname 属性を使用して、global.fullname フィールドに値が取り込まれる可能性があります。

このような事例をより正確に実装するには、1 つのリソースを信頼できる属性のソースとして宣言し、次のような Derivation 規則を作成します。

```
<Field name='global.fullname'>
 <Derivation>
       <or>
         <ref>accounts[LDAP res].fullname</ref>
         <ref>accounts[AD res].fullname</ref>
       </or>
 </Derivation>
      <concat>
          <ref>global.firstname</ref><s> </s>
          <ref>global.lastname</ref>
       </concat>
   </Expansion>
</Field>
   <Expansion>
```

Derivation 規則を定義したことで、fullname フィールドへの値の取り込みには、最初に LDAP リソース内の fullname 属性が使用されます。LDAP に値が存在しない場合は、AD リソースからの値が取り込まれます。

accountInfo 属性

ユーザーに関連付けられているリソースアカウントの、読み取り専用の情報を持ちます。これは、ユーザービュー以外にも、システムビューで使用されます。このビューの一部の情報は、waveset.accounts 属性に格納されている情報の複製です。この複製には、2 つの理由があります。

- このビューの情報は、フォームで使いやすいように構成されている
- waveset ビュー全体を含めなくても、このビューは別のビューのコンポーネントとして使用できる

accountsInfo.accounts 属性には、ほとんどのアカウント情報が格納されます。その他の属性は、単にアカウント名のリストを含むに過ぎません。いずれかの名前リスト属性に格納されている名前に対して繰り返しを行い、この名前を使用してアカウントリスト属性にインデックスを指定する場合は、フォームの FieldLoop がよく使用されます。

たとえば、次のフォーム要素は、ロールを通じて間接的に割り当てられた各リソースの名前を持つラベルのリストを生成します。

```
    <Field name='accountInfo.accounts[$(name)].name>
       <FieldLoop for='name' in='accountInfo.fromRole'>
         <Display class='Label'/>
    </Field>
</FieldLoop>
```

次の表は、accountInfo ビュー属性を示しています。これらの属性は、ユーザーの特性を表現しています。

表 3-9 accountInfo の属性 (ユーザービュー)

属性	説明
accountInfo.accounts	ユーザーに関連付けられている各リソースアカウントに関する情報を持つオブジェクトのリスト (たとえば、created、disabled)。
accountInfo.assigned	ユーザーに割り当てられているリソースのリスト。
accountInfo.fromRole	ルールを通じてユーザーに割り当てられているリソースの、フラットリスト形式のリスト。
accountInfo.privates	ユーザーに直接割り当てられているリソースの、フラットリスト形式のリスト。
accountInfo.toCreate	ユーザーには割り当てられているが、そのアカウントが Identity Manager にはまだ存在しないすべてのリソースの名前のリスト。
accountInfo.toDelete	ユーザーには割り当てられなくなったが、存在は認識されているリソースの名前のリスト。
accountInfo.types	ユーザーに現在割り当てられている、または予約グループを通じて割り当てられている、各リソースタイプのリスト。
accountInfo.typeNames	割り当てられているすべてのリソースの、一意のタイプ名のリスト。

accountInfo.accounts

関連付けられている各リソースアカウントに関する情報をそれ自体に持つ、オブジェクトのリストを持ちます。アカウントリストの要素は、名前で参照されます。この名前は、リソース名です。

例

```
accountInfo.accounts[Active Directory].type
```

`accountInfo.accounts` リストを構成するオブジェクトは、次の表に示される属性を持ちます。

表 3-10 accountInfo.accounts の属性 (ユーザービュー)

属性	説明
`attributes`	このリソースによって定義されるすべてのアカウント属性に関する情報。
`name`	アカウントが存在するか、またはアカウントが作成されるリソースの名前。
`id`	リソースのリポジトリ ID。
`type`	リソースのタイプ名。
`accountId`	このリソースでのユーザーアカウントの名前。
`assigned`	アカウントが現在割り当てられている場合は、true となります。割り当てられていないアカウントは、Identity Manager によって削除される可能性があります。
`protected`	アカウントが現在保護されている場合は、true となります。この場合、そのアカウントに対する更新や削除の操作は無視されます。
`passwordPolicy`	このリソースに適用されるパスワードポリシーに関する情報。

accountInfo.accounts[ ].attributes[ ]

このリソースによって定義されるすべてのアカウント属性に関する情報を持ちます。これらの属性は、リソースのスキーママップページに一覧されます。属性の値は、オブジェクトのリスト (List) です。

次の表は、これらのオブジェクトが持つ属性を示しています。

表 3-11　accountInfo.accounts の属性 (ユーザービュー)

属性	説明
name	Identity Manager リソースアカウント属性の名前。この名前は、リソーススキーママップに定義されます。
syntax	属性値の構文。syntax 属性の値は、次のいずれかです。 int string boolean encrypted binary complex リソースで、バイナリ属性または複雑な属性がサポートされるかどうかを調べるときは、『Identity Manager リソースリファレンス』を参照してください。バイナリ属性や複雑な属性をサポートしないリソースに対して、このような属性を送信しようとすると、例外がスローされます。 バイナリ属性は、できるだけ小さく維持してください。350K バイトを超えるバイナリ属性を管理しようとすると、Identity Manager は例外をスローします。350K バイトを超える属性の管理が必要な場合は、カスタマサポートまでお問い合わせください。
multi	属性が複数の値をサポートする場合は、true となります。

フォームの設計時には、宣言されたリソースアカウントの属性タイプを気にする必要はありません。必要に応じて、ユーザービューのプロセッサシステムが、適切なタイプを強制します。

accountInfo.accounts[].passwordPolicy

リソースには、パスワードポリシーを割り当てることができます。属性にパスワードポリシーが割り当てられている場合、その属性の値には、そのポリシーに関する情報が含まれます。

次の表は、accountInfo.accounts[resname].passwordPolicy の属性を示しています。

表 3-12　accountInfo.accounts[resname].passwordPolicy の属性 (ユーザービュー)

属性	説明
name	ポリシーの名前。これは、Identity Manager リポジトリ内の policy オブジェクトの名前に対応します。
summary	各ポリシー属性に関する情報などの、ポリシーの簡単な説明。
attributes	この属性の値は、各ポリシー属性の名前と値を持つ別のオブジェクトです。

ポリシー情報を表示するアプリケーションは、通常は概要テキストを表示しますが、各ポリシー属性の表示をより詳細に制御する必要がある場合は、属性マップを使用できます。

パスワードの変更と同期のためのインタフェースを持つフォームは、多くの場合、この情報を使用します。

accountInfo.accounts[Lighthouse]

この accountInfo リストの特別エントリは、Identity Manager のデフォルトパスワードポリシーに関する情報の保持に使用されます。リソースアカウントに関する情報以外にも、Identity Manager のパスワードとポリシーに関する情報を表示する必要があるため、これはパスワードフォームを表示するときに便利です。

この要素は、パススルー認証が使用されていない場合にのみ表示されます。リソースタイプは Lighthouse です。

accountInfo のリソース名リスト

accountInfo ビューには、リソース名のリストを持つ属性が含まれます。各リストは、特定の特性を持つリソースに対して繰り返しを行う場合に、フォームで FieldLoop 構成体と組み合わせて使用することを目的としています。

リソース名を持つことができる accountInfo 属性は次のとおりです。

- assigned
- created
- fromRole
- private
- toCreate
- toDelete

accountInfo.assigned

ユーザーに割り当てられるリソースを指定します。フォームの設計時には、この属性を呼び出すことで、ロールやアプリケーションを通じて割り当てられているリソースのリストや、ユーザーに直接割り当てられているリソースのリストを表示できます。

accountInfo.typeNames

割り当てられているすべてのリソースの、一意のタイプ名のリスト。このリストは、特定タイプのリソースが選択されていないフィールドを無効にする場合に、フォームの Disable 式で使用されます。

```
<Field name='HomeDirectory' prompt='Home Directory'>
   <Display class='Text'/>
      <Disable>
         <not>
            <contains>
               <ref>accountInfo.typeNames</ref>
               <s>Solaris</s>
            </contains>
         </not>
      </Disable>
</Field>
```

これは、accountInfo.types[*].name というパスと同じ情報を返しますが、こちらのほうが効率的です。これは、Disable 式で使用する場合に重要な条件です。このリストには、共通リソースタイプを含めることができます。

Identity Manager の管理者インタフェースでこのリソースリストを表示することで、リソースタイプの名前を確認できます。このページの「**タイプ**」列には、現在定義されているリソースのタイプ名が表示されます。「**新規リソース**」の横のオプションリストにも、現在インストールされているリソースアダプタの名前が表示されます。

accountInfo.types

この属性は、現在割り当てられている各リソースタイプに関する情報を持ちます。この属性の値は、List (オブジェクト) です。

次の表は、各オブジェクトの属性を示しています。

表 3-13 accountInfo.types の属性 (ユーザービュー)

属性	説明
accounts	ユーザーに割り当てられている、このタイプの各アカウントの accountId のリスト

表 3-13　accountInfo.types の属性 (ユーザービュー) (続き)

属性	説明
name	リソースタイプ名

たとえば、次のパスによって、UNIX アカウントの ID のリストを指定できます。

```
accountInfo.types[Unix].accounts
```

display 属性

display 属性は、ビューの処理に適用されるコンテキストに関連する情報を持ちます。ほとんどの属性は、フォームの対話的な処理でのみ有効です。

次の表は、頻繁に使用される display ビュー属性を示しています。

表 3-14　頻繁に使用される display 属性 (ユーザービュー)

属性	説明
eventType	create または update という読み取り専用の値によって、ユーザービューが作成要求または更新要求を処理しているかどうかを示します。
session	認証された Identity Manager セッションへのハンドル。この属性は、Identity Manager の管理者インタフェースで対話的な編集セッションを実行している場合にのみ有効です。これは、Identity Manager リポジトリへのアクセスポイントを提供します。この属性の値は、com.waveset.ui.FormUtil クラス内のメソッドに渡すことができます。 display.session 属性は、フォームの処理が行われる可能性がある、次のような場合には無効です。 一括読み込みに含まれる場合 バックグラウンド再プロビジョニングの実行中 アクションまたは承認の同期がとれていない場合 推奨される方法は、この属性を Property 要素内、または Constraints 要素内のみで使用することです。ほとんどすべての既存のフォームでは、display.session 属性は Constraints 要素内のみで使用されます。

表 3-14　頻繁に使用される display 属性 (ユーザービュー) (続き)

属性	説明
subject	Identity Manager ユーザーまたは管理者のクレデンシャルに関する情報を保持するオブジェクト。この情報は、ほとんどすべての場合に設定されますが、通常は、display.session が無効になっている場合に、バックグラウンドアクティビティーで呼び出されるワークフローアプリケーションで使用されます。subject は、新規セッションの取得に使用できます。この場合は、リポジトリへのアクセスの取得に使用されます。
state	_javax.servlet.http.HttpSession_ などの HTTP 要求に関連するオブジェクトへのハンドルを持つ _com.waveset.ui.util.RequestState_ オブジェクトへのハンドル。

itemType のデフォルト動作

要求者が作業項目の所有者である場合に、ワークフローを作業項目に直接遷移させることができるのは、通常、wizard 項目タイプ (itemType) のみです。

itemType を次のように設定すると、ワークフローは作業項目に遷移しなくなり、その代わりに「承認」タブに表示されるようになります。

- approval
- custom
- itemType

デフォルト動作のオーバーライド

次のように、フォームのプロパティーとして allowedWorkItemTransitions オプションを設定することで、ユーザービューでのデフォルトの動作をオーバーライドすることができます。

```
<Form ......>
   <Properties>
     <Property name='allowedWorkItemTransitions'>
       <list>
         <s>myCustomType</s>
        </list>
<      /Property>
    </Properties>
```

deferred 属性

deferred 属性は、別のアカウントの属性値から値を取得する属性です。ビュー (および WSUser モデル) で deferred 属性を宣言すると、プロビジョニングエンジンは、アダプタを呼び出す前に、この置き換えをただちに実行します。

deferred 属性が、別のリソースの GUID 属性から値を取得する場合は、ソースアダプタはアクションを実行する必要がありません。しかし、ソース属性が GUID でない場合は、realCreate 操作の二次的な影響として、アダプタは ResourceInfo._resultsAttributes マップ内の属性を返さなければなりません。アダプタが属性を返さない場合は、プロビジョニングエンジンはアカウントをフェッチして、値を取得します。これは、値を返すようにアダプタを修正するよりも非効率です。

deferred 属性を使用する状況

deferred 属性は、新しいアカウントを作成し、アカウント属性の値を、ソースアカウントが作成されるまで認識されない別のアカウントの属性の値から取得するように指定するときに使用します。一例を挙げれば、生成される一意の識別子の値を属性に設定する場合がこれに該当します。

deferred 属性の使用

deferred 属性は、主に次の 2 つの手順で定義されます。

1. アカウントは、必ず 2 番目のアカウントの作成前に、ソースリソースに作成します。これは、リソースと、ユーザーへのリソースグループの割り当ての両方を含む、順序が付けられたリソースグループを作成することで行われます。
2. 次の例のように、作成するアカウントのユーザービューに特別な属性を設定します。それぞれの deferred 属性は、ソースアカウントを指定するビュー属性と、ソース属性を指定するビュー属性の 2 つを必要とします。これは、次の形式のパスを使用して設定されます。

   ```
   accounts[<resource>].deferredAttributes.<attname>.resource

   accounts[<resource>].deferredAttributes.<attname>.attribute
   ```

   この <resource> は実際のリソース名に置き換えられ、<attname> は実際の属性名に置き換えられます。

   たとえば、1) アカウントの作成時に uid 属性を生成する LDAP というリソースと、2) LDAP リソースの uid と同じ値を保持する directoryid という directoryid 属性を持つ HR というリソースの、2 つのリソースを作成する場合を考えてみましょう。

   次のフォームフィールドは、必要なビュー属性を設定し、この関係を定義します。

```
<Field name='accounts[HR].deferredAttributes.directoryid.resource'>
   <Expansion><s>LDAP</s></Expansion>
</Field>
<Field name='accounts[HR].deferredAttributes.directoryid
   <Expansion><s>uid</s></Expansion>
</Field>
```

ユーザービューのデバッグ

ユーザービューをデバッグするときは、ビューの内容を新しいファイルにダンプすると便利な場合があります。ダンプファイルを作成するには、次の Derivation ステートメントをユーザービューに追加します。

```
<Field name='DumpView'>
   <Derivation>
      <invoke name='dumpFile'>
         <ref>form_inputs</ref>
            <s>c:/temp/view.xml</s>
      </invoke>
   </Derivation>
</Field>
```

この Derivation 式は、`dumpFile` メソッドを呼び出すことで、ユーザーフォームが初めて表示されたあとにファイルを生成します。`form_inputs` 変数は、このフォームで使用されるビューに自動的にバインドされます。

前述の例では、`dumpFile` メソッドへの String 引数は、ファイルシステムのパスです。このパスは、実際には `c:/temp/view.xml` という有効パスに置き換えられています。

アカウント相関ビュー

指定されたアカウント (またはアカウント属性) と相互に関連するユーザーの検索に使用されます。このビューは、アカウント調整プロセスの一部として使用されます。

このビューには、次の root 属性が含まれます。これらの属性の値は GenericObjects です。新しい ID は <account_name>@<resource_name> となります。

表 3-15 アカウント相関ビューの最上位属性

属性	説明
correlation	相互関係の設定方法に関する情報を持ちます
matches	相互関係の結果を持ちます

相関要求は、ビューの取得処理と表示更新要求の両方で実行されます。表示更新の場合は、ビューに指定されている要求が使用されます (ただし、ビュー ID の値が優先的に適用される accountId と resource を除く)。取得要求の場合は、ビュー属性と同じ名前のビューオプション (たとえば、correlator) を使用して、ビューから提供される要求部分を指定できます。

注 ビューオプションとして指定される場合は、accountAttributes は WSUser (リソースアダプタメソッドから返される) または GenericObject として指定できます。

correlation

表 3-16 correlation 属性の属性 (アカウント相関ビュー)

属性	編集の可能性	データ型	必要性
accountId	読み取り	String	あり
accountGUID	読み取り / 書き込み	String	なし (ただし、accountId と resource がリソースを明確に識別できない場合を除く)
リソース	読み取り	String	あり
accountAttributes	読み取り / 書き込み	String	

表 3-16　correlation 属性の属性 (アカウント相関ビュー) (続き)

属性	編集の可能性	データ型	必要性
correlator	読み取り / 書き込み	String	なし
confirmer	読み取り / 書き込み	String	なし

accountId

相互に関連付けるアカウントの名前を指定します。これは、ビュー ID から自動的に取得されます。

accountGUID

相互に関連付けるアカウントの GUID を指定します。accountId と resource がリソースを明確に識別できない場合にのみ、必須属性となります。

resource

アカウントが存在するリソースの名前を指定します。この値は、ビュー ID から自動的に取得されます。

accountAttributes

アカウントの属性を指定します。指定されている場合、ビューアは、相関規則と確認規則に渡すときに、現在のアカウント属性をフェッチしません。その代わりに、これらの属性が渡されます。

correlator

使用する相関規則を指定します。指定しない場合は、リソースの調整ポリシーによって指定される相関規則が使用されます。NULL 値を指定した場合は、相関規則は使用されません。

confirmer

使用する確認規則を指定します。指定しない場合は、リソースの調整ポリシーによって指定される確認規則が使用されます。NULL 値を指定した場合は、確認規則は使用されません。

これらのリストは、ユーザーの概要属性を持つ GenericObject から構成されます。

表 3-17　confirmer 属性の属性 (アカウント相関ビュー)

属性	編集の可能性	データ型
claimants	読み取り	List
correlated	読み取り	List
unconfirmed	読み取り	List

claimant

要求者 (claimant) をもう一方のリストにも表示させるために、相関アルゴリズムに関係なく計算された要求者のリスト。要求者の検出は、ビューオプションで ignoreClaimants を true に設定することで無効にできます。アカウントを明示的に参照する ResourceInfo を持つ場合、ユーザーはアカウントを要求します。

correlated

リソースアカウントと相互に関連付けられているユーザーのリスト。

unconfirmed

相関規則によって選択されたが、確認規則によって拒否されたユーザーのリスト。このリストは、ビューオプションで includeUnconfirmed が true に設定されている場合にのみ存在します。

管理者ロールビュー

ユーザーに管理者ロールを作成または更新するときに使用されます。管理者ロールを使用すると、組織の組み合わせごとに一意の機能セットを定義できます。機能と制御する組織は、直接割り当てるだけでなく、ロールを通じて間接的に割り当てることもできます。

1 つまたは複数の管理者ロールを 1 人のユーザーに割り当てたり、1 人または複数のユーザーを同じ管理者ロールに割り当てたりすることができます。

表 3-18 管理者ロールビューの最上位属性

名前	編集の可能性	データ型	必要性
id	読み取り / 書き込み	String	なし
name	読み取り / 書き込み	String	あり
capabilities		List	あり
capabilitiesRule		String	あり
controlledOrganizations		List	あり
controlledOrganizationsRule		String	あり
controlledOrganizationsUserform		String	あり
controlledSubOrganizations		List (オブジェクト)	なし
memberObjectGroup		List	あり

id

Identity Manager 内の AdminRole オブジェクトを一意に識別します。システム生成の値。

name

管理者ロールの名前を指定します。

capabilities

この管理者ロールに割り当てられる機能名のリストを指定します。

capabilitiesRule

ゼロ個以上の割り当て機能名のリストを返す、評価対象となる規則の名前を指定します。

controlledOrganizations

関連付けられている機能が許可される組織名のリスト。

controlledOrganizationsRule

評価対象となる規則の名前を指定します。この規則は、割り当てられるゼロ個以上の制御対象組織名のリストを返します。

controlledOrganizationsUserform

この管理者ロールによる制御の対象となる組織の範囲内で、ユーザーの編集または作成に使用されるユーザーフォームを指定します。この管理者規則が適用されるユーザーに、このユーザーフォームが直接割り当てられていない場合に有効です。

controlledSubOrganizations

使用可能オブジェクトのサブセットが含められるか、または除外される、制御対象組織のリスト。この属性の値は、`controlledSubOrganization` オブジェクトのリストから構成されます。各 ControlledOrganization オブジェクトのビュー属性は次のとおりです。

表 3-19 controlledSubOrganizations のビュー属性 (管理者ロールビュー)

属性	データ型	必要性
`name`	String (制御対象オブジェクトグループの名前)	
`types`	List (オブジェクト)	

`types` は、タイプ (たとえば、Resource、Role、および Policy) ごとに含められるか、または除外されるオブジェクトのリストです。各オブジェクトタイプのビューは次のとおりです。

表 3-20 controlledSubOrganizations ビュー属性のオブジェクトタイプ (管理者ロールビュー)

属性	データ型	必要性
`name`	String	
`include`	List (オブジェクト)	
exclude	List (オブジェクト)	

name
オブジェクトタイプの名前を指定します。

include
含められる関連オブジェクトタイプのオブジェクト名のリスト。

exclude
除外される関連タイプのオブジェクト名のリスト。

memberObjectGroup

この管理者ロールがメンバーとして属す ObjectGroup のリスト。これは、この管理者ロールを使用できるオブジェクトグループ (組織) です。

ユーザーの秘密質問の回答変更ビュー

1 つまたは複数のログインインタフェースで使用される、既存ユーザーの秘密質問の回答を変更するときに使用されます。

2 つの上位属性があります。

表 3-21 ユーザーの秘密質問の回答変更ビューの属性

属性	編集の可能性	データ型	必要性
questions		List	
loginInterface		String	

questions

質問を表現します。含まれる属性は次のとおりです。

表 3-22 questions の属性 (ユーザーの秘密質問の回答変更ビュー)

属性	データ型	必要性
qid	String	
question	String	
answer	String	
answerObfuscated	Boolean	

qid

質問を一意に識別します。これは、その質問を、ポリシーに定義されている質問に関連付けるために使用されます。

question

ポリシーに定義される質問の文字列を指定します。

answer

指定する場合、qid の値と関連付けられる、ユーザーの質問の回答を指定します。

answerObfuscated

回答をそのまま表示するか、暗号化するかを指定します。

loginInterface

この質問が関連付けられるログインインタフェースを指定します。値は、各ログインインタフェースの、一意のメッセージカタログキーです。

含まれる属性は次のとおりです。

表 3-23　loginInterface の属性 (ユーザーの秘密質問の回答変更ビュー)

属性	データ型	必要性
name	String	
questionPolicy	String	
questionCount	String	

name

質問が関連付けられるログインインタフェースの名前を指定します。

次に有効な値を示します。

- UI_LOGIN_CONFIG_DISPLAY_NAME_ALL_INTERFACES
- UI_LOGIN_CONFIG_DISPLAY_NAME_ADMIN_INTERFACE
- UI_LOGIN_CONFIG_DISPLAY_NAME_CLI_INTERFACE
- UI_LOGIN_CONFIG_DISPLAY_NAME_DEFAULT_USER_INTERFACE
- UI_LOGIN_CONFIG_DISPLAY_NAME_IVR_INTERFACE
- UI_LOGIN_CONFIG_DISPLAY_NAME_QUESTION_INTERFACE
- UI_LOGIN_CONFIG_DISPLAY_NAME_USER_INTERFACE

questionPolicy

この質問が関連付けられるポリシーを指定します (たとえば、All、Random、Any、または RoundRobin)。

questionCount

`questionPolicy` 属性を Any または Random に設定した場合にのみ、設定します。

ユーザー機能変更ビュー

Identity Manager ユーザーの機能を変更するときに使用されます。

表 3-24　ユーザー機能変更ビューの属性

属性	編集の可能性	データ型	必要性
adminRoles		List [String]	
capabilities		List [String]	
controlledOrganizations		List [String]	

adminRoles

ユーザーに割り当てられている管理者ロールのリスト。

capabilities

このユーザーに割り当てられている機能のリスト。

controlledOrganizations

ユーザーが、割り当てられている能力によって制御する組織のリスト。

作業項目の委任ビュー

このビューは、特定のユーザーの作業項目を委任するときに使用します。

次の最上位属性があります。

manager

workItem を委任するユーザーの accountId を指定します。idmManager が割り当てられていないユーザーの場合は、この値は NULL になります。

name

作業項目を委任するユーザーを (名前で) 識別します。

delegates

委任オブジェクトを workItemType インデックスに基づいて指定します。オブジェクトごとに作業項目 (workItem) タイプ固有の委任情報を指定します。

delegatesHistory

委任オブジェクトを 0 ～ *n* で指定します。*n* は、委任履歴オブジェクトの現在の数 (最大数は委任履歴の深さ) です (委任履歴の深さとは、再利用のためにこれまでに記録された委任の数のことです。システム設定オブジェクトに記録される数は、security.delegation.historyLength 属性に 0 より大きな整数値を設定することによって設定できます。デフォルトの記録数は 10 です)。

これらの各属性には、次の属性があります。

表 3-25

accounts[Lighthouse].delegate* 属性の属性	説明
workItemType	委任される workItem のタイプを識別します。有効な workItem タイプの一覧については、委任オブジェクトモデルの説明を参照してください。
workItemTypeDisplayName	ユーザーにわかりやすい workItem タイプ名を指定します。この名前は Identity Manager の製品インタフェースに表示されます。
workItemTypeObjects	名前を指定したロール、リソース、または組織について、ユーザーは今後の workItem 承認要求を委任します。この属性は、workItemType の値が roleApproval、resourceApproval、または organizationApproval のときに有効です。 指定しない場合は、このユーザーが承認者であるすべてのロール、リソース、または組織について今後の workItem 要求を委任することが、この属性の値になります (デフォルト)。

表 3-25　　(続き)

accounts[Lighthouse].delegate* 属性の属性	説明
toType	委任するユーザーのタイプ。有効な値は次のとおりです。 • manager • delegateWorkItemsRule • selectedUsers
toUsers	委任するユーザーの名前を指定します (toType が selectedUsers の場合)。
toRule	規則を評価することによって委任するユーザーを決定するときに、規則の名前を指定します (toType が delegateWorkItemsRule の場合)。
startDate	委任が開始する日付を指定します。
endDate	委任が終了する日付を指定します。
status	委任の開始日と終了日、および現在の委任リストに含まれているかどうかに基づいて、委任の概要が表示されます。

フォームからの DelegateWorkItems ビューオブジェクトの参照

次のコード例は、DelegateWorkItems ビュー委任オブジェクトをフォームから参照する方法を示しています。

```
<Field name='delegates[*].workItemType'>
<Field name='delegates[*].workItemTypeDisplayName'>
<Field name='delegates[*].workItemTypeObjects'>
<Field name='delegates[*].toType'>
<Field name='delegates[*].toUsers'>
<Field name='delegates[*].toRule'>
<Field name='delegates[*].startDate'>
<Field name='delegates[*].endDate'>
<Field name='delegates[*].status'>
```

サポートされるインデックス値 (*) は workItemType 値です。

次のコード例は、委任履歴オブジェクトを DelegateWorkItems ビューから参照する方法を示しています。

```
<Field name='delegatesHistory[*].workItemType'>
```

```
<Field name='delegatesHistory[*].workItemTypeDisplayName'>
<Field name='delegatesHistory[*].workItemTypeObjects'>
<Field name='delegatesHistory[*].toType'>
<Field name='delegatesHistory[*].toUsers'>
<Field name='delegatesHistory[*].toRule'>
<Field name='delegatesHistory[*].startDate'>
<Field name='delegatesHistory[*].endDate'>
<Field name='delegatesHistory[*].selected'>
<Field name='delegatesHistory[*].status'>
```

サポートされるインデックス値 (*) は 0 ～ *n* です。*n* は委任履歴オブジェクトの現在の数 (最大は委任履歴の深さ) です。

表 3-26 作業項目のタイプ

workItem タイプ	説明	表示名
Approval	WorkItem を継承する	承認
OrganizationApproval	Approval を継承する	組織の承認
ResourceApproval	Approval を継承する	リソースの承認
RoleApproval	Approval を継承する	ロールの承認
Attestation	WorkItem	アクセスレビューアテステーション
review	WorkItem	是正
accessReviewRemediation	WorkItem	アクセス

プロビジョン解除ビュー

プロビジョニング解除するリソースリストの表示と選択に使用されます。1 つの最上位属性があります。

resourceAccounts

この属性は、次の属性を持ちます。

表 3-27　resourceAccounts の属性 (プロビジョン解除ビュー)

名前	編集の可能性	データ型	必要性
id	読み取り / 書き込み	String	
selectAll	読み取り / 書き込み	Boolean	
unassignAll	読み取り / 書き込み	Boolean	
unlinkAll	読み取り / 書き込み	Boolean	
currentResourceAccounts	読み取り	List (オブジェクト)	
fetchAccounts	読み取り / 書き込み	Boolean	
fetchAccountResources	読み取り / 書き込み	List	

id

アカウントの一意の識別子を指定します。

selectAll

すべてのリソースを選択するかどうかを制御します。

unassignAll

プライベートリソースのユーザーのリストから、すべてのリソースを削除するかどうかを指定します。

unlinkAll

Identity Manager ユーザーから、すべてのリソースとのリンクを解除するかどうかを指定します。

tobeCreatedResourceAccounts

この Identity Manager ユーザーに割り当てられているが、まだ作成されていないアカウントを表します。まだ作成されていないアカウントでは、パスワードのロックを解除することはできません。

tobeDeletedResourceAccounts

すでに作成されているが、このユーザーには割り当てられなくなったアカウントを表します。削除されるアカウントでは、パスワードを変更することはできません。

3 つのアカウントリストはすべて、各リソースのアカウントの状態を表現するオブジェクトを含み、ユーザーはアカウントを個別に選択できます。

currentResourceAccounts

Identity Manager によって現在管理されているアカウントのセットを表します (Identity Manager アカウント自体を含む)。

すべてのアカウントリストには、リソース名によるインデックスが付けられます。

表 3-28 currentResourceAccounts の属性 (プロビジョン解除ビュー)

名前	編集の可能性	データ型
selected	読み取り / 書き込み	Boolean
unassign	読み取り / 書き込み	Boolean
unlink	読み取り / 書き込み	Boolean
name	読み取り	String
type	読み取り	String
accountId	読み取り	String
exists	読み取り	Boolean
disabled	読み取り	Boolean
authenticator	読み取り	Boolean
directlyAssigned	読み取り	Boolean

selected

`true` に設定されている場合は、指定されたリソースで、関連付けられているアカウントがプロビジョニング解除されることを示します。選択されているアカウントが Lighthouse である場合は、すでに選択されている場合を除き、Identity Manager ユーザーとすべての関連リソースの割り当てが削除されます。ただし、関連付けられているリソースアカウントは削除されません。

unassign

`true` に設定されている場合は、ユーザーのプライベートリソースのリスト (たとえば、`waveset.resources`) から、指定されているリソースが削除されることを示します。

unlink

`true` に設定されている場合は、Identity Manager ユーザーから、指定されているリソースとのリンクが解除されることを示します (たとえば、関連 ResourceInfo オブジェクトの削除)。

注	selected または unassign が true に設定されている場合は、unlink も true に設定されていることになります。しかし、その反対は必ずしもそのように限定されません。unlink を true に設定し、selected と unassign を false に設定することは可能です。

name

リソースの名前を指定します。これは、Identity Manager リポジトリ内の `resource` オブジェクトの名前に対応します。

type

リソースのタイプ (たとえば、Solaris) を指定します。Identity Manager の管理者インタフェースでこのリソースリストを表示することで、リソースタイプの名前を確認できます。このページの「**タイプ**」列には、現在定義されているリソースのタイプ名が表示されます。「**新規リソース**」の横のオプションリストにも、現在インストールされているリソースアダプタの名前が表示されます。

accountId

リソースアカウントの ID を指定します。

exists

リソースにすでにアカウントが存在するかどうかを示します (`currentResourceAccounts` のみ)。

disabled

アカウントが現在有効であるか、無効であるかを示します (`currentResourceAccount` のみ)。

authenticator

アカウントが、ユーザーのログインが設定されているアカウントの 1 つであるかどうかを示します。

directlyAssigned

`true` に設定されている場合は、そのユーザーにアカウントが直接割り当てられていることを示します。値が `false` の場合は、ロールまたはアプリケーションを通じて、アカウントが間接的に割り当てられていることを示します。

fetchAccounts

ビューに、ユーザーに割り当てられているリソースのアカウント属性を含めます。

詳細については、この章の「フォームでのビューオプションの設定」を参照してください。

fetchAccountResources

フェッチ先リソースの名前のリスト。指定しない場合は、割り当てられているすべてのリソースが使用されます。

詳細については、この章の「フォームでのビューオプションの設定」を参照してください。

無効化ビュー

Identity Manager ユーザーのアカウントを無効にするときに使用されます。このビューは、カスタムワークフローでよく使用されます。

resourceAccounts

このビューで属性にアクセスするときの、最上位属性を表します。

表 3-29　resourceAccounts 属性の属性 (無効化ビュー)

名前	編集の可能性	データ型	必要性
id	読み取り	String	
selectAll	読み取り	Boolean	
currentResourcesAccount	読み取り	String	
fetchAccounts	読み取り / 書き込み	Boolean	
fetchAccountResources	読み取り / 書き込み	List	

id

ユーザーの Identity Manager ID を指定します。

selectAll

設定すると、Identity Manager アカウントを含むすべてのリソースアカウントが無効化されます。

currentResourceAccounts

Identity Manager によって現在管理されているアカウントのセットを表します (Identity Manager アカウント自体を含む)。特定のリソースを有効に指定するときは、selected フィールドを使用します。

表 3-30　resourceAccounts.currentResourceAccounts の属性 (無効化ビュー)

名前	編集の可能性	データ型
name	読み取り	String
type	読み取り	String

表 3-30　resourceAccounts.currentResourceAccounts の属性 (無効化ビュー) (続き)

名前	編集の可能性	データ型
accountId	読み取り	String
exists	読み取り	Boolean
disabled	読み取り	Boolean
selected	読み取り / 書き込み	Boolean

fetchAccounts

ビューに、ユーザーに割り当てられているリソースのアカウント属性を含めます。

詳細については、この章の「フォームでのビューオプションの設定」を参照してください。

fetchAccountResources

フェッチ先リソースの名前のリスト。指定しない場合は、割り当てられているすべてのリソースが使用されます。

詳細については、この章の「フォームでのビューオプションの設定」を参照してください。

有効化ビュー

Identity Manager ユーザーのアカウントを有効にするときに使用されます。このビューは、カスタムワークフローでよく使用されます。

resourceAccounts

このビューで属性にアクセスするときの、最上位属性を表します。

表 3-31　resourceAccounts 属性の属性 (有効化ビュー)

名前	編集の可能性	データ型	必要性
id	読み取り	String	
selectAll	読み取り	Boolean	
currentResourcesAccount	読み取り	String	
fetchAccounts	読み取り / 書き込み	Boolean	
fetchAccountResources	読み取り / 書き込み	List	

id

ユーザーの Identity Manager ID を指定します。

selectAll

設定すると、Identity Manager アカウントを含むすべてのリソースアカウントが有効化されます。

currentResourceAccounts

Identity Manager によって現在管理されているアカウントのセットを表します (Identity Manager アカウント自体を含む)。特定のリソースを有効に指定するときは、selected フィールドを使用します。

表 3-32　resourceAccount.currentResourceAccounts の属性 (有効化ビュー)

名前	編集の可能性	データ型
name	読み取り	String
type	読み取り	String

表 3-32　resourceAccount.currentResourceAccounts の属性 (有効化ビュー) (続き)

名前	編集の可能性	データ型
accountId	読み取り	String
exists	読み取り	Boolean
disabled	読み取り	Boolean
selected	読み取り / 書き込み	Boolean

fetchAccounts

ビューに、ユーザーに割り当てられているリソースのアカウント属性を含めます。

詳細については、この章の「フォームでのビューオプションの設定」を参照してください。

fetchAccountResources

フェッチ先リソースの名前のリスト。指定しない場合は、割り当てられているすべてのリソースが使用されます。

詳細については、この章の「フォームでのビューオプションの設定」を参照してください。

オブジェクト検索ビュー

カスタマイズが可能で、一般的な Identity Manager リポジトリ検索インタフェースを提供します。Identity Manager に定義されているオブジェクトタイプと適切な権限を持ち、非推奨とされていたり、内部使用に限定されたりしていないオブジェクトを検索できます。オブジェクト検索ビューハンドラには、1 つまたは複数の属性クエリー条件とパラメータを指定したり、検索結果を表示したりするための関連フォームが用意されています。また、ビューオプションを使用して、属性クエリー条件とパラメータを指定できます。

このビューは、次の属性を持ちます。

表 3-33 最上位属性 (オブジェクト検索ビュー)

名前	編集の可能性	データ型	必要性
objectType	読み取り / 書き込み	String	あり
allowedAttrs	読み取り / 書き込み	List	なし
attrsToGet	読み取り / 書き込み	List	なし
attrConditions	読み取り / 書き込み	List	なし
maxResults	読み取り / 書き込み	String	なし
results	読み取り	List	なし
sortColumn	読み取り / 書き込み	String	なし
selectEnable	読み取り / 書き込み	Boolean	なし

objectType

検索する Identity Manager リポジトリオブジェクトのタイプ (たとえば、Role、User、または Resource) を指定します。

allowedAttrs

デフォルトでは objectType の `listQueryableAttributeAttrs()` メソッドを呼び出すことで取得される、クエリー可能な属性名のオブジェクトタイプ (`objectType` 属性によって指定) を指定するリスト。このメソッドは、PersistentObject を拡張する各クラスによって開示されます。オブジェクトタイプクラスによってオーバーライドされないときは、すべての PersistentObject でサポートされる、クエリー可能な属性のデフォルトセットを返す PersistentObject 実装を継承します。

デフォルトの設定は、findObjectsDefaults.xml 設定ファイルのデフォルトセクションまたは objectType に固有のセクションのいずれかに、allowedAttrs のセットを指定することでオーバーライドできます。このファイルは、サンプルディレクトリに格納されています。許可されるそれぞれの属性を、次のように sample/findObjectsDefaults.xml ファイルに指定します。

name

属性を指定します。

displayName

Identity Manager の管理者インタフェースに表示する属性名を指定します。指定しない場合は、この属性の値はデフォルト値である name と同じ値になります。

syntax

属性値のデータ型を指定します。サポートされる値は、string、int、boolean です。指定しない場合は、この値はデフォルト値である string になります。

multiValued

属性が複数の値をサポートするかどうかを指定します。true という値は、属性が複数の値をサポートすることを表します。指定しない場合は、この値はデフォルト値である false になります。この属性は、属性構文が string である場合にのみ適用されます。

allowedValuesType

属性の許可される値が Identity Manager タイプのインスタンス (たとえば、Role、Resource) である場合に、Identity Manager のタイプ名を指定します。指定しない場合は、この属性にはデフォルト値である NULL が適用されます。

name 属性が、Identity Manager によって定義される属性である場合は、name の指定のみが必須となります。name 属性が拡張属性である場合は、name の指定は必須となりますが、デフォルト値が適切である限り、その他の属性の指定は省略可能です。

許可される属性の設定例については、sample/findObjectsDefaults.xml を参照してください。

allowedAttrs のリストは、文字列のリスト、オブジェクトのリスト、または両者の組み合わせによって指定できます。

attrsToGet

指定された属性クエリー条件と一致する各オブジェクトとともに返される、指定されたオブジェクトタイプ (objectType) の概要属性名のリスト。オブジェクトタイプの、サポートされる概要属性のセットを取得するときは、そのオブジェクトタイプの listSummaryAttributeAttrs() メソッドを呼び出します。このメソッドは、PersistentObject を拡張する各クラスによって開示されます。objectType クラスによってオーバーライドされないときは、すべての PersistentObject でサポートされる、概要属性のデフォルトセットを返す PersistentObject 実装を継承します。

デフォルトは、sample/findObjectsDefaults.xml 設定ファイルのデフォルトセクションまたは objectType に固有のセクションのいずれかに、resultColumnNames のリストを指定することでオーバーライドできます。

attrConditions

指定された属性条件 (attrConditions) と一致する、指定されたオブジェクトタイプ (objectType) のオブジェクトの検索に使用される属性条件のリスト。リストを構成する各属性条件は、次のように指定します。

selectedAttr

許可される属性のリスト (allowedAttrs) から、いずれかの属性名を指定します。

selectedAttrRequired

(省略可能) この属性条件に合わせて、選択された属性 (selectedAttr) を変更できるかどうかを指定します。true という値は、選択された属性を、この属性条件に合わせて変更できないことを表し、この場合、属性条件のリストからその属性条件を削除することはできません。

defaultAttr

(省略可能) 許可される属性のリストがインタフェースに表示されるときに、デフォルトで選択される allowedAttrs の名前を指定します。

allowedOperators

選択された属性 (selectedAttr) に指定されている構文に基づいて許可される演算子のリスト。デフォルトでは、getAllowedOperators メソッドを呼び出し、選択された属性 (selectedAttr) の syntax 属性と multiValued 属性の値を渡すことで、このリストを取得できます。デフォルトは、sample/findObjectsDefaults.xml 設定ファイルのデフォルトセクションまたは objectType に固有のセクションのいずれかに、許可される演算子 (allowedOperators) のセットを指定することでオーバーライドできます。

selectedOperator

allowedOperators に指定されているリストから、いずれかの演算子の名前を指定します。

selectedOperatorRequired

(省略可能) この属性条件に合わせて、選択された演算子 (selectedOperator) を変更できるかどうかを指定します。true という値は、選択された演算子を、この属性条件に合わせて変更できないことを表し、この場合、属性条件のリストからその属性条件を削除することはできません。

defaultOperator

(省略可能) 許可される演算子 (allowedOperators) のリストがフォームに表示されるときに、デフォルトで選択される演算子 (allowedAttrs) の名前を指定します。

value

指定されたオブジェクトタイプ (objectType) のオブジェクトを返す必要があると Identity Manager が判断した場合にテストの対象となる、選択された属性名と演算子の値またはオペランドを指定します。selectedOperator の値が exists または notPresent である場合は、この属性を省略できます。

valueRequired

(省略可能) 属性条件の value を変更できるかどうかを指定します。値が true の場合は、値を変更できます。これは、属性条件のリストからその属性条件を削除できないことも意味します。

removeAttrCond

この属性条件の削除の必要性を指定します (内部)。

FindObjects.ATTR_CONDITIONS 定数または attrCondition 文字列を使用することで、属性条件をビューオプションとして指定できます。attrConditions を指定しない場合は、Identity Manager は指定されたオブジェクトタイプのすべてのオブジェクトを返します。

maxResults

(省略可能) 検索要求によって Identity Manager から返される、指定された objectType のオブジェクトの最大数を指定します。指定しない場合のデフォルト値は 100 です。デフォルトは、sample/findObjectsDefaults.xml 設定ファイルのデフォルトセクションまたは objectType に固有のセクションのいずれかに、resultMaxRows 属性の値を指定することでオーバーライドできます。

この属性を使用することで、指定したタイプの Identity Manager リポジトリオブジェクトが多数存在する場合のパフォーマンスを改善できます。

results

attrsToGet の値が NULL の場合、result の値は、指定された属性条件と一致するオブジェクト名のリストとなります。attrsToGet の値が NULL 以外の場合は、results は、指定された attrConditions と一致するオブジェクトのリストとなります。各オブジェクトは、次の内容から構成されます。

- *columns* - 要求された attrsToGet と一致する、表示可能な列名のリスト
- *rows* - 0 から行数 (たとえば、10) までの番号が名前として付けられた *row* オブジェクトのリスト
- *row* - 0 から列数 (たとえば、6) までの番号と、それに該当する *rows* 列の値が名前として付けられたオブジェクトのリスト

sortColumn

(省略可能) 結果をソートする列の値を指定します。指定しない場合のデフォルト値は 0 です。デフォルトは、sample/findObjectsDefaults.xml 設定ファイルのデフォルトセクションまたは objectType に固有のセクションのいずれかに、resultSortColumn の値を指定することでオーバーライドできます。

selectEnable

(省略可能) 複数の結果行を同時に選択できるかどうかを指定します。値が `true` の場合は、複数の結果行を選択できます。デフォルトは `false` です。デフォルトは、sample/findObjectsDefaults.xml 設定ファイルのデフォルトセクションまたは `objectType` に固有のセクションのいずれかに、`resultSelectEnable` の値を指定することでオーバーライドできます。

組織ビュー

作成する組織のタイプと、それを処理するオプションのタイプの指定に使用されます。

代表的な属性

次の表は、このビューの上位属性を示しています。

表 3-34 組織ビューの属性

名前	編集の可能性	データ型	必要性
orgName	読み取り	String	システム生成値
orgDisplayName	読み取り / 書き込み	String	あり
orgType	読み取り / 書き込み	String	なし
orgId	読み取り	String	システム生成値
orgAction	書き込み	String	なし
orgNewDisplayName	書き込み	String	なし
orgParentName	読み取り / 書き込み	String	なし
orgChildOrgNames	読み取り	List	システム生成値
orgApprovers	読み取り / 書き込み	List	なし
allowsOrgApprovers	読み取り	List	システム生成値
allowedOrgApproverIds	読み取り	List	システム生成値
orgUserForm	読み取り / 書き込み	String	なし
orgViewUserForm	読み取り / 書き込み	String	なし
orgPolicies	読み取り / 書き込み	List	なし
orgAuditPolicies	読み取り / 書き込み	List	なし
renameCreate	読み取り / 書き込み	String	なし
renameSaveAs	読み取り / 書き込み	String	なし

orgName

組織の UID を指定します。短い組織名は同じであっても、親組織が異なる場合があるため、この値はほとんどのビューオブジェクト名と異なります。

orgDisplayName

短い組織名を指定します。これは、表示のためにのみ使用される値であり、一意である必要はありません。

orgType

組織のタイプを指定します。指定できる値は、`junction` または `virtual` です。`junction` または `virtual` というタイプ以外の組織は、値を持ちません。

orgId

Identity Manager 内で組織を一意に識別するための ID を指定します。

orgAction

ディレクトリジャンクション、仮想組織、および動的組織のみでサポートされます。指定できる値は、`refresh` です。組織がディレクトリジャンクションまたは仮想組織である場合、表示更新の動作は、`orgRefreshAllOrgsUserMembers` の値によって異なります。

orgNewDisplayName

組織の名前を変更する場合の、新しい短い名前を指定します。

orgParentName

親組織の完全パス名を指定します。

orgChildOrgNames

直接的および間接的なすべての子組織の Identity Manager インタフェース名のリスト。

orgApprovers

この組織に追加または変更されたユーザーの承認を担当する Identity Manager 管理者のリスト。

allowedOrgApprovers

この組織に追加または変更されたユーザーの承認者になりうる、ユーザーの名前のリスト。

allowedOrgApproverIds

この組織に追加または変更されたユーザーの承認者になりうる、ユーザーの ID のリスト。

orgUserForm

この組織に属すユーザーが、ユーザーの作成時または編集時に使用する `userForm` を指定します。

orgViewUserForm

この組織に属すユーザーが、ユーザーを表示するときに使用するビューユーザーフォームを指定します。

orgPolicies

この組織に属すすべてのユーザーに適用されるポリシーを指定します。これは、String 型のキーが付けられたオブジェクトのリストです。各ポリシーオブジェクトには、`orgPolicies[<type>]` というプレフィックスを持つ、次のビュー属性が含まれます。このプレフィックスの `<type>` は、ポリシーのタイプ (たとえば、Lighthouse アカウント) を表します。

- `policyName` -- 名前を指定します
- `id` -- ID を表します
- `implementation` -- このポリシーを実装するクラスを識別します

orgAuditPolicies

この組織に属すすべてのユーザーに適用される監査ポリシーを指定します。

renameCreate

true に設定すると、この組織を複製し、`orgNewDisplayName` の値を使用して新しい組織を作成します。

renameSaveAs

true に設定すると、`orgNewDisplayName` の値を使用して、この組織の名前を変更します。

ディレクトリジャンクションと仮想組織の属性

表 3-35　ディレクトリジャンクションと仮想組織の属性

名前	編集の可能性	データ型	必要性
orgContainerId	読み取り	String	システム生成値
orgContainerTypes	読み取り	List	システム生成値
orgContainers	読み取り	List	システム生成値
orgParentContainerId	読み取り	String	システム生成値
orgResource	読み取り / 書き込み	String	あり (ディレクトリジャンクションまたは仮想組織の場合)
orgResourceType	読み取り	String	システム生成値
orgResourceId	読み取り	String	システム生成値
orgRefreshAllOrgsUserMembers	書き込み	String	なし

orgContainerId

関連付けられている LDAP ディレクトリコンテナの dn (たとえば、cn=foo,ou=bar,o=foobar.com) を指定します。

orgContainerTypes

別のリソースオブジェクトを持つことができる、許可されるリソースオブジェクトタイプのリスト。

orgContainers

選択リストを表示するときに Identity Manager インタフェースが使用する、リソースのベースコンテナのリスト。

orgParentContainerId

関連付けられている親 LDAP ディレクトリコンテナの dn (たとえば、ou=bar,o=foobar.com) を指定します。

orgResource

ディレクトリジャンクションと仮想組織の同期に使用される Identity Manager リソースの名前 (たとえば、West Directory Server) を指定します。

orgResourceType

ディレクトリジャンクションと仮想組織の同期に使用される Identity Manager リソースのタイプ (たとえば、LDAP) を指定します。

orgResourceId

ディレクトリジャンクションと仮想組織の同期に使用される Identity Manager リソースの ID を指定します。

orgRefreshAllOrgsUserMembers

`orgAction` の値が `refresh` の場合に `true` に設定すると、Identity Manager 組織に属すユーザーと、選択された組織とそのすべての子組織のリソースコンテナに属すユーザーが同期されます。`false` に設定した場合は、リソースコンテナに属すユーザーは同期されず、リソースコンテナと、選択された組織とそのすべての子組織の Identity Manager 組織のみが同期されます。

動的組織の属性

表 3-36　動的組織の属性

名前	編集の可能性	データ型	必要性
orgUserMembersRule	読み取り / 書き込み	String	なし
orgUserMembersRuleCacheTimeout	読み取り / 書き込み	String	なし

orgUserMembersRule

ユーザーの所属を確認するために実行時に評価される、`authType` が UserMembersRule の規則を、名前または UID で指定します。

orgUserMembersCacheTimeout

orgUserMembersRule によって返されるユーザーがキャッシュされる場合に、タイムアウトとなるまでの時間をミリ秒単位で指定します。値 0 は、キャッシュなしを表します。

パスワードビュー

Identity Manager ユーザー、またはそのリソースアカウントのパスワードを変更するときに、管理者によって使用されます。

このビューには、1 つの最上位属性があります。

resourceAccounts

この属性は、次の属性を持ちます。

表 3-37　resourceAccounts の属性 (パスワードビュー)

属性	編集の可能性	データ型	必要性
id	読み取り / 書き込み	String	あり
selectAll	読み取り / 書き込み	Boolean	なし
currentResourceAccounts	読み取り	List (オブジェクト)	なし
tobeCreatedResourceAccounts	読み取り	List (オブジェクト)	なし
tobeDeletedResourceAccounts	読み取り	List (オブジェクト)	なし
password	読み取り / 書き込み	encrypted	あり
confirmPassword	読み取り / 書き込み	encrypted	あり (ビューを対話的に使用する場合)
fetchAccounts	読み取り / 書き込み	Boolean	
fetchAccountResources	読み取り / 書き込み	List	

id

パスワード変更の対象となる Identity Manager ユーザーのアカウント ID を指定します。通常はビューハンドラによって設定され、フォームから変更することはありません。

selectAll

すべてのパスワードを選択するかどうかを制御します。

currentResourceAccounts

Identity Manager によって現在管理されているアカウントのセットを表します (Identity Manager アカウント自体を含む)。

tobeCreatedResourceAccounts

この Identity Manager ユーザーに割り当てられているが、まだ作成されていないアカウントを表します。まだ作成されていないアカウントでは、パスワードを変更することはできません。

tobeDeletedResourceAccounts

このユーザーに割り当てられてはいるが、まだ Identity Manager による管理の対象となっていないリソースのセット (たとえば、`resinfo` オブジェクトがまだ関連付けられていないリソース) を表します。削除されるアカウントでは、パスワードを変更することはできません。

3 つのアカウントリストはすべて、各リソースのアカウントの状態を表現するオブジェクトを含み、ユーザーはアカウントを個別に選択できます。

どちらのリソースアカウントリストにもリソース名によるインデックスが付けられ、このユーザーがアカウントを持つリソースを表現するオブジェクトを含みます。

表 3-38 tobeDeletedResourceAccounts の属性 (パスワードビュー)

属性	編集の可能性	データ型
selected	読み取り / 書き込み	Boolean
`name`	読み取り	String
`type`	読み取り	String
`accountId`	読み取り	String
`exists`	読み取り	ブール型 (`currentResourceAccounts` のみ)
`disabled`	読み取り	ブール型 (`currentResourceAccounts` のみ)
`passwordPolicy`	読み取り	Object
`authenticator`	読み取り	Boolean

表 3-38　tobeDeletedResourceAccounts の属性 (パスワードビュー) (続き)

属性	編集の可能性	データ型
changePasswordLocation	読み取り	String (currentResourceAccounts のみ)
expirePassword	読み取り / 書き込み	Boolean

password

Identity Manager アカウントまたはリソースアカウントに割り当てる新しいパスワードを指定します。

confirmPassword

password 属性に指定したパスワードを確認します。ビューを対話的に使用する場合は、フォームの「password」フィールドと「confirmPassword」フィールドに同じ値を入力してください。ワークフローでの使用のように、ビューをプログラム的に使用する場合は、confirmPassword 属性は無視されます。このビューを対話的に使用している場合は、この属性を設定してください。

selected

指定されたリソースが、新しいパスワードを必要とすることを指定します。

name

リソースの名前を指定します。これは、Identity Manager リポジトリ内の resource オブジェクトの名前に対応します。

type

リソースのタイプ (たとえば、Solaris) を指定します。Identity Manager の管理者インタフェースでこのリソースリストを表示することで、リソースタイプの名前を確認できます。このページの「**タイプ**」列には、現在定義されているリソースのタイプ名が表示されます。「**新規リソース**」の横のオプションリストにも、現在インストールされているリソースアダプタの名前が表示されます。

accountId

このリソースにアカウントが作成されている場合は、そのアカウントの ID を指定します。

exists

リソースにすでにアカウントが存在するかどうかを示します。

disabled

アカウントが現在無効化されているかどうかを示します。

passwordPolicy

このリソースにパスワードポリシーが設定されている場合は、その説明を指定します。NULL も指定できます。これには、次の属性があります。

表 3-39　passwordPolicy の属性 (パスワードビュー)

属性	説明
name	String
summary	String

また、この属性には、宣言されている各ポリシー属性のビュー属性も含まれます。ビュー属性の名前 (name) は、ポリシーに定義されている名前と同じです。

概要文字列 (summary) には、事前に書式が設定された、ポリシー属性の説明が含まれます。

authenticator

true に設定した場合は、このリソースが Identity Manager のパススルー認証リソースとして機能することを示します。

changePasswordLocation

(省略可能) パスワードの変更が行われる場所 (たとえば、Active Directory のドメインコントローラの DNS 名) を指定します。このフィールドの値の形式は、リソースによって異なります。

expirePassword

変更の直後に、パスワードに有効期限切れのマークを付けるかどうかを制御するときは、NULL 以外のブール型の値を設定します。NULL に設定した場合は、パスワードが変更されたユーザーが、パスワードを変更したユーザーと異なる場合に、デフォルトとして、パスワードを有効期限切れにします。

tobeCreatedResourceAccounts

この Identity Manager ユーザーに割り当てられているが、まだ作成されていないアカウントを表します。まだ作成されていないアカウントでは、パスワードを変更することはできません。

tobeDeletedResourceAccounts

すでに作成されているが、このユーザーには割り当てられなくなったアカウントを表します。削除されるアカウントでは、パスワードを変更することはできません。

fetchAccounts

ビューに、ユーザーに割り当てられているリソースのアカウント属性を含めます。

詳細については、この章の「フォームでのビューオプションの設定」を参照してください。

fetchAccountResources

フェッチ先リソースの名前のリスト。指定しない場合は、割り当てられているすべてのリソースが使用されます。

詳細については、この章の「フォームでのビューオプションの設定」を参照してください。

プロセスビュー

ワークフローやレポートなどのタスクの呼び出しに使用されます。呼び出されるタスクは、Identity Manager の TaskDefinition オブジェクトまたは TaskTemplate オブジェクトを使用して定義しておく必要があります。タスクを呼び出すと、TaskInstance オブジェクトが作成されます。

このビューには、`task` という 1 つの最上位属性があります。それ以外の最上位属性は、入力としてタスクに渡されますが、値の指定は任意です。

task

この最上位属性は、タスクを呼び出す方法を定義します。

表 3-40 プロセスビューの属性

属性	編集の可能性	データ型	必要性
`process`	読み取り / 書き込み	String	あり
`taskName`	読み取り / 書き込み	String	あり
`organization`	読み取り / 書き込み	String	あり
`taskDisplay`	読み取り / 書き込み	String	なし
`description`	読み取り / 書き込み	String	なし
`execMode`	読み取り / 書き込み	String	なし
`result`	読み取り / 書き込み	WavesetResult	なし
`owner`	読み取り / 書き込み	String	なし

process

呼び出すプロセスの名前。この属性には、Identity Manager の TaskDefinition オブジェクトまたは TaskTemplate オブジェクトの名前を指定できます。また、System Configuration オブジェクトの `process` の設定を利用してマップされる抽象プロセス名も指定できます。これは、必須属性です。

taskName

タスクの実行時状態を保持するために作成される、TaskInstance オブジェクトに付ける名前を指定します。この属性に値を指定しない場合は、生成されるランダムな名前が適用されます。

organization

TaskInstance の配置先となる組織の名前を指定します。この属性に値を指定しない場合、TaskInstance は最上位に配置されます。

taskDisplay

TaskInstance の表示名を指定します。

description

TaskInstance を説明する文字列を指定します。この文字列は、製品インタフェースの「サーバータスク」テーブルに表示されます。

execMode

実行モードを指定します。通常は、値を指定しません。この場合、実行モードは TaskDefinition によって決定されます。この属性に指定した設定は、TaskDefinition の値に優先して適用されます。

`execMode` の許可される値は次のとおりです。

表 3-41　　execMode 属性の値 (プロセスビュー)

値	説明
`sync`	同期実行またはフォアグラウンド実行を指定します
`async`	非同期実行またはバックグラウンド実行を指定します
`asyncImmediate`	ただちにスレッドを開始する非同期実行を指定します

`asyncImmediate` 実行モードは、直列化できない値をビューを通じてタスクに渡さなければならない、特別なシステムタスクのみで使用してください。タスクスレッドはただちに開始されます。デフォルトの動作では、TaskInstance は一時的にリポジトリに保存され、後からスケジューラによって再開されます。

result

TaskInstance の初期結果を指定します。この設定を利用して、タスクの完了時にタスクの結果を最終的に表示するためのタスクに、情報を渡すことができます。

owner

タスクの所有者と見なされるユーザーの名前を指定します。値を設定しない場合は、ログインユーザーが所有者と見なされます。

ビューオプション

次のオプションは、createView メソッドと checkinView メソッドが認識するオプションです。

endUser

タスクが Identity Manager のユーザーインタフェースから呼び出されることを指定します。これにより、正式な権限を持たないユーザーも、特別に指定されたエンドユーザータスクを呼び出すことができます。

process

呼び出すプロセスの名前。この名前は、createView メソッドによって認識され、ビューの process 属性の値となります。

suppressExecuteMessage

`true` に設定すると、非同期タスクを呼び出したときに、タスクの結果に追加されるデフォルトのメッセージが抑制されます。デフォルトの英文テキストは、「`The task is being executed in the background` (タスクはバックグラウンドで実行中)」です。

checkinView メソッドの結果

checkinView メソッドから返される WavesetResult オブジェクトには、次の名前が付けられた結果項目が記録されます。

表 3-42 checkinView メソッドの結果

結果	説明
`taskId`	TaskInstance のリポジトリ ID を識別します
`taskState`	TaskInstance の現在の状態を識別します。値は、ready (準備完了)、executing (実行中)、suspended (保留)、または finished (完了) のいずれかです。
`extendedResults`	`true` が指定されている場合は、TaskInstance が拡張された結果を持つことを表します。

調整ビュー

リソースに対する調整処理の要求またはキャンセルに使用されます。このビューは、ワークフローの一部として、オンデマンド調整を行うときに使用されます。また、調整のためのカスタムスケジューラの実装にも使用されます。

このビューは書き込み専用です。取得やチェックアウトの操作はサポートされません。

request

実行する操作を指定します。次の有効な操作のいずれかを指定してください。

表 3-43　request 属性で有効な操作 (調整ビュー)

操作	説明
FULL	リソースの完全調整を開始します
INCREMENTAL	リソースの差分調整を開始します
ACCOUNT	アカウントの調整を開始します
CANCEL	現在アクティブなリソース調整プロセスをキャンセルします

accountId

調整するアカウントを指定します。要求が ACCOUNT 以外の場合、この文字列は無視されます。

例

- リソースの単一アカウントの調整を要求する方法は次のとおりです。ここでは、Active Directory リソースを例に取ります。

```
request = “ACCOUNT”

accountId = “cn=maurelius, ou=Austin, DC=Waveset, DC=com”
```

- 保留中、または現在アクティブな、リソースの調整プロセスをキャンセルします。

```
request = “CANCEL”
```

調整ポリシービュー

Identity Manager のシステム設定オブジェクトの一部として格納されている調整ポリシーを表示、変更するときに使用されます。調整ポリシーと調整ポリシービュー

調整ポリシーの設定は、ツリー構造に格納されます。

このツリーの一般的な構造は次のとおりです。

- デフォルトまたはグローバルポリシー (`Default`)。これは root ポリシーレベルです。
- リソースタイプ (`ResType:`) ポリシー
- リソースポリシー (`Resource:`)

設定は、ツリーの任意のポイントに指定できます。レベルがポリシーの値を指定しない場合は、その次に上位のポリシーから継承されます。

ビューは、ポリシーツリー上の指定のポイントで、有効なポリシーを表します。これは、ビュー名で識別されます。

表 3-44 調整ポリシーツリーとビュー名

ビュー名	説明
`Default`	ポリシーツリーの root を示します
`ResType`: リソースタイプ	root の下位の指定リソースタイプを示します
`Resource`: リソース名	リソースのリソースタイプの下位の指定リソースを示します

ポリシー値

ポリシー設定の値は、常にポリシー値となります。ポリシー値には、最大で、次の表に示される 3 つのコンポーネントを含めることができます。

表 3-45 ポリシー値の設定属性 (調整ポリシービュー)

ポリシー値の設定	説明
`value`	設定の値を指定します。

表 3-45　ポリシー値の設定属性 (調整ポリシービュー) (続き)

ポリシー値の設定	説明
scope	この設定の取得元となる範囲を指定します。範囲の値には、Local、ResType、および Default があります。これは、ポリシーを指定しているラベルを示します。たとえば、SCOPE_LOCAL という値は、現在のポリシーレベルに設定される値を表します。 SCOPE_LOCAL -- リソースレベルまたは現在のポリシーレベルにポリシーが設定されます SCOPE_RESTYPE -- restype (リソースタイプ) レベルにポリシーが設定されます SCOPE_GLOBAL. -- グローバルレベルにポリシーが設定されます
inheritance	このレベルで継承されたポリシー設定を識別します。scope が Local でない場合、inheritance は有効な値と一致します。Default レベルのポリシー設定には存在しません。

認証の必要性

ビューを変更するユーザーには、Reconcile Administrator 機能が必要です。

ビューにアクセスするユーザーには、Reconcile Administrator 機能または Reconcile Request Administrator 機能が必要です。

ビュー属性

次の表は、このビューの上位属性を示しています。

表 3-46　調整ポリシービューの属性

属性	説明
scheduling	調整の自動化されたスケジュールに関する情報を含みます。
correlation	リソースアカウントの所有を決定する方法に関する情報を含みます。
ワークフロー	ユーザーが調整プロセスに指定する拡張に関する情報を含みます。
response	検出された状況に対して調整がどのように応答するかに関する情報を含みます。
リソース	調整がリソースとどのように連携するかに関する情報を含みます。

scheduling

表 3-47　scheduling の属性 (調整ポリシービュー)

属性	編集の可能性	データ型
reconcileServer	読み取り / 書き込み	String
reconcileModes	読み取り / 書き込み	String
fullSchedule	読み取り / 書き込み	Schedule
incrementalSchedule	読み取り / 書き込み	Schedule
nextFull	読み取り	Date
nextIncremental	読み取り	Date

reconcileServer

スケジューリングされた調整の実行に使用される調整サーバーを指定します。

reconcileModes

有効にする調整モードを指定します。有効な値は、BOTH、FULL、NONE です。

fullSchedule

完全調整が有効な場合のスケジュールを指定します。

incrementalSchedule

差分調整が有効な場合のスケジュールを指定します。

nextFull

差分調整が有効な場合の、次の調整の日時を指定します。

nextIncremental

スケジュールの繰り返し回数を指定します。スケジュールの値は、次の属性を持つ GenericObject です。

- count -- スケジュールの繰り返し回数を指定します
- units -- スケジュールの繰り返し単位を指定します
- time -- スケジュールの開始時刻を指定します

correlation

相関規則の名前を指定します。

表 3-48　相関規則 (調整ポリシービュー)

属性	編集の可能性	データ型
correlationRule	読み取り / 書き込み	String
confirmationRule	読み取り / 書き込み	String

correlationRule

アカウントとユーザーを関連付けるときに使用される、相関規則の名前を指定します。

confirmationRule

関連付けられているユーザーをアカウントに対して確認するときに使用される、確認規則の名前を指定します。確認が必要ない場合は、CONFIRMATION_RULE_NONE という値を指定します。

workflow

表 3-49　workflow の属性 (調整ポリシービュー)

属性	編集の可能性	データ型
proxyAdministrator	読み取り / 書き込み	String
preReconWorkflow	読み取り / 書き込み	String
perAccountWorkflow	読み取り / 書き込み	String
postReconWorkflow	読み取り / 書き込み	String

proxyAdministrator

管理機能を持つユーザーの名前を指定します。

preReconWorkflow、*perAccountWorkflow*、*postReconWorkflow*

調整プロセスの適切なタイミングで実行するワークフローの名前を指定します。実行するワークフローを指定しないときは、AR_WORKFLOW_NONE という値を指定します。

response

表 3-50　response の属性 (調整ポリシービュー)

属性	編集の可能性	データ型
situations	読み取り / 書き込み	List
explanations	読み取り / 書き込み	Boolean

situations

指定された状況で実行する、自動化された応答を指定します。有効な応答は次のとおりです。

表 3-51　situations 属性のオプション (調整ポリシービュー)

応答	説明
DO_NOTHING	自動化された応答を実行しません
CREATE_NEW_USER	リソースアカウントに基づいて、新規ユーザーを作成します
LINK_ACCOUNT	要求ユーザーにアカウントを割り当てます
CREATE_ACCOUNT	リソースにアカウントを再作成します
DELETE_ACCOUNT	リソースからアカウントを削除します
DISABLE_ACCOUNT	リソースのアカウントを無効にします

explainActions

調整が、アクションの詳細な説明をアカウントインデックスに記録するかどうかを指定します。

resource

表 3-52　resource の属性 (調整ポリシービュー)

属性	編集の可能性	データ型
reconcileNativeChanges	読み取り / 書き込み	Boolean
reconciledAttributes	読み取り / 書き込み	List (String のリスト)
listTimeout	読み取り / 書き込み	Integer
fetchTimeout	読み取り / 書き込み	Integer

reconcileNativeChanges

アカウント属性へのネイティブ変更を調整の対象とするかどうかを指定します。

reconciledAttributes

ネイティブ変更の中で監視の対象となるアカウント属性のリストを指定します。

listTimeout

リソースに存在するアカウントを列挙するときに、調整が応答を待機する最大時間をミリ秒単位で指定します。

fetchTimeout

リソースからアカウントをフェッチするときに、調整プロセスが応答を待機する最大時間をミリ秒単位で指定します。

調整状態ビュー

最後に要求された調整処理の状態を取得するときに使用されます。このビューは読み取り専用です。

status

状態コード要求 (文字列) を示します。有効な状態コードは次のとおりです。

表 3-53　調整状態ビューの属性

状態コード	説明
UNKNOWN	状態を特定できません。その他の属性の値は指定されません。
PENDING	要求は受信されましたが、まだ処理されていません。
RUNNING	要求は、現在処理中です。
COMPLETE	要求は完了しています。その他の要求の成功または失敗を特定するには、属性を確認します。
CANCELLED	要求は管理者によってキャンセルされました。

reconcileMode

要求の調整モードを示します。有効な値は、FULL または INCREMENTAL です。

reconciler

調整要求を処理する Identity Manager サーバーを指定します。

requestedAt

要求を受信した日付を示します。

startedAt

調整処理が開始された日付を示します。調整処理がまだ開始されていないか、または保留中にキャンセルされた場合、この値は NULL となります。

finishedAt

調整処理が完了した日付を示します。調整処理がまだ完了していない場合は、この値は NULL となります。

errors.fatal

調整処理の中止原因となったエラー (存在する場合) を説明します。エラーは、文字列のリストとして返されます。

errors.warnings

調整処理の実行中に発生した、致命的ではないエラーを説明します。エラーは、文字列のリストとして返されます。

statistics.accounts.discovered

調整処理の実行時にリソースで検出されたアカウントの数を示します。

statistics.situation[<situation>].resulting

応答処理の実行後に (成功、失敗を問わず)、指定された調整状態にあるアカウントの数を示します。

有効な状況は次のとおりです。

- CONFIRMED
- FOUND
- DELETED
- MISSING
- COLLISION
- UNMATCHED
- UNASSIGNED
- DISPUTED

ユーザー名変更ビュー

Identity Manager ユーザーとリソースアカウントアイデンティティーの名前の変更に使用されます。このビューは、通常は企業内のユーザーの名前を変更するときに使用されます。また、それ以外にも、ディレクトリ構造での移動の要因となるディレクトリユーザーのアイデンティティーを変更するときに、このビューが使用されます。

表 3-54 ユーザー名変更ビューの属性

名前	編集の可能性	データ型	必要性
newAccountId	読み取り / 書き込み	String	
toRename	読み取り	List	
noRename	読み取り	List	
resourceAccounts	読み取り		
fetchAccounts	読み取り / 書き込み	Boolean	
fetchAccountResources	読み取り / 書き込み	List	

newAccountId

Identity Manager ユーザーに設定され、リソースアカウントのアイデンティティーテンプレートで使用される、新しい accountId を指定します。

toRename

名前変更処理をサポートする currentResourceAccounts リストに含まれるアカウントのリストを指定します。

noRename

名前変更機能をサポートしないアカウントのリストを指定します。

resourceAccounts

リソースアカウントに関する、主に読み取り専用の情報を持ちます。リソースアカウント名の変更に使用される属性は次のとおりです。

表 3-55 resourceAccounts の属性

属性	データ型	説明
selectAll	Boolean	すべてのアカウントの名前を変更するかどうかを制御します。

表 3-55　resourceAccounts の属性 (続き)

属性	データ型	説明
currentResourceAccounts [<resourcename>].selected	Boolean	このリソースアカウントのアイデンティティー名の変更に、新しい accountId を使用することを指定します。
currentResourceAccounts [Lighthouse].selected	Boolean	アカウントの名前を変更するかどうかを制御します。selectAll=true は、この設定に優先して適用されます。

accounts[<resourcename>].identity

このリソースアカウントの accountId の作成時に、アイデンティティーテンプレートの使用をオーバーライドします。

accounts[<resourcename>].<attribute>

新規 accountId の作成時に、アイデンティティーテンプレートに属性を渡す accounts[<resourcename>].identity 属性を指定しない場合に使用されます。

fetchAccounts

ビューに、ユーザーに割り当てられているリソースのアカウント属性を含めます。

詳細については、この章の「フォームでのビューオプションの設定」を参照してください。

fetchAccountResources

フェッチ先リソースの名前のリスト。指定しない場合は、割り当てられているすべてのリソースが使用されます。

詳細については、この章の「フォームでのビューオプションの設定」を参照してください。

例

```
renameView.newAccountId="saurelius"
renameView.resourceAccounts.selectAll="false"
renameView.resourceAccounts.currentResourceAccounts[Lighthouse].selected="true"
renameView.accounts[AD].identity="cn=saurelius,OU=Austin,DC=Waveset,DC=com"
renameView.resourceAccounts.currentResourceAccounts[AD].selected="true"
```

```
renameView.newAccountId="saurelius"
renameView.accounts[LDAP].identity="CN=saurelius,CN=Users,DC=us,DC=com"
renameView.resourceAccounts.currentResourceAccounts[LDAP].selected="true"
renameView.accounts[AD].identity="Marcus Aurelius"
renameView.resourceAccounts.currentResourceAccounts[AD].selected="true"
```

再プロビジョンビュー

再プロビジョニングするリソースのリストの表示と選択に使用されます。このビューには、1 つの最上位属性 (resourceAccounts) があります。

resourceAccounts

この属性は、次の属性を持ちます。

表 3-56　resourceAccounts の属性 (再プロビジョンビュー)

名前	編集の可能性	データ型	必要性
id	読み取り	String	
selectAll	読み取り / 書き込み	Boolean	
currentResourceAccounts	読み取り	List (オブジェクト)	
fetchAccounts	読み取り / 書き込み	Boolean	
fetchAccountResources	読み取り / 書き込み	List	

id

アカウントの一意の識別子を指定します。

selectAll

すべてのリソースを選択するかどうかを制御します。

currentResourceAccounts

Identity Manager によって現在管理されているアカウントのセットを表します (Identity Manager アカウント自体を含む)。

すべてのアカウントリストには、リソース名によるインデックスが付けられます。

表 3-57　currentResourceAccounts の属性 (再プロビジョンビュー)

名前	編集の可能性	データ型
selected	読み取り / 書き込み	Boolean
name	読み取り	String

表 3-57　currentResourceAccounts の属性 (再プロビジョンビュー) (続き)

名前	編集の可能性	データ型
type	読み取り	String
accountId	読み取り	String
exists	読み取り	Boolean
disabled	読み取り	Boolean
authenticator	読み取り	Boolean

selected

`true` に設定されている場合は、指定されたリソースで、関連付けられているアカウントが再プロビジョニングされることを示します。選択されているアカウントが Lighthouse である場合は、すでに選択されている場合を除き、Identity Manager ユーザーとすべての関連リソースの割り当てが再プロビジョニングされます。ただし、関連付けられているリソースアカウントは再プロビジョニングされません。

name

リソースの名前を指定します。これは、Identity Manager リポジトリ内の `resource` オブジェクトの名前に対応します。

type

リソースのタイプ (たとえば、Solaris) を指定します。Identity Manager の管理者インタフェースでこのリソースリストを表示することで、リソースタイプの名前を確認できます。このページの「**タイプ**」列には、現在定義されているリソースのタイプ名が表示されます。「**新規リソース**」の横のオプションリストにも、現在インストールされているリソースアダプタの名前が表示されます。

accountId

リソースアカウントの ID を指定します。

exists

リソースにすでにアカウントが存在するかどうかを示します (`currentResourceAccounts` のみ)。

disabled

アカウントが現在有効であるか、無効であるかを示します (`currentResourceAccount` のみ)。

authenticator

アカウントが、ユーザーのログインが設定されているアカウントの 1 つであるかどうかを示します。

fetchAccounts

ビューに、ユーザーに割り当てられているリソースのアカウント属性を含めます。

詳細については、この章の「フォームでのビューオプションの設定」を参照してください。

fetchAccountResources

フェッチ先リソースの名前のリスト。指定しない場合は、割り当てられているすべてのリソースが使用されます。

詳細については、この章の「フォームでのビューオプションの設定」を参照してください。

ユーザーパスワードのリセットビュー

管理者がパスワードをランダム生成のパスワードにリセットするときに使用されます。また、オプションとして、新しいパスワードをリソースアカウントに反映させます。

resourceAccounts

リソースアカウントの特性を定義します。この属性は、次の属性を持ちます。

表 3-58　resourceAccounts の属性 (ユーザーパスワードのリセットビュー)

属性	編集の可能性	データ型	必要性
id	読み取り	String	
selectAll	読み取り / 書き込み	Boolean	
currentResourceAccounts	読み取り	List (オブジェクト)	
tobeCreatedResourceAccounts	読み取り	List (オブジェクト)	
tobeDeletedResourceAccounts	読み取り	List (オブジェクト)	

id

パスワード変更の対象となる Identity Manager ユーザーのアカウント ID を指定します。

selectAll

すべてのパスワードを選択するかどうかを制御します。

currentResourceAccounts

Identity Manager によって現在管理されているアカウントのセットを表します (Identity Manager アカウント自体を含む)。

tobeCreatedResourceAccounts

この Identity Manager ユーザーに割り当てられているが、まだ作成されていないアカウントを表します。まだ作成されていないアカウントでは、パスワードを変更することはできません。

tobeDeletedResourceAccounts

すでに作成されているが、このユーザーには割り当てられなくなったアカウントを表します。削除がスケジューリングされるアカウントでは、パスワードを変更することはできません。

アカウントリストの属性には、tobeDeletedResourceAccounts、tobeCreatedResourceAccounts、currentResourceAccounts の 3 つがあります。これらの属性は、次の表に示される属性を持ちます。これらの属性は、各リソースのアカウントの状態を表現し、ユーザーはアカウントを個別に選択できます。

表 3-59 tobeDeletedResourceAccounts の属性 (ユーザーパスワードのリセットビュー)

属性	編集の可能性	データ型	必要性
selected	読み取り / 書き込み	Boolean	
name	読み取り	String	
type	読み取り	String	
accountId	読み取り	String (currentResourceAccounts のみ)	
exists	読み取り	ブール型 (currentResourceAccounts のみ)	
disabled	読み取り	ブール型 (currentResourceAccounts のみ)	
passwordPolicy	読み取り	Object	
authenticator	読み取り	Boolean	
changePasswordLocation	読み取り	String	

selected

このアカウントのパスワードをリセットする場合は、true に設定します。

name

リソースの名前を指定します。これは、Identity Manager リポジトリ内の Resource オブジェクトの名前に対応します。

type

リソースのタイプ (たとえば、Solaris) を指定します。Identity Manager の管理者インタフェースでこのリソースリストを表示することで、リソースタイプの名前を確認できます。このページの「**タイプ**」列には、現在定義されているリソースのタイプ名が表示されます。「**新規リソース**」の横のオプションリストにも、現在インストールされているリソースアダプタの名前が表示されます。

accountId

このリソースにアカウントが作成されている場合は、そのアカウントの ID を指定します。

exists

リソースにすでにアカウントが存在するかどうかを示します。

disabled

アカウントが現在無効化されているかどうかを示します。

passwordPolicy

このリソースにパスワードポリシーが設定されている場合は、その説明を指定します。NULL も指定できます。これには、次の属性があります。

表 3-60 passwordPolicy の属性 (ユーザーパスワードのリセットビュー)

属性	データ型	編集の可能性	必要性
`name`	String		
`summary`	String		

また、この属性には、宣言されている各ポリシー属性のビュー属性も含まれます。ビュー属性の名前 (name) は、ポリシーの WSAttribute の名前と同じです。

概要文字列 (summary) には、事前に書式が設定された、ポリシー属性の説明が含まれます。

authenticator

`true` に設定した場合は、このリソースが Identity Manager のパススルー認証リソースとして機能することを示します。

changePasswordLocation

パスワードの変更が行われる場所 (たとえば、Active Directory のドメインコントローラの DNS 名) を指定します。このフィールドの値の形式は、リソースによって異なります。

リソースビュー

リソースの変更に使用されます。

具体的にはこのビューを作成するビューハンドラによって、さまざまなビューメソッドのリソースパラメータが次のようにインスタンス化されます。

- createView メソッドは、リソースタイプに適した prototypeXML の特定に使用される、typeString オプションを必要とします。prototypeXML には、リソースパラメータの初期セットと、その初期値が含まれます。そのため、ビューには、この初期リソースパラメータのリストとそのデフォルト値が取り込まれます。
- getView メソッドと checkoutView メソッドは、リソースオブジェクトに存在するリソースパラメータのみを返します。実際のリソースオブジェクトにいずれかのリソースパラメータが不足している場合は、このリストへのデータの取り込みに prototypeXML は使用されません。
- リポジトリに格納されているリソースオブジェクトのリソースパラメータのリストは、checkinView メソッドによって置き換えられます。この場合も、checkinView 処理の実行中に提供されなかった不足リソースパラメータへのデータの取り込みに、prototypeXML は使用されません。

  checkinView メソッドは、リソース管理ワークフローを呼び出し、リポジトリへの変更は実際にはこのワークフローによって適用されます。承認または通知を取り込むためにこのワークフローを修正することができます。

最上位属性

このビューには次の最上位属性があります。

表 3-61　リソースビューの属性

属性	編集の可能性	データ型	必要性
accountAttributes	読み取り / 書き込み	List (View)	なし
accountId	読み取り / 書き込み	String	なし
accountPolicy	読み取り / 書き込み	String	なし
adapterClassName	読み取り / 書き込み	String	あり
allowedApprovers	読み取り	List (String)	なし
allowedApproversIds	読み取り	List (String)	なし
approvers	読み取り / 書き込み	List (String)	なし
available	読み取り	View	なし
description	読み取り	String	なし

表 3-61　リソースビューの属性 (続き)

属性	編集の可能性	データ型	必要性
displayName	読み取り	String	なし
excludedAccountsRule	読み取り / 書き込み	String	なし
facets	読み取り	String	なし
identityTemplate	読み取り / 書き込み	String	なし
name	読み取り / 書き込み	String	あり
organizations	読み取り / 書き込み	List (String)	あり
passwordPolicy	読み取り / 書き込み	String	なし
resourceAttributes	読み取り / 書き込み	List (View)	なし
resourcePasswordPolicy	読み取り / 書き込み	String	なし
retryMax	読み取り / 書き込み	Integer	なし
retryDelay	読み取り / 書き込み	Integer	なし
retryEmail	読み取り / 書き込み	String	なし
retryEmailThreshold	読み取り / 書き込み	Integer	なし
startupType	読み取り / 書き込み	String	なし
syncSource	読み取り / 書き込み	Boolean	なし
typeDisplayString	読み取り / 書き込み	String	あり
typeString	読み取り / 書き込み	String	あり

accountAttributes

このリソースで管理の対象となるアカウントを指定します。属性はリソースタイプによって異なり、スキーママップに直接関連付けられます。このリストの各要素は、resourceAttributes に指定されるリスト内の要素に関連付けられます。

このリストの各要素には、次の属性があります。

表 3-62　accountAttribute リソースビュー属性の属性

属性	データ型	説明
attributeName	String	Identity Manager のフォームとワークフローに表示される属性の名前を指定します。
syntax	String	値の型を宣言します。有効な値は string、int、boolean、encrypted、または binary です。

表 3-62　accountAttribute リソースビュー属性の属性 (続き)

属性	データ型	説明
name	String	自動生成される値を指定します。この値は無視してください。
mapName	String	リソースアダプタによって認識される属性の名前を指定します。
required	Boolean	true の場合、このアカウント属性は必須です。
audittable	Boolean	true の場合、このアカウント属性はユーザーイベントの監査時に常に監査されるはずです。
multi	Boolean	true の場合、このアカウント属性に複数の値が含まれる可能性があります。
ordered	Boolean	true の場合、アカウント属性の値の順序を保持する必要があります。
readonly	Boolean	true の場合、このアカウント属性は読み取りのみが許可され、変更できません。
writeonly	Boolean	true の場合、このアカウント属性は書き込みのみが許可され、読み取ることはできません。

accountId

リソースがこのアカウントを識別するための ID を指定します。

accountPolicy

このリソースのアカウント ID に適用されるポリシーを指定します。

adapterClassName

リソースへのプロビジョニングで使用されるリソースアダプタクラスを指定します。

allowedApprovers

(読み取り専用の計算値) リソースの承認を実行する権限を持つユーザーの表示名を指定します。表示属性として使用されるユーザー属性を指定するときは、UserUIConfig オブジェクトを編集します。Identity Manager では、管理者の name 属性がデフォルトで使用されます。

allowedApproversIds

(読み取り専用の計算値) allowedApprovers に使用される表示属性が name 以外の名前の場合にだけ計算されます。

approvers

このリソースを承認する管理者のリスト。

available

次の表に記載されている available 属性を指定します。

表 3-63 リソースビューの available 属性の属性

available 属性の属性	説明
available.formFieldNames	「*global.*」または「*accounts[<resourcename>].*」で始まる属性の名前を指定します。これらの属性は、スキーママップの左側の名前のオプション名ドロップダウンリストに取り込まれます。
available.extendedAttributes	#ID#Configuration:UserExtendedAttributes 設定オブジェクトから読み取られる属性を指定します。これらの属性は、スキーママップの左側の名前のオプション名ドロップダウンリストに取り込まれます。

description

リソースを説明するテキストを指定します。

displayName

Identity Manager のユーザー編集ページおよびパスワードページに表示される名前を指定します。

excludedAccountsRule

アカウントリストからリソースアカウントを除外するためのポリシーを指定します。

facets

`provision`、`activesync`、または `none` の値で構成されるコンマ区切り値のリスト。この文字列に `activesync` が含まれる場合は、そのリソースで ActiveSync 処理が有効になっています (つまり、無効になってはいない)。この文字列に `provision` が含まれる場合は、Identity Manager に接続関連の基本的なリソースパラメータが表示されます。

identityTemplate

このリソースでのユーザーのアイデンティティーの生成に使用される、アイデンティティーテンプレートを指定します。

name

リソースを外部的に識別します。ユーザーが指定するこの名前は、リソースオブジェクト間で重複しません。

organizations

リソースで使用できる組織を指定します。

passwordPolicy

このリソースのアカウントに適用されるパスワードポリシーを指定します。

resourceAttributes

ビューを指定します。このリストの各要素には、次の属性があります。

一部の属性は、設定されているアダプタの種類によって異なります。これらの属性は、少なくともリソースとの接続方法を指定します。

次の属性は、リソースオブジェクトを一意に識別します。

表 3-64

属性	データ型	説明
name	String	属性名を指定します。
displayName	String	表示用の I18N 対応ラベルを指定します。
type	String	値の型を宣言します。有効な値は `string`、`int`、`boolean`、`encrypted`、または `binary` です。
multivalued	Boolean	true の場合、この属性に複数の値を含めることができます。
description	String	この属性の目的を説明するヘルプテキストを指定します。
noTrim	Boolean	true の場合、先行する空白および末尾の空白が削除されます。
provision	Boolean	true の場合、これは標準設定属性です。
activesync	Boolean	true の場合、この属性は ActiveSync を設定するときに必要になります。

表 3-64　(続き)

属性	データ型	説明
value	Object または ListObject	現在の値

例 : `<Field name='resourceAttributes[Display Name Attribute].value'>`

resourcePasswordPolicy

このリソースのリソースアカウントに適用されるリソースパスワードポリシーを指定します。

retryMax

リソース上のオブジェクトの管理時に発生するエラーに対して、再試行を行う最大回数を指定します。

retryDelay

再試行の間隔を秒単位の時間で指定します。

retryEmail

通知の再試行しきい値に達した場合に送信される通知の、送信先電子メールアドレスを指定します。

retryEmailThreshold

電子メールの送信後の再試行回数を指定します。

startupType

activeSync リソースの起動を自動で行うか、手動で行うかを指定します。

syncSource

true に設定すると、リソースが同期化イベントをサポートすることを表します。

typeDisplayString

リソースタイプの表示名を指定します。これは、メッセージカタログに含まれるメッセージキーまたは ID です。

typeString

リソースタイプの内部名を指定します。

リソースオブジェクトビュー

リソースオブジェクトの変更に使用されます。

更新する属性レベルの変更の計算に使用される <resourceobjectType>.oldAttributes を除くすべての属性は、編集可能です。

実際には、<resourceobjectType> を、リソースに固有のオブジェクトタイプの小文字の名前に置き換えます (たとえば、group、organizationalunit、organization、または role)。

表 3-65　リソースオブジェクトビューの属性

属性	編集の可能性	データ型	必要性
resourceType	読み取り / 書き込み	String	
resourceName	読み取り / 書き込み	String	
resourceId	読み取り / 書き込み	String	
objectType	読み取り / 書き込み	String	
objectName	読み取り / 書き込み	String	
objectId	読み取り / 書き込み	String	
requestor	読み取り / 書き込み	String	
attributes	読み取り / 書き込み	Object	
oldAttributes	読み取り	Object	
organization	読み取り / 書き込み	String	
attrstoget	読み取り / 書き込み	List	
searchContext	読み取り / 書き込み	Object	
searchAttributes	読み取り / 書き込み	List	

<resourceobjectType>.resourceType

Identity Manager のリソースタイプ名のリスト (たとえば、LDAP、Active Directory)。

<resourceobjectType>.resourceName

Identity Manager のリソース名のリスト。

<resourceobjectType>.resourceId

Identity Manager のリソース ID または名前のリスト。

<resourceobjectType>.objectType

リソースに固有のオブジェクトタイプ (たとえば、`Group`) を指定します。

<resourceobjectType>.objectName

リソースオブジェクト名のリスト。

<resourceobjectType>.objectId

リソースオブジェクト (たとえば、`dn`) の完全修飾名を指定します。

<resourceobjectType>.requestor

ビューを要求しているユーザーの ID を指定します。

<resourceobjectType>.attributes

新しい、または更新されたリソースオブジェクト属性の名前と値のペア (オブジェクト) を指定します。この属性は、次の下位属性を持ちます。

`resourceattrname` -- 指定されたリソース属性の値を取得または設定するための文字列 (たとえば、<*objectType*>`.attributes.cn`、この `cn` はリソース属性 `common name`)。

<resourceobjectType>.oldAttributes

フェッチされたリソースオブジェクト属性の名前と値のペア (オブジェクト) を指定します。この値を編集することはできません。ビューはこの属性を使用して、更新の属性レベルの変更を計算します。

<resourceobjectType>.organization

リソースがメンバーとして属す組織のリストを指定します。このリストは、将来の分析やレポート生成のために用意されている関連監査イベントレコードにアクセスできる組織の特定に使用されます。

<resourceobjectType>.attrstoget

`checkoutView` メソッドまたは `getView` メソッドを使用してオブジェクトを要求したときに返される、オブジェクトタイプに固有の属性のリスト。

<resourceobjectType>.searchContext

階層的なネームスペースを使用して、リソース内の不完全な修飾名を検索するためのコンテキストを指定します。

<resourceobjectType>.searchAttributes

階層的なネームスペースを使用して、指定された `searchContext` 内でリソース名を検索するときに使用される、リソースオブジェクトタイプに固有の属性名のリスト。

<resourceobjectType>.searchTimelimit

フォームに入力された名前を検索する最大時間 (リソースがサポートする場合) を指定します。

ロールビュー

Identity Manager のロールオブジェクトの定義に使用されます。

このビューがチェックインされると、管理ロールワークフローが呼び出されます。デフォルトでは、このワークフローは単にビューの変更をリポジトリに適用するだけですが、承認や、その他のカスタマイズのきっかけを提供することもできます。

次の表は、このビューの上位属性を示しています。

表 3-66 ロールビューの属性

属性	編集の可能性	データ型	必要性
applications	読み取り / 書き込み	List	なし
approvers	読み取り / 書き込み	List	なし
approversRule	読み取り / 書き込み	String	なし
assignedResources	読み取り / 書き込み	List	なし
containedRoles	読み取り / 書き込み	List	なし
description	読み取り / 書き込み	String	なし
disabled	読み取り / 書き込み	Boolean	なし
name	読み取り / 書き込み	String	あり
notifications	読み取り / 書き込み	List	なし
notificationsRule	読み取り / 書き込み	String	なし
organizations	読み取り / 書き込み	List	あり
owners	読み取り / 書き込み	List	なし
ownersRule	読み取り / 書き込み	String	なし
properties	読み取り / 書き込み	List	なし
resources	読み取り / 書き込み	List	なし
roles	読み取り / 書き込み	List	なし
type	読み取り / 書き込み	String	なし
types	読み取り	List	なし

applications

ローカルに割り当てられたアプリケーションの名前 (リソースグループ) を指定します。

approvers

ユーザーへのこのロールの割り当てを承認する承認者の名前を指定します。

approversRule

このロールがユーザーに割り当てられるときおよびプロビジョニングされるときに、承認者である 1 人以上のユーザーのリストを返すための規則を指定します。

assignedResources

リソース、リソースグループ、およびロールによって割り当てられたすべてのリソースの、フラット化されたリスト。

表 3-67 assignedResource 属性の属性 (ロールビュー)

属性	編集の可能性	データ型
resourceName	なし	String
name	なし	String
属性	なし	Object

resourceName

割り当てられているリソースの名前を指定します。

name

リソースの名前または ID (可能な限り ID) を指定します。

属性

リソースの特性を指定します。すべての下位属性は文字列であり、編集可能です。

表 3-68 attribute 属性のオプション (ロールビュー)

属性	説明
name	リソース属性の名前
valueType	この属性に設定される値のタイプ。Rule、text、none などの値を指定できます。

表 3-68　attribute 属性のオプション (ロールビュー) (続き)

属性	説明
requirement	この属性によって設定される値のタイプ。指定できる値には、Default value (デフォルト値)、Set to value (値を設定)、Merge with Value (値とマージ)、Remove from Value (値から削除)、Merge with Value clear existing (値とマージ、既存の値をクリア)、Authoritative set to value (強制的に値を設定)、Authoritative merge with value (強制的に値とマージ)、Authoritative merge with value clear existing (強制的に値とマージ、既存の値をクリア) があります。
rule	値のタイプが Rule である場合に、規則名を指定します。
value	値のタイプが Text である場合に、値を指定します。

containedRoles

含まれるロールの情報が含まれるオブジェクトを指定します。

表 3-69　containedRoles 属性の属性 (ロールビュー)

属性	編集の可能性	データ型
name	なし	String
info	なし	String
associationType	あり	String
approvalRequired	あり	Boolean
condition	あり	Object

name

ロール名を指定します。

info

ロールの情報 (description、id、name、noApprovers、および type) を指定します。

associationType

関連付けが required、conditional、または optional のいずれであるかを指定します。

approvalRequired

associationType が optional の場合は、このロールがユーザーによって要求されたときに承認が必要かどうかを示す、ブール型のフラグです。

condition

`associationType` が conditional の場合は、このロールが特定のユーザーに割り当てられるかどうかを決定する条件です。

description

このロールを説明します。

disabled

指定されたロールが無効かどうかを指定します。デフォルト値は `false` です。

name

ロールの名前を指定します。これは、Identity Manager リポジトリ内の Role オブジェクトの名前に対応します。

notifications

ユーザーへのこのロールの割り当てを承認する管理者の名前のリスト。

notificationsRule

このロールがユーザーに割り当てられるときおよびプロビジョニングされるときに、通知される 1 人以上のユーザーのリストを返すための規則を指定します。

organizations

このロールがメンバーとして属す組織のリスト。

owners

このロールへの変更の承認者として指定された 1 人以上のユーザーを指定します。

ownersRule

このロールへの変更の承認者である 1 人以上のユーザーのリストを返すための規則を指定します。

properties

このロールに格納される、ユーザー定義のプロパティーを指定します。

resources

ローカルに割り当てられたリソースの名前を指定します。

roles

ローカルに割り当てられたロールの名前を指定します。

type

ロール設定オブジェクトに定義されているこのロールのタイプを識別します。

types

ビューで使用するためにロール設定オブジェクトからキャッシュされたタイプ情報 (読み取り専用)。

タスクスケジュールビュー

TaskSchedule オブジェクトの作成と変更に使用されます。

このビューは、次の属性を持ちます。

表 3-70　タスクスケジュールビューの属性

名前	編集の可能性	データ型	必要性
scheduler	読み取り / 書き込み	String	
task	読み取り / 書き込み	Boolean	

scheduler

スケジューリングされているすべてのタスクに共通する、スケジューラ自体に関連する属性を持ちます。次の属性があります。

表 3-71　scheduler 属性の属性 (タスクスケジュールビュー)

名前	編集の可能性	データ型	必要性
name	読み取り / 書き込み	String	なし
id	読み取り	String	なし
definition	読み取り / 書き込み	String	なし
template	読み取り / 書き込み	String	なし
taskOrganization	読み取り / 書き込み	String	なし
taskName	読み取り / 書き込み	String	なし
description	読み取り / 書き込み	String	なし
disabled	読み取り / 書き込み	Boolean	なし
skipMissed	読み取り / 書き込み	Boolean	なし
start	読み取り / 書き込み	Date	なし
repeatCount	読み取り / 書き込み	Int	なし
repeatUnit	読み取り / 書き込み	String	なし
resultOption	読み取り / 書き込み	String	なし
allowMultiple	読み取り / 書き込み	Boolean	なし

注	通常は、scheduler.definition または scheduler.template を値として指定します。いずれの値も指定しない場合は、後から編集して定義またはテンプレートを指定できる TaskSchedule オブジェクトが作成されます。

name

既存の TaskSchedule オブジェクトの名前、または新しい TaskSchedule オブジェクトの適切な名前を指定します。これは必須属性ではありませんが、指定しない場合は、システムによってランダムな識別子が生成されます。

id

既存の TaskSchedule オブジェクトを一意に識別します。

definition

スケジューリングする TaskDefinition オブジェクトの名前を指定します。

template

スケジューリングする TaskTemplate オブジェクトの名前を指定します。definition と template の両方を指定した場合は、template が優先されます。

taskOrganization

スケジュールタスクの呼び出し時に TaskInstance が配置される組織の名前を指定します。

taskName

スケジュールタスクの呼び出し時に作成される TaskInstance の名前を指定します。

description

スケジュールタスクの呼び出し時に作成される TaskInstance に保存される、説明テキストを指定します。この説明は、製品インタフェースのタスクテーブルに表示されます。

disabled

タスクスケジューラが TaskSchedule オブジェクトを処理するかどうかを制御します。disable 属性が true に設定された TaskSchedule は、スケジューラによって無視されます。この属性を使用することで、TaskSchedule オブジェクトを削除して再作成することなく、スケジュールタスクの実行を一時的に停止できます。

start

タスクを呼び出す日時を指定します。

repeatCount

`repeatUnit` と組み合わせて、タスクの実行頻度を決定します。repeatCount をゼロに設定するか、または設定しない場合、スケジューリングされたタスクの実行は 1 回のみです。repeatCount に正の値を指定した場合は、タスクは repeatUnit に指定された間隔で複数回実行されます。

repeatUnit

`repeatCount` に正の値が指定されたタスクの実行間隔を指定します。有効な値は、`second`、`minute`、`hour`、`day`、`week`、`month` です。たとえば、1 年間を通じて週に 1 回実行するようにタスクをスケジューリングする場合は、`repeatUnit` を week に、`repeatCount` を 52 に設定し、タスクの実行を開始する日付を `start` に設定します。

resultOption

スケジューリングされたタスクの実行時に、指定された名前の TaskInstance がすでに存在する場合のスケジューラの反応を指定します。有効な値は、`wait`、`delete`、`rename`、および `terminate` です。

wait

スケジューラがタスクを再実行するか、次の繰り返しを待機するかを指定します。この属性が有効となるのは、`repeatCount` と `repeatUnit` が設定されている場合のみです。

delete

既存の TaskInstance の実行が完了している場合に、スケジューラはそのインスタンスを削除します。

rename

既存の TaskInstance の実行が完了している場合に、スケジューラはそのインスタンスの名前を変更します。

skipMissed

スケジューリングされた時刻にタスクを実行できなかった場合に、それを回復するために Identity Manager がタスクをただちに試行するか (`false`)、スケジューリングされている次回の時刻まで単に待機するか (`true`) を指定します。

false に設定した場合は、Identity Manager はスケジューリングされた時刻に実行できなかったタスクの実行をただちに試行します。true に設定した場合は、Identity Manager はスケジューリングされている次回の時刻まで待機します。デフォルトは false です。

terminate

delete に似ていますが、既存のタスクが現在も実行中の場合は、そのタスクを終了します。

allowMultiple

同じタスク定義またはタスクテンプレートの、複数のインスタンスの実行を許可するかどうかを制御します。true (デフォルト) に設定した場合は、スケジューラはタスクの新しいインスタンスを常に作成します。false に設定した場合は、すでに実行中のインスタンスが存在するとき、スケジューラは新しいインスタンスを作成しません。

task

タスクに固有の属性を持ちます。各タスクは独自の属性を定義し、タスクのフォームは、task ネームスペースに相対的な属性としてそれらを参照します。

ロック解除ビュー

リソース側の機能としてアカウントのロックがサポートされている場合、それらのリソースに対するアカウントのロック解除に使用されます。このビューは、ロック解除するリソースアカウントのリストの表示と選択に使用されます。

注 リソースがアカウントのネイティブロックをサポートするアカウントでは、無効化ビューの代わりにロック解除ビューを使用します。

次の上位属性を持ちます。

表 3-72 ロック解除ビューの属性

名前	編集の可能性	データ型	必要性
id	読み取り	String	あり
selectAll	読み取り / 書き込み	Boolean	なし
currentResourceAccounts	読み取り	List (オブジェクト)	なし
tobeCreatedResourceAccounts	読み取り	List (オブジェクト)	なし
tobeDeletedResourceAccounts	読み取り	List (オブジェクト)	なし
fetchAccounts	読み取り / 書き込み	Boolean	
fetchAccountResources	読み取り / 書き込み	List	

id

パスワードのロック解除の対象となる Identity Manager ユーザーのアカウント ID を指定します。

selectAll

すべてのパスワードのロックを解除するかどうかを制御します。

currentResourceAccounts

Identity Manager によって現在管理されているアカウントのセットを表します (Identity Manager アカウント自体を含む)。

tobeCreatedResourceAccounts

この Identity Manager ユーザーに割り当てられているが、まだ作成されていないアカウントを表します。まだ作成されていないアカウントでは、パスワードのロックを解除することはできません。

tobeDeletedResourceAccounts

すでに作成されているが、このユーザーには割り当てられなくなったアカウントを表します。削除されるアカウントでは、パスワードを変更することはできません。

3 つのアカウントリストはすべて、各リソースのアカウントの状態を表現するオブジェクトを含み、ユーザーはアカウントを個別に選択できます。

どちらのリソースアカウントリストにもリソース名によるインデックスが付けられ、このユーザーがアカウントを持つリソースを表現するオブジェクトを含みます。

表 3-73 tobeDeletedResourceAccounts の属性 (ロック解除ビュー)

名前	編集の可能性	データ型
`selected`	読み取り / 書き込み	Boolean
`name`	読み取り / 書き込み	String
`type`	読み取り / 書き込み	String
`accountId`	読み取り / 書き込み	String
exists	読み取り / 書き込み	Boolean
`locked`	読み取り / 書き込み	Boolean
`authenticator`	読み取り / 書き込み	Boolean

selected

ロック解除の対象として、このリソースが選択されていることを示します。

name

リソースの名前を指定します。これは、Identity Manager リポジトリ内の `resource` オブジェクトの名前に対応します。

type

リソースのタイプ (たとえば、Solaris) を指定します。Identity Manager の管理者インタフェースでこのリソースリストを表示することで、リソースタイプの名前を確認できます。このページの「**タイプ**」列には、現在定義されているリソースのタイプ名が表示されます。「**新規リソース**」の横のオプションリストにも、現在インストールされているリソースアダプタの名前が表示されます。

accountId

このリソースにアカウントが作成されている場合は、そのアカウントの ID を指定します。

exists

リソースにすでにアカウントが存在するかどうかを示します (`currentResourceAccounts` のみ)。

locked

アカウントが現在ロックされているか、ロック解除されているかを示します。`exists` の値は、リソースにすでにアカウントが存在するかどうかを示します (`currentResourceAccounts` のみ)。

authenticator

`true` に設定した場合は、このリソースが Identity Manager のパススルー認証リソースとして機能することを示します。

fetchAccounts

ビューに、ユーザーに割り当てられているリソースのアカウント属性を含めます。

詳細については、この章の「フォームでのビューオプションの設定」を参照してください。

fetchAccountResources

フェッチ先リソースの名前のリスト。指定しない場合は、すべてのリソースが使用されます。

詳細については、この章の「フォームでのビューオプションの設定」を参照してください。

ユーザーエンタイトルメントビュー

ユーザーエンタイトルメントオブジェクトを作成および変更するときに使用します。このビューには、次の最上位属性があります。

表 3-74　ユーザーエンタイトルメントビューの最上位属性

名前	編集の可能性	データ型	必要性
name		String	あり
status		String	あり
user		String	あり
userId		String	あり
attestorHint		String	なし
userView		GenericObject	あり
reviewInstanceId		String	あり
reviewStartDate		String	あり
scanId		String	あり
scanInstanceId		String	あり
approvalWorkflowName		String	あり
organizationId		String	あり
attestorComments.name		String	なし
attestorComments.attestor		String	なし
attestorComments.time		String	なし
attestorComments.timestamp		String	なし
attestorComments.status			なし

name

一意識別子によってユーザーエンタイトルメントを識別します。

status

ユーザーエンタイトルメントオブジェクトの状態を指定します。有効な状態は PENDING、ACCEPTED、REJECTED、REMEDIATING、CANCELLED です。

user

このエンタイトルメントに関連付けられている WSUser の名前を識別します。

userId

関連付けられている WSUser の ID を指定します。

attestorHint

レビュー決定規則によって提供されるアテスターへのヒント (String) を表示します。このヒントは、規則からアテスターへの「アドバイス」になります。

userView

ユーザーエンタイトルメントスキャナによって収集されるユーザービューが入ります。アクセススキャンオブジェクトの設定に応じて、このビューにはゼロ個以上のリソースアカウントが取り込まれます。

reviewInstanceId

PAR タスクインスタンスの ID を指定します。

reviewStartDate

PAR タスクの開始日 (String) を正規の書式で指定します。

scanId

アクセススキャンタスク定義の ID を指定します。

scanInstanceId

アクセススキャンタスクインスタンスの ID を指定します。

approvalWorkflowName

承認のために実行するワークフローの名前を識別します。この値は、アクセススキャンタスク定義から取得されます。

organizationId

スキャンしたときに WSUser が属していた組織の ID を指定します。

attestorComments

エンタイトルメントのアテステーションレコードを指定します。各アテステーションレコードによって、そのエンタイトルメント (承認、却下、再スキャンなど) で実行されたアクションまたはステートメントがわかります。

attestorComments[timestamp].name

リスト内でこの要素を識別するために使用されるタイムスタンプ。

attestorComments[timestamp].attestor

そのエンタイトルメントのコメントを作成するアテスターの WSUser 名を識別します。

attestorComments[timestamp].time

アテスターがこのレコードをアテストした時刻を指定します。タイムスタンプと一致していなくてもかまいません。

attestorComments[timestamp].status

アテスターによって割り当てられた状態を指定します。任意の文字列を指定できますが、通常はアテスターが選択したアクションを示す文字列 (承認、却下、再スキャン、是正など) を指定します。

attestorComments[name].comment

アテスターによって追加されたコメントが入ります。

作業項目ビュー

リポジトリ内の WorkItem オブジェクトの表示と変更に使用されます。

WorkItem オブジェクトは、ワークフロープロセスに定義されている手動アクションが作動するたびに作成されます。作業項目ビューには、WorkItem オブジェクト自体を表現するいくつかの属性と、ワークフロータスクからコピーされた、選択されたワークフロー変数の値が含まれます。

Identity Manager は、`workItem.related` 属性の下の作業項目ビューに含まれる、作業項目に関する情報を返します。

すべてのアクティブ作業項目に関する情報の取得

このビューでは、ワークフロータスクで現在アクティブなすべての作業項目に関する情報を取得できます。デフォルトでは、関連するすべての作業項目ではなく、指定された作業項目のみに関する情報が返されます。ただし、別のオプションを使用して、表示する作業項目をフィルタリングしたり、作業項目に関連する属性をフィルタリングしたりすることができます。

このビューのデフォルトの動作を変更するには、次の 3 つのフォームプロパティーを使用します。

表 3-75

実行する処理	使用するフォームプロパティー
デフォルトで関連するすべての項目を取得する	`includeRelatedItems` フォームプロパティー
取得する追加属性を要求する	`relatedItemAttributes` フォームプロパティー
取得する項目を制限する	`relatedItemFilter` フォームプロパティー

例 : includeRelatedItems フォームプロパティーの使用

デフォルトでは、Identity Manager は作業項目の表示に承認フォームを使用します。このフォームを編集し、関連作業項目を含めるための `includeRelatedItems` 要素を追加します。

```
<Properties>

   <Property name='includeRelatedItems' value='true'/>
</Properties>
```

例 : relatedItemAttributes フォームプロパティーの使用

relatedItemAttributes オプションを使用して、追加属性を要求することもできます。このオプションには、名前の CSV 文字列、または名前のリストを指定できます。要求できる標準属性は次のとおりです。

- request
- requester
- description
- activityName

このリストに含まれない属性名を要求した場合は、Identity Manager はそれを任意のワークフロー変数と見なします。作業項目にその編集が存在する場合は、その値が返されます。標準ワークフローに共通する変数は次のとおりです。

- accountId
- objectType
- objectName
- diagramLabel

例 : includeRelatedItems フォームプロパティーの使用

request 属性と description 属性を含めるには、承認フォームに次のプロパティーを追加します。

```
<Properties>

   <Property name='includeRelatedItems' value='true'/>

   <Property name='relatedItemAttributes'
value='request,description'/>

</Properties>
```

例 : relatedItemFilter フォームプロパティーの使用

指定できるフィルタ属性は次のとおりです。

表 3-76　relatedItemFilter オプションの値

relatedItemFilter オプションの値	フィルタリングの結果
itemType	itemType が一致する作業項目のみが返されます
activityName	同じアクティビティーから作成された作業項目のみが返されます

表 3-76　relatedItemFilter オプションの値 (続き)

relatedItemFilter オプションの値	フィルタリングの結果
request	同じユーザー定義要求文字列を持つ作業項目のみが返されます
locked	現在編集がロックされている作業項目のみが返されます

リストに複数のフィルタ属性が含まれる場合は、それらは論理 AND で連結されます。たとえば、同じ要求文字列を持ち、現在ロックされている作業項目のみを取得するときは、承認フォームに次のプロパティーを追加します。

```
<Properties>
   <Property name='includeRelatedItems' value='true'/>
     <Property name='relatedItemAttributes' value='request,description'/>
     <Property name='relatedItemFilter' value='request,locked'/>
</Properties>
```

関連作業項目に関する情報のテーブルを示すフィールド例が承認ライブラリフォームライブラリに追加され、フィールド名が「Related Approvers」であるとします。標準の認証フォームからこのフィールドを参照するには、次のように指定します。

```
<FieldRef name='Related Approvers'/>
```

作業項目のリポジトリロックタイムアウトの変更

リポジトリ内の作業項目のロックに適用されるデフォルトの間隔は 5 分間です。RepositoryConfiguration 設定オブジェクトの RelocatedTypes 要素に次の要素を追加することで、この値を変更できます。

```
<TypeDataStore typeName='WorkItem' lockTimeoutMillis='10000'/>
```

最上位属性

次の表は、作業項目ビューの最上位属性を示しています。

表 3-77　作業項目ビューの属性

属性	編集の可能性	データ型	必要性
id	読み取り	String	
name	読み取り	String	

表 3-77　作業項目ビューの属性 (続き)

属性	編集の可能性	データ型	必要性
taskId	読み取り	String	
taskName	読み取り	String	
processName	読み取り	String	
activityName	読み取り	String	
description	読み取り / 書き込み	String	
owner	読み取り / 書き込み	String	
complete	読み取り / 書き込み	Boolean	
variables			
workItem			

id

WorkItem オブジェクトのリポジトリ ID を指定します。この ID は、通常、Identity Manager によって生成され、表示されません。

name

WorkItem オブジェクトのリポジトリ名を指定します。

taskId

ワークフローの TaskInstance のリポジトリ ID を指定します。この属性は、システムが作業項目とワークフロータスクを関連付けるときに使用され、変更できません。

taskName

ワークフローの TaskInstance のリポジトリ名を指定します。この名前は、通常は説明的な値に設定され、表示可能です。この値を変更しないでください。ユーザー更新の一般的なタスク名は、たとえば `Updating User jdoe` となります。

processName

手動アクションを含むワークフロープロセス定義の名前を指定します。

activityName

手動アクションを含むワークフローアクティビティーの名前を指定します。

description

作業項目を説明するテキストを指定します。この内容は、ワークフロープロセス定義によって設定されます。この説明は、通常は作業項目の概要を示すテーブルに表示され、作業項目フォームにも表示されます。

owner

ワークフロープロセスを作成した、現在の Identity Manager 管理者またはユーザーの名前を指定します。この属性の値は、通常は Identity Manager ユーザーの名前です。この作業項目が匿名ユーザーに割り当てられている場合は、名前に **Temp:** というプレフィックスが付けられます。

complete

手動アクションが完了したときにワークフローを再開する場合は、`true` に設定します。complete 属性の割り当ては、作業項目フォームで実行してください。

このブール値は編集可能です。

variables

ワークフロータスクからコピーされた変数を含む属性を持つ、別のオブジェクトを指定します。デフォルトでは、手動アクションの作動時に範囲に含まれるすべてのワークフロー変数が、作業項目にコピーされます。これは、プロセス定義の Exposed Variables オプションと Editable Variables オプションで制御できます。ほとんどの作業項目は、検出された情報を `variables` 属性の下に表示します。この属性の使用の詳細については、この章で後述する「variables 属性の使用」の節を参照してください。

workItem

作業項目に関する追加情報を指定します。含まれる属性は次のとおりです。

ビュー

views という値を持つワークフロー変数のリストが含まれます。システムはこの属性を使用して、作業項目ビューの表示更新時に、ビューに固有の表示更新処理を行います。

この値を変更しないでください。

related

指定された作業項目を表現する属性のリストを含みます。

表 3-78　workItem.related 属性の下位属性 (作業項目ビュー)

属性	説明
name	作業項目のリポジトリ ID を指定します。
owner	項目の所有者を指定します。
locked	作業項目が編集中であるかどうかを指定します。true という値は、作業項目が現在編集中であることを示します。
complete	作業項目が完了しているかどうかを指定します。true という値は、作業項目が完了していることを示します。
itemType	プロセスによって定義される項目タイプを指定します。デフォルトは approval です。

request

作業項目の目的を簡単に説明します。この説明は、通常、description 属性の値より短く、多くの場合は概要テーブルに表示されます。

requester

承認を開始したユーザーを指定します。

ignoreTimeOut

タイムアウトを無視するかどうかを指定します。システムによって割り当てられる true という値は、これが読み取り専用の作業項目であり、表示中にタイムアウトになる可能性があることを示します。これは、作業項目がすでに存在しない場合に、エラーメッセージを表示する代わりに、無視する作業項目ビューのチェックイン失敗をシステムに伝えます。これは、ユーザーにはメッセージを表示しながらワークフローを継続できるように、ただちにタイムアウトになる状態メッセージのみを目的とした作業項目で便利です。

この値を変更しないでください。

variables 属性の使用

作業項目フォームの記述時にもっともよく参照される属性は、complete と variables です。ワークフローを再開するには、complete 属性の値が true に設定されていなければなりません。多くの場合、これは「**承認**」や「**却下**」などのラベルが付けられた押しボタンフィールドへの応答として、非表示フィールドによって設定されます。

variables 属性には、ワークフロータスクからコピーされた変数を値として持つオブジェクトが含まれます。作業項目で頻繁に使用されるワークフロー変数の 1 つに user があります。この変数には、ユーザービューが含まれます。たとえば、作業項目フォームから global.email 属性を参照するには、次のパス式を使用します。

```
variables.user.global.email
```

これは、標準のユーザーフォームで使用される属性パスとは異なります。まず、user というワークフロー変数にビュー全体が格納されます。このため、属性パスに user. というプレフィックスが必要になります。次に、ワークフロー変数が作業項目ビューの variables 属性の下に格納されます。このため、属性パスに variables. という追加プレフィックスが必要になります。

ユーザービュー属性のこの入れ子構造のため、作業項目ビューでは、修正なしで標準のユーザーフォームを使用することはできません。ただし、base context オプションを使用して、ユーザーフォームを参照する作業項目フォームを定義することができます。

例

```
<Form name='WorkItemForm'>
   <Include>
      <ObjectRef Type='UserForm' name='Default User Form'/>
   </Include>
   <FormRef name='Default User Form' baseContext='variables.user'/>
</Form>
```

注 実際には、作業項目フォームは「承認」や「却下」などのボタンの追加フィールドを必要としますが、作業項目フォームに表示されるデフォルトのユーザーフォームに、一部の項目を表示しないようにすることが必要となる場合もあります。通常は、ユーザーフォームと作業項目フォームの両方で参照できるフォームライブラリに、ユーザーフォームから除外するフィールドを指定できます。

作業項目リストビュー

リポジトリ内の作業項目の集合に関する情報の表示と、複数の作業項目に対する処理の同時実行に使用されます。

このビューのハンドラは、次の情報を収集します。

- 選択されたユーザーに割り当てられているすべての作業項目
- 表示可能な作業項目が割り当てられているユーザー
- 作業項目の転送先にできるユーザー

このビューは、Identity Manager の管理者インタフェースの「承認」ページで使用されます。このビューで使用されるデフォルトフォームの名前は、「作業項目リスト」です。

次の表は、作業項目リストビューの最上位属性を示しています。

表 3-79 作業項目ビューの属性

属性	編集の可能性	データ型
authType	読み取り / 書き込み	String
userId	読み取り	String
user	読み取り / 書き込み	String
self	読み取り	Boolean
forwardedUser	読み取り	Boolean
itemType	読み取り / 書き込み	String
users	読み取り	List
userIds	読み取り	String
forwardingApproverStyle	読み取り	
forwardingUsers	読み取り	List
forwardingUserIds	読み取り	List
workItems	読み取り / 書き込み	String
selectedWorkItems	読み取り / 書き込み	String
forwardTo	読み取り / 書き込み	Boolean
forwardToNow	読み取り / 書き込み	String
variables	読み取り / 書き込み	String
action	読み取り / 書き込み	Boolean

表 3-79　作業項目ビューの属性 (続き)

属性	編集の可能性	データ型
confirm	読み取り / 書き込み	Boolean

authType

作業項目へのアクセスをタイプによって指定します。たとえば、EndUserRule という認証タイプが組み込まれています。すべてのエンドユーザーは、EndUserRule という認証タイプがタグ付けされたすべての規則へのアクセスを暗黙的に取得します。

userId

workItem リストに作業項目が含まれる Identity Manager ユーザーの名前を指定します。この属性の初期値は、現在のセッションユーザーの名前です。NULL に設定し、認証権限を持つすべての制御対象ユーザーの作業項目を表示することもできます。この属性の値は常に Identity Manager のユーザー名であり、表示名ではありません。

フォームでこの値を変更しないでください。ユーザーを変更するには、`user` 属性を設定します。

user

作業項目が一覧表示される Identity Manager ユーザーの表示名を指定します。表示名を使用していない場合、この値は `userId` と同じになります。フォームでこの値を変更すると、システムは表示更新時に作業項目リストを再計算します。NULL 値は、すべての作業項目の表示を示します。

self

`userId` が現在のセッションユーザーと同じ場合は、true に設定します。

forwardedUser

この属性を設定すると、`userId` によって特定されるユーザーが、別ユーザーへの作業項目の転送を希望したことを示します。その他のユーザーは、表示名で識別されます。

users

現在のユーザーによって制御され、作業項目機能を持つ Identity Manager ユーザーの表示名のリスト。この値は、通常はユーザー選択ボックスの作成に使用されます。カスタムフォームに別の方法でユーザーリストを計算させる場合は、ビューオプションまたはフォームプロパティーとして CustomUserLists ビューオプションを指定します。

userIds

通常は NULL に設定されます。代替表示名を使用するように設定されている場合は、ユーザーリストには表示名が含まれ、このリストには本当のリポジトリ名が含まれます。

forwardingUsers

現在のユーザーが作業項目を転送できる Identity Manager ユーザーの表示名のリスト。この値は、`ForwardingApproverStyle` 属性の値によって異なります。この属性のデフォルト値は `peers` です。

itemType

これを設定した場合は、その値と一致する項目タイプを持つ作業項目のみを残すように、リストに含まれる作業項目がフィルタリングされます。これにより、作業項目リストビューは、作業項目のタイプに基づいて項目リストをフィルタリングする機能を持ちます。

forwardingUserIds

通常は NULL に設定されます。代替表示名を使用するように設定されている場合は、`forwardingUsers` リストには表示名が含まれ、このリストには本当のリポジトリ名が含まれます。

workItems

選択されたユーザーの作業項目に関する情報を持つオブジェクトのリスト。オブジェクト名は、作業項目のリポジトリ ID です。

workItems[].owner

所有者の表示名を指定します。`user` が NULL で、すべての作業項目が表示される場合にのみ設定します。

workItems[].request

要求しているオブジェクトの簡単な説明を指定します。この値は、ワークフロープロセスで実行される手動アクションの WorkItemRequest 式によって計算されます。

workItems[].requester

要求を行なったユーザーの表示名を識別します。

workItems[].description

作業項目のより詳細な説明を指定します。値は、ワークフロープロセスで実行される手動アクションの WorkItemDescription 式によって計算されます。この説明は、通常は作業項目の概要を示すテーブルに表示され、作業項目フォームにも表示されます。

workItems[].selected

個々の項目の選択フラグ。`selectedWorkItems` に代わる属性です。

selectedWorkItems

次のアクションで処理される項目を表す、作業項目 ID のリスト。作業項目オブジェクト内に設定される `selected` 属性に代わる属性です。SortingTable コンポーネントでは、こちらのほうが便利です。この属性と個別の選択フラグの両方が設定されている場合は、この属性の値が優先されます。

forwardTo

`action` 属性が `Forward` に設定されている場合の、選択されているすべての作業項目の転送先となる Identity Manager ユーザーの名前を指定します。

forwardToNow

`forwardTo` に似ていますが、これは `action` 属性でもあります。この属性は、`forwardTo` 属性と `action` 属性が個別に設定されているかのように、`forwardTo` の値をコピーし、`action=Forward` に設定した上で表示更新を行います。フォームコンポーネントでユーザを選択した直後に、フォームに転送処理を実行させるようにするときは、この属性を使用します。転送をボタンで制御する場合は、フォームコンポーネントに `forwardTo` 属性を設定し、`Forward` の `action` 値を送信するボタンを用意します。

action

(ブール型) NULL 以外の場合は、選択された作業項目に対する処理を開始します。

次に有効な値を示します。

- approve
- reject
- forward
- refresh

NoConfirm オプションが設定されている場合は、アクションはただちに開始されます。それ以外の場合は、Identity Manager は confirm 属性が true に設定されるまで待機します。通常であれば、独自の確認ページの表示がフォームによって定義されます。

confirm

(ブール型) action 属性に指定されている処理が実行可能であることを示します。

variables 属性の使用

個々の作業項目を編集するときは、承認または却下に関する追加情報を監査のためにワークフロープロセスに渡せるように、フォームは comments などの作業項目変数を設定できます。

作業項目リストビューでアクションを実行するときは、任意の作業項目変数も設定できます。variables 属性の値は、承認または却下されたときに属性が作業項目にコピーされるオブジェクトに設定できます。たとえば、variables オブジェクトが comments という属性を持つ場合、選択されるすべての作業項目とともに同じコメントが保存されます。

```
<Form name='variables.comments'>
   <Default>
<concat>
       <s>Approval performed on </s>
<invoke class='com.waveset.util.Util' name='dateToString'>
         <new class='java.util.Date'/>
         </invoke>
     </concat>
   </Default>
</Form>
```

注 実際には、作業項目フォームは「承認」や「却下」などのボタンの追加フィールドを必要としますが、作業項目フォームに表示されるデフォルトのユーザーフォームに、一部の項目を表示しないようにすることが必要となる場合もあります。通常は、ユーザーフォームと作業項目フォームの両方で参照できるフォームライブラリに、ユーザーフォームから除外するフィールドを指定できます。

ビューオプション

ビューの作成時または表示更新時に次のオプションを指定して、作業項目リストビューアの動作を制御できます。

userId

作業項目が表示される初期ユーザーの名前を指定します。この名前は、デフォルトでは現在のセッションユーザーとなりますが、このオプションによってデフォルト設定をオーバーライドできます。

CustomUserLists

`true` に設定すると、`users` リストと `forwardingUsers` リストの両方がフォームによってカスタム生成され、ビューハンドラによる生成が抑制されます。システムに多くの承認者が存在する場合は、これらのリストの生成には時間がかかります。フォームがデフォルトの `users` リストと `forwardingUsers` リストを必要としない場合は、このオプションを有効にします。

ForwardingApproverStyle

forward To リストに名前を含める管理者のタイプを指定します。この属性のデフォルト値は `peers` です。設定できる値は次のとおりです。

表 3-80　ForwardingApproverStyle ビューオプションの値

オプションの値	説明
`peers`	現在のユーザーと同じ組織レベル以上の管理者を指定します
`controlled`	現在のユーザーが制御している組織の管理者を指定します
`all`	controlled と peers の両方の管理者を指定します

このオプションとその他のビューオプションは、フォームプロパティーとして設定できます。

```
<Form...>
   <Properties>
      <Property name='ForwardingApproverStyle" value='peers'/>
   </Properties>
 ...
</Form>
```

NoUserListCache

true に設定した場合は、ビューハンドラは users リストと forwardingUsers リストをキャッシュせずに、フォームの表示を更新するたびにそれらを再計算します。ユーザーリストの計算は負荷が大きいため、通常はリストをキャッシュし、action 属性を Refresh に設定して明示的に指定した場合にのみ表示を更新します。

UserDisplayName

このオプションは、ユーザーリストで、値がリポジトリ名の代わりに使用される拡張ユーザー属性の名前に設定できます。UserUIConfig オブジェクトにも設定できますが、フォームに設定するほうが便利な場合もあります。

NoUserDisplayName

true に設定すると、UserUIConfig オブジェクトに表示名が設定されている場合でも、表示名を使用しません。このオプションをフォームに設定し、UserUIConfig の設定を選択的にオーバーライドすることができます。

NoConfirm

true に設定すると、action 属性によって指定されるアクションが、確認なしでただちに実行されます。

フォームでのビューオプションの設定

一部のフォームでは、ビューオプションを簡単に設定できます。フォームにビューオプションを設定する手順は次のとおりです。次に示す手順では、例として作業項目リストビューを使用します。

1. フォームを Identity Manager IDE または任意の XML エディタにコピーします。
2. フォーム名を変更します。
3. それを form.workItemList 属性の下の System Configuration オブジェクトに登録します。

カスタムフォームでは、次の例に示されるように、フォームのプロパティーとしてビューオプションを指定できます。

例

```
<Form>
 <Properties>
 <Property name='CustomUserLists' value='true'/>
```

```
</Properties>
 ...
</Form>
```

deferred 属性

deferred 属性は、別のアカウントの属性値から値を取得する属性です。ビュー (および WSUser モデル) で deferred 属性を宣言すると、プロビジョニングエンジンは、アダプタを呼び出す前に、この置き換えをただちに実行します。

deferred 属性が、別のリソースの GUID 属性から値を取得する場合は、ソースアダプタはアクションを実行する必要がありません。しかし、ソース属性が GUID でない場合は、realCreate 操作の二次的な影響として、アダプタは ResourceInfo._resultsAttributes マップ内の属性を返さなければなりません。アダプタが属性を返さない場合は、プロビジョニングエンジンはアカウントをフェッチして、値を取得します。これは、値を返すようにアダプタを修正するよりも非効率です。

deferred 属性を使用する状況

deferred 属性は、新しいアカウントを作成し、アカウント属性の値を、ソースアカウントが作成されるまで認識されない別のアカウントの属性の値から取得するように指定するときに使用します。一例を挙げれば、生成される一意の識別子の値を属性に設定する場合がこれに該当します。

deferred 属性の使用

deferred 属性は、主に次の 2 つの手順で定義されます。

1. アカウントは、必ず 2 番目のアカウントの作成前に、ソースリソースに作成します。これは、リソースと、ユーザーへのリソースグループの割り当ての両方を含む、順序が付けられたリソースグループを作成することで行われます。
2. 次の例のように、作成するアカウントのユーザービューに特別な属性を設定します。それぞれの deferred 属性は、ソースアカウントを指定するビュー属性と、ソース属性を指定するビュー属性の 2 つを必要とします。これは、次の形式のパスを使用して設定されます。

   ```
   accounts[<resource>].deferredAttributes.<attname>.resource

   accounts[<resource>].deferredAttributes.<attname>.attribute
   ```

この <resource> は実際のリソース名に置き換えられ、<attname> は実際の属性名に置き換えられます。

たとえば、1) アカウントの作成時に uid 属性を生成する LDAP というリソースと、2) LDAP リソースの uid と同じ値を保持する directoryid という directoryid 属性を持つ HR というリソースの、2 つのリソースを作成する場合を考えてみましょう。

次のフォームフィールドは、必要なビュー属性を設定し、この関係を定義します。

```
<Field name='accounts[HR].deferredAttributes.directoryid.resource'>
   <Expansion><s>LDAP</s></Expansion>
</Field>
<Field name='accounts[HR].deferredAttributes.directoryid
   <Expansion><s>uid</s></Expansion>
</Field>
```

ビューの拡張

パスワードや有効化フラグなどの特定のリソースアカウント属性を設定するビューを使用して、追加のアカウント属性を設定できます。ただし、セキュリティーを確保するために、これらの拡張属性は登録を必要とします。

属性の登録

属性は、次のいずれかの場所に登録できます。

表 3-81 属性の登録場所

場所	属性をその場所に登録する状況
リソース内の AccountAttributeType 定義	... 更新する属性が、そのタイプのすべてのリソースではなく、特定のリソースのみに適用されます。
システム設定オブジェクト	... 特定タイプのすべてのリソースに適用されるグローバル登録を行います。このような登録は、XML フォーマットで行います。

異なる属性を異なるビューに登録できます。たとえば、パスワードビューに lock 属性を登録し、名前変更ビューに firstname 属性を登録することができます。

グローバル登録

すべてのリソースに適用されるグローバル登録を行うときは、次のパスを使用して、属性を System Configuration オブジェクトに追加します。

```
updatableAttributes.ViewName.ResourceTypeName
```

この *ViewName* は Password、Reset、Enable、Disable、Rename、Delete のいずれかで、*ResourceTypeName* はリソースのタイプ名です。`all` というタイプ名は、すべてのリソースに適用される登録のために予約されています。

この属性の値は、List (String のリスト) です。各文字列は、更新する属性の名前です。

次の例は、すべてのリソースを対象とした `delete before action` という属性をプロビジョニング解除ビューに登録します。

```
<Attribute name='updatableAttributes'>
   <Object>
      <Attribute name='Delete'>
         <Object>
            <Attribute name='all'>
               <List>
                  <String>delete before action</String>
               </List>
            </Attribute>
         </Object>
      </Attribute>
      <Attribute name='Enable'>
         <Object>
            <Attribute name='all'>
               <List>
                  <String>enable before action</String>
               </List>
            </Attribute>
         </Object>
      </Attribute>
   </Object>
</Attribute>
```

リソースに固有の登録

リソースに固有の登録を行うには、Identity Manager の「デバッグ」ページでリソースオブジェクトを修正し、`AccountAttributeType` 要素に `<Views>` 下位要素を挿入します。`<Views>` には、この属性が更新されるビューの名前を示す文字列のリストを指定します。

```
<AccountAttributeType name='lastname' mapName='sn'mapType='string'>
   <Views>
      <String>Rename</String>
   </Views>
</AccountAttributeType>
```

ビューでは、変更する属性が、次のオブジェクトの内部に配置されます。

```
resourceAccounts.currentResourceAccounts[ResourceTypeName].attribut
es
```

```
<Field name=
'resourceAccounts.currentResourceAccounts[OS400ResourceName].attributes.dele
te before action' hidden='true'>
   <Expansion>
      <s>os400BeforeDeleteAction</s>
   </Expansion>
</Field>
```

第 4 章

XPRESS 言語

この章では、式とスクリプトを記述するための XML ベースの言語、XPRESS の基本機能について説明します。この言語で記述されたステートメントは式と呼ばれ、フォームにデータ変換機能を追加したり、ワークフローやフォームなどのオブジェクトに状態遷移ロジックを組み込んだりするために、Identity Manager 全体で使用されます。

この章の内容

この章で説明する主な内容は次のとおりです。

- プレフィックス表記や XML 構文の使用などの XPRESS 言語の基本機能
- Identity Manager で使用される一般的な式の例
- Identity Manager に付属する関数のライブラリ
- 関数から返されるデータの型

XPRESS 言語について

XPRESS は、XML に基づいた構文を使用する関数言語です。この言語で記述されるすべてのステートメントは、0 個以上の引数をとる関数を呼び出し、値を返します。Identity Manager には多数の関数が組み込まれているだけでなく、新しい関数を定義することもできます。XPRESS は、式の中で任意の Java クラスでのメソッドの呼び出しと、JavaScript の評価もサポートします。

プレフィックス表記

XPRESS 言語は、関数の呼び出しと、C などの言語が式の演算子として参照する内容を区別しません。このため、プレフィックス表記と呼ばれる構文スタイルが採用されています。プレフィックス表記は、式の演算子を記述したあとにオペランドを続けるという点で一般的なインフィックス表記とは異なります。たとえば、インフィックス表記を使用して C 言語で記述された簡単な論理式を考えてみましょう。

```
x == 42
```

このステートメントを、プレフィックス表記を使用した C 言語で記述すると、次のようになります。

```
== x 42
```

C 言語に式の演算子がなく、関数だけを利用できる場合は、このステートメントは次のように表記されます。

```
equals(x, 42)
```

式を記述するのではなく、関数を呼び出すという観点に立つと、プレフィックス表記は簡単に理解できます。

XML 構文と例

XPRESS は、解析と操作が簡単で、Identity Manager で使用されるその他の XML 表記に自然に組み込むことができる XML 構文を使用します。XML 要素の名前は、呼び出す関数の名前です。入れ子構造に含まれる要素は、関数に渡される引数です。また、各要素には、開始と終了のタグがあります (次の例では、<add></add>)。

例

```
<add> <ref>counter</ref> <i>10</i> </add>
```

この例の <add> 要素は、add という関数の呼び出しを表します。この関数には、2 つの引数が渡されます。

- 最初の引数 - ref という関数の呼び出しによって定義される値。ref 関数の引数は、変数名のリテラル文字列です。ref 関数から返される値は、*counter* という変数の現在値です。
- 2 番目の引数 - i という関数の呼び出しによって定義される値。i 関数の引数は、整数のリテラル文字列です。i 関数から返される値は、10 という整数です。

add 関数から返される値は、*counter* 変数の現在値に整数 10 を加えた結果となります。すべての関数呼び出しは、値を返すか、またはいずれかの引数に対する処理を実行します。たとえば、ref の呼び出しによってカウンタの値が得られると、<i> 呼び出しは整数 10 を返し、<add> 呼び出しは 2 つの呼び出しの結果を合計します。

次の例は、古典的な Hello World プログラムです。これは、XPRESS では次のように表記されます。

```
<print><s>Hello World!</s></print>
```

Identity Manager との統合

XPRESS はスタンドアロンインタプリタでも使用できますが、通常は、動作の制御とカスタマイズに XPRESS ステートメントを使用するアプリケーションに組み込まれます。このようなアプリケーションをホストアプリケーションと呼びます。Identity Manager システムで利用される 2 つの重要なホストアプリケーションは、ワークフローとフォームです。

ホストアプリケーションは XPRESS インタプリタへの呼び出しを作成し、このインタプリタにサービスを提供します。ホストアプリケーションが提供する重要なサービスの 1 つに、外部変数参照の解決があります。多くの式は、式に定義されていない変数を参照します。これらの変数の値は、ホストアプリケーションによって提供されます。ワークフローホストアプリケーションでは、式は、ワークフロープロセスに定義されているあらゆる変数を参照できます。フォームホストアプリケーションでは、式は、フォームフィールドのあらゆる値、または式の評価前に設定される defvar の値を参照できます。

式を使用する理由

式は、主に次のタスクに使用されます。

- **ユーザーインタフェースと管理者インタフェースのフォームのカスタマイズ**: フォームは、フィールドの可視性の制御と、表示できる形式へのデータの変換に XPRESS を使用します。
- **ワークフローでの制御の流れの定義**: ワークフローは、ワークフロープロセスの実行手順の順序を決定する遷移条件の定義に XPRESS を使用します。
- **ワークフローアクションの実装**: ワークフローアクションは、XPRESS を使用して実装できます。アクションの式は、単純な計算を実行するだけでなく、Java クラスまたは JavaScript を呼び出して、複雑な処理を実行することもできます。

ワークフロースクリプトとフォームの編集での式の使用については、15 ページの第 1 章「ワークフロー」を参照してください。

式の操作

ここでは、Identity Manager での一般的な式の使用、特に次の内容について説明します。

- フィールドの可視性の制御
- フィールドのデフォルト値の計算
- フィールド値の取得
- フィールド値の生成
- ワークフローの遷移条件
- ワークフローアクション
- ワークフローアクションからの Java メソッドの呼び出し

フィールドの可視性の制御

フォームの設計でよく問題となるのは、特定の条件と一致するまで、一部のフィールドの表示を抑制する必要がある場合です。たとえば、特定のリソースに固有のフィールドは、そのリソースがユーザーに割り当てられた場合にのみ意味を持ちます。このようなフィールドは、そのようなリソースが割り当てられた場合にのみ表示されるようにします。それ以外の場合は、これらのフィールドが表示されないようにし、評価の対象からも外します。次の例は、このようなフィールドの可視性を制御するために、<Disable> 要素内の式を使用するフィールド定義を示しています。

```
<Field name='HomeDirectory'>
   <Display class='Text'/>
      <Property name='title' value='HomeDirectory'/>
   </Display>

   <Disable>
      <not>
         <contains>
            <ref>accountInfo.typeNames</ref>
            <s>Solaris</s>
         </contains>
      </not>
   </Disable>
</Field>
```

<Disable> 要素は、フォーム XML 言語の一部です。<Disable> 要素には、XPRESS 言語で記述された任意の式を含めることができます。この例の式は、accountInfo.typeNames という外部変数に格納されているリストに、Solaris という文字列が含まれているかどうかを確認します。フォームでは、この変数には、ユーザーに現在割り当てられているすべてのリソースタイプのリストが格納されます。

画面表示のためにフォームが処理されると、<Disable> 要素に定義されている式が評価されます。true が返される場合は、このフィールドは表示されません。

NULL および 0 という値は、論理 false を表します。NULL 以外またはゼロ以外の値が設定されたフィールドは、論理 true を表します。つまり、<s>false</s> という式で表される文字列は、NULL 値ではないので、論理 true となります。

フィールド宣言に指定された Derivation、Default、Expansion のいずれかの要素を使った XPRESS によりフィールド値を算出することができます。

フィールドのデフォルト値の計算

フィールドの値は、別のフィールドから計算することも、<Default> 要素を使用して初期値として単純に設定することもできます。<Default> 要素は、編集可能なフィールドの初期化でよく使用され、フィールドにまだ値が割り当てられていない場合にのみ評価されます。また、<Default> 要素は、ユーザーの姓と名に基づいてアカウント ID を算出する場合にもよく使用されます。次の例は、文字列操作式を使用して、ユーザーの名の頭文字と姓を組み合わせたデフォルトのアカウント ID を算出するフィールド定義を示しています。

```
<Field name='waveset.accountId'>
   <Display class='Text'/>
      <Property name='title' value='AccountID'/>
   </Display>
   <Default>
      <concat>
         <substr>
            <ref>accounts[AD].firstname</ref>
            <i>0</i>
            <i>1</i>
         </substr>
         <ref>accounts[AD].lastname</ref>
      </concat>
   </Default>
</Field>
```

<Default> 要素は、フォーム XML 言語の一部です。この要素には、XPRESS 式か、または XML オブジェクトという別の言語で表現された要素を含めることができます。XML オブジェクト言語の詳細については、「XML オブジェクト言語」の章を参照してください。

このフィールドを処理すると、システムはまず、waveset.accountId 属性にすでに値が設定されているかどうかを確認します。値が存在しない場合は、<Default> 要素内の式を評価します。この例では、名の頭文字と姓を組み合わせることで値が計算されます。

次の例のように、firstname フィールドと lastname フィールドに値が設定されていることを確認しなければならない場合もあります。

```
<cond>
  <and>
    <notnull><ref>accounts[AD].firstname</ref></notnull>
    <notnull><ref>accounts[AD].lastname</ref></notnull>
  </and>
  <concat>
    <substr>
         <ref>accounts[AD].firstname</ref>
         <i>0</i>
         <i>1</i>
     </substr>
    <ref>accounts[AD].lastname</ref>
  </concat>
</cond>
```

このコード例は、その他のプログラミング言語の IF-THEN ステートメントのように構築されています。この cond 式には、次の 2 つの引数があります。

- 条件式
- THEN 式

まず、条件式が評価されます。この式の結果が論理 true であれば、cond の値が THEN 式の値となります。条件式の結果が false となる場合は、cond の値は NULL になります。

この例の cond ステートメントは、2 つのアカウント属性に値が設定されていることを確認した上で、それぞれの値を使用して accountID を計算します。Default 式は、前提条件が最終的に満たされるまで、またはユーザーがフィールドに値を指定するまで、フォームの表示を更新またはフォームを保存するたびに評価されます。関連するフィールドに NULL 以外の値が設定されている場合は、Default 式は評価されません。

フィールド値の導出

<Derivation> 式は <Default> 式に似ていますが、フィールドに NULL 以外の値が設定されている場合でもフィールドの値を常に計算するという点で異なっています。一般的には、別のフィールドの値によって値が置き換えられるフィールドの表示に使用されます。これは、リソースの属性値がエンコードされ、ユーザーに明示されない場合に有効な設計機能です。

次の例は、条件ロジックを使用して、値のセットを別のセットにマップするフィールド定義を示しています。

```
<Field name='location' prompt='Location'>
   <Display class='Text'/>
      <Derivation>
         <switch>
            <ref>accounts[Oracle].locCode</ref>
            <case>
               <s>AUS</s>
               <s>Austin</s>
            </case>
            <case>
               <s>HOU</s>
               <s>Houston</s>
            </case>
            <case>
               <s>DAL</s>
               <s>Dallas</s>
            </case>
            <case default='true'>
               <s>unknown</s>
            </case>
         </switch>
   </Derivation>
</Field>
```

<Derivation> 要素は、式を含めることができるフォーム XML 言語の一部です。このフィールドを処理すると、<Derivation> 要素内の式が評価され、このフィールドに表示される値が決定されます。

この例では、リソースアカウント属性 accounts[Oracle].locCode の値と、各 case 式の最初の値が比較されます。一致が見つかると、switch 式の結果は、一致した case 式の 2 番目の値となります。一致が見つからない場合、switch 式の結果は、デフォルトの case 式の値となります。

フィールド値の生成

一部のフォームでは、最初は抽象的に生成されたフィールドのセットをユーザーに表示し、その後のフォームの送信時に、また別の、具体的なリソースアカウントの属性値のセットを生成することが必要となる場合があります。これは、フォームの展開と呼ばれます。<Expansion> 要素は、通常は、フォーム内の編集可能フィールドに依存する非表示フィールドで使用されます。<Expansion> 要素の目的の 1 つは、エンドユーザーが慣れ親しんだ判読可能な形式のデータを、リソースが認識できる形式のデータに変換することです。たとえば、ユーザーはフォームでマネージャーのフルネームを確認できますが、システムが受け取るのは、そのマネージャーを認識するための一意の ID です。

次の例は、条件ロジックを使用して、前述の例の location フィールドから得られる値を、3 文字の略号に変換するフィールド定義を示しています。

```
<Field name='accounts[Oracle].locCode'>
   <Expansion>
      <switch>
         <ref>location</ref>
         <case>
            <s>Austin</s>
            <s>AUS</s>
         </case>
         <case>
            <s>Houston</s>
            <s>HOU</s>
         </case>
         <case>
            <s>Dallas</s>
            <s>DAL</s>
         </case>
      </switch>
   </Expansion>
</Field>
```

<Expansion> 要素は、式を含めることができるフォーム XML 言語の一部です。このフィールドを処理すると、<Expansion> 要素内の式が評価され、このフィールドの値が決定されます。

この例では、location フィールドによるマッピングとは逆のマッピングが行われます。また、Display クラスが割り当てられていないため、このフィールドは非表示になります。Display クラスを取り除くことで、フィールドはフォームに表示されなくなります。ただし、フィールドは依然としてフォームのアクティブな部分と見なされ、<Expansion> 式を利用してリソース属性の値を生成します。

注 ユーザービュー以外のすべてのフォームでは、Expansion 規則は、ページを再計算するとき、またはフォームを保存するときに実行されます。ユーザービューでは、ユーザーフォームを最初に読み込むときにも <Expansion> タグが実行されます。

ワークフローの遷移条件

ワークフロープロセスを定義するときは、あるワークフローアクティビティーから別のワークフローアクティビティーにコントロールを移行するための規則を指定します。2 つのアクティビティー間の移行を遷移と呼びます。遷移の使用を制御する規則は、遷移条件と呼ばれます。

たとえば、次のようなアクティビティー定義を考えてみましょう。

```
<Activity name='Check Results'>


   <Transition to='Log Errors'>
      <gt> <ref>ERROR_COUNT</ref> <i>0</i> </gt>
   </Transition>


   <Transition to='end'/>


</Activity>
```

このアクティビティーは Log Errors と end という別々のアクティビティーへの 2 つの異なる遷移を定義しています。ワークフローがこのアクティビティーを処理するときは、条件が true を返す最初の遷移が採用されます。

この例の最初の遷移には、ERROR_COUNT 変数の値がゼロより大きいかどうかを調べる条件が設定されています。この遷移は、エラーカウントが正の値である場合にのみ採用されます。2 番目の遷移には、条件は設定されていません。このため、最初の遷移条件の結果が false となる場合は、常にこの遷移が採用されます。

ワークフローアクション

ワークフローアクティビティーは、1 つまたは複数のアクションを実行できます。アクションの 1 つに XPRESS 式の評価があります。次に例を示します。

```
<Activity name='Increment Counter'>
   <Action>
      <expression>
         <set name='counter'>
            <add> <ref>counter</ref> <i>1</i> </add>
         </set>
      </expression>
   </Action>


   <Transition to='Next'/>


</Activity>
```

XPRESS によってワークフローアクションが実装されると、XPRESS 式は `expression` 要素に囲まれ、`Action` 要素内に配置されます。この例では、式は *counter* という変数の現在値を参照して、その値に 1 を追加し、加算後の値を同名の変数に割り当てます。

ワークフローアクションからの Java メソッドの呼び出し

複雑なワークフローアクションは、Java によって実装できます。複雑なワークフローアクションの代表的な例には、リレーショナルデータベースへのデータの格納や、ヘルプデスクシステムへのメッセージの送信などがあります。これらの Java クラスは、XPRESS を使用してワークフローに統合できます。

```
<Activity name='Log Status'>


   <Action>
      <expression>
         <invoke name='logStatus'
            class='custom.OracleStatusLog'>
            <ref>accountId</ref>
            <ref>email</ref>
            <ref>status</ref>
         </invoke>
      </expression>
   </Action>
   <Transition to='Next'/>


</Activity>
```

この例では、カスタム Java クラス `custom.OracleStatusLog` に定義されている `logStatus` という静的なメソッドが、XPRESS の `invoke` 関数を使用して呼び出されます。このメソッドに渡される 3 つの引数は、ワークフロー変数から取得された値です。

このような場合、主な計算は Java クラスで実行され、XPRESS によってそのクラスがワークフローに統合されます。

式の検証

式の検証は、次の 2 段階で行われます。

1. lh コマンドによる XML 構文の検証
2. XPRESS 評価のトレース

lh コマンドによる式構文の検証

ロジックを実際に評価せずに、式の XML 構文を検証する方法は次のとおりです。

1. `PATH` 環境変数に `%WSHOME%\bin` が含まれていることを確認します。Identity Manager で使用する環境変数の変更については、『Identity Manager インストール』のコマンド行ツールの使用に関する説明を参照してください。

パスに %WSHOME%\bin が設定されていないときは、%WSHOME%\bin を設定するまでツールを実行できません。

2. コマンド行に **lh xmlparse <xpress_file>** と入力します。この **xpress_file** は、検証する XML が記述されたファイルの名前です。このコマンドは XML の正確さを解析し、コンソールにエラーメッセージを表示します。

注 PATH 環境変数に %WSHOME%\bin を含めてください。こうすることで、どのディレクトリも作業ディレクトリとして使用できるようになります。また、Identity Manager の lh コマンドを、どの作業ディレクトリからも実行できるようになります。

XPRESS 評価のトレース

式を正しく記述し、それをリポジトリに格納すると、XPRESS トレースを有効にして、式が正しく機能するかどうかを確認することができます。XPRESS トレースメッセージは、標準出力デバイスに送信されます。多くの場合、XPRESS はアプリケーションサーバー内で評価されるため、トレースメッセージはコンソールウィンドウ、またはアプリケーションサーバーの起動時にアクティブであったログファイルに出力されます。

XPRESS トレースには、次の 2 つの形式があります。

- **グローバルトレース**。グローバルトレースが有効な場合は、すべての XPRESS 式が追跡の対象となります。
- **ブロックレベルトレース**。ブロックレベルのトレースが有効な場合は、指定したブロック内の式のみがトレースの対象となります。ブロックレベルトレースを設定できるのは、フォームのフィールド要素、またはワークフローに含まれる式に限定されます。

トレース出力の量を減らして分析しやすくできるため、通常はブロックレベルトレースが利用されます。

トレースの有効化

グローバルトレースを有効にするには、Waveset.properties ファイルの xpress.trace というエントリの値を true に設定します。アプリケーションサーバーの稼動中、Waveset.properties ファイルに変更を加えたときは、アプリケーションサーバーを再起動するか、「デバッグ」ページの「**Reload Properties**」をクリックしてください。

ブロックレベルトレースを実行するときは、トレースする式を <block> 要素で囲みます。ブロックの開始タグには、trace='true' という属性を設定してください。

```
<block trace='true'>
   <invoke name='getTime' class='java.util.Date'/>
</block>
```

```
or


<Default>
   <block trace='true'>
      <ref>global.accountId</ref>
   </block>
</Default>
```

無効な設定の例

<block> 要素を、次のように使用しないでください。

```
<block trace='true'>
   <Field name ='field1'>
...
   </Field>
</block>


or


<Field name='Field2'>
   <block trace='true'>
      <Default>
         <ref>global.accounts</ref>
      </Default>
   </block>
</Field>
```

トレースメッセージには、関数名、各引数の値、戻り値が含まれます。

XPRESS のトレースを無効にするときは、xpress.trace の値を false に変更し、Waveset.properties ファイルを読み込みなおします。

関数

Identity Manager には、式で使用できる XPRESS 関数のライブラリが含まれます。これらの関数は、次のように分類されます。

- 値コンストラクタ式
- 演算式
- 論理式
- 文字列操作式
- リスト操作式
- 条件、繰り返し、およびブロックの式
- 変数と関数の定義式
- オブジェクト操作式
- Java および JavaScript 式
- デバッグと検証の式

値コンストラクタ式

XPRESS では、リテラル値は XML 要素内のテキストとして記述されます。要素名は関数の名前で、リテラルテキストはその関数の引数として扱われます。次に、簡単な原子データ型を構成する関数の例を示します。

array 関数

各引数式を評価し、その戻り値を連結することで、*list* 型の値を構成します。この関数は、複数の引数をとることができます。NULL 値はフィルタリングされません。

例

```
<array>
   <s>apples</s>
   <s>oranges</s>
   <s>wiper blades</s>
</array>
```

i 関数

整数値を構成します。この関数は 1 つの引数をとり、その引数がリテラルテキストとして扱われます。テキストに含めることができるのは数値のみで、オプションとして先頭にプラスまたはマイナス記号を追加できます。

例 1

```
<i>0</i>
```

例 2

```
<i>42</i>
```

例 3

```
<i>-1234</i>
```

list 関数

各引数式を評価し、その戻り値を連結することで、*list* 型の値を構成します。この関数は、複数の引数をとることができます。NULL 値はフィルタリングされません。

例

```
<list>
   <s>apples</s>
   <s>oranges</s>
   <s>wiper blades</s>
</list>
```

map 関数

キーと値の部分式のペアから構成されるマップを作成します。

例

```
<map>
   <!-- キー 1-->
   <!-- 値 1-->
   <!-- キー n-->
   <!-- 値 n-->
</map>
```

null 関数

NULL 値を構成します。

例 1

```
<null/>
```

例 2

```
<null></null>
```

s 関数

文字列値を構成します。この関数は 1 つの引数をとり、その引数がリテラルテキストとして扱われます。文字列の長さは、Java 環境で利用できる隣接メモリーの容量のみによって制限されます。

例

```
<s>Now is the time</s>
```

演算式

式で演算処理を行うには、次の関数を使用します。

add 関数

すべての引数の値に整数を加算します。整数以外の引数は、強制的に整数として扱われます。

例

次の式の結果は、整数 (42) となります。

```
<add> <i>40</i> <i>1</i> <s>1</s> </add>
```

div 関数

すべての引数の値を整数値で割ります。整数以外の引数は、強制的に整数として扱われます。

例

次の式の結果は、整数 (42) となります。

```
<div> <i>84</i> <i>2</i> </div>
```

mod 関数

すべての引数の値を、整数値でモジュロ演算します。引数の値は、強制的に整数として扱われます。NULL が指定された引数は無視されます。

例

次の式の結果は、整数 (42) となります。

```
<mod> <i>142</i> <i>100</i> </mod>
```

mult 関数

すべての引数の値に整数値を掛けます。整数以外の引数は、強制的に整数として扱われます。

例

次の式の結果は、整数 (42) となります。

```
<mult> <i>7</i> <i>3</i> <i>2</i> </mult>
```

sub 関数

すべての引数の値を整数値で減算します。整数以外の引数は、強制的に整数として扱われます。

例

次の式の結果は、整数 (42) となります。

```
<sub> <i>50</i> <i>6</i> <i>2</i> </sub>
```

論理式

式で論理操作を行うには、次の関数を使用します。ほとんどの論理関数は、true または false を表す 1 および 0 を返します。例外は、cmp、ncmp、and、および or です。

and 関数

すべての引数の値を取得し、いずれかの引数値が論理 false である場合にゼロを返します。1 つの子が false と評価された場合は、後続の子は評価されません。すべての引数が論理 true となる場合は、最後の引数の値が返されます。ゼロ (0 または <s>0</s>) と <null> は論理 false と見なされます。

例 1

次の式はゼロを返します。

```
<and> <i>42</i> <s>cat</s> <i>null</i> </and>
```

例 2

次の式は cat を返します。

```
<and> <i>42</i> <s>cat</s> </and>
```

cmp 関数

2 つの文字列値を比較します。この関数を使用して、文字列のリストをソートできます。

この関数から返される値は次のとおりです。

- 負の値 - 最初の引数の値が、語彙的に 2 番目の引数の値より小さい
- 正の値 - 最初の引数の値が、語彙的に 2 番目の引数の値より大きい
- 0 (ゼロ) - 2 つの引数の値が等しい

引数の値は、必要に応じて、強制的に文字列として扱われます。

例 1

次の式は -1 を返します。

```
<cmp>
   <i>20</i>
   <i>100</i>
</cmp>
```

例 2

次の式は -16 を返します。この式は、r と b の間にアルファベット順序の違いがあることを示します。b と r の間に 16 文字分の開きがあるため、ray に対して bob を比較した場合は、-16 という値が返されます。反対に、b に対して r を比較した場合は、返される値は 16 となります。

```
<cmp>
   <s>bob</s>
   <s>ray</s>
</cmp>
```

例 3

次の式は 0 (ゼロ) を返します。

```
<cmp>
   <s>daryl</s>
   <s>daryl</s>
</cmp>
```

eq 関数

等価性を検証します。この関数は、通常は 2 つの値をとりますが、複数の値をとることもできます。等価性の検証方法は、最初の引数のデータ型によって決定されます。最初の引数のデータ型に応じて、検証方法が次のように切り替わります。

- string - 以後のすべての引数は強制的に文字列として扱われ、文字列の比較が行われます
- integer - 以後のすべての引数は強制的に整数として扱われ、数値の比較が行われます
- object - 以後のすべての引数は、object 型でなければいけません。Object.equals の値はすべての引数について true です

この関数から返される値は次のとおりです。

0 - ステートメントが論理 false となる

1 - ステートメントが論理 true となる

例

```
<eq> <ref>role</ref> <s>engineering</s> </eq>
```

gt 関数

2 つの引数をとります。

この関数から返される値は次のとおりです。

- `0` - 最初の引数の値が、数値的に 2 番目の引数の値より小さいか、または等しい
- `1` - 最初の引数の値が、数値的に 2 番目の引数の値より大きい

例

```
<gt>
   <ref>age</ref>
   <i>42</i>
</gt>
```

gte 関数

2 つの引数をとります。

この関数から返される値は次のとおりです。

- `0` - 最初の引数の値が、2 番目の引数の値より小さい
- `1` - 最初の引数の値が、数値的に 2 番目の引数の値より大きいか、または等しい

例

次の式は 1 を返します。

```
<gte>
   <i>10</i>
   <i>5</i>
</gte>
```

isFalse 関数

0 と 1 の数値ではなく、`true` と `false` の文字列で表現されるブール型の値を参照するときに使用されます。1 つの引数をとります。

この関数から返される値は次のとおりです。

- `0` - 引数の値が論理 true となる。以下の場合に true と見なされる : 文字列 `true`、ブール型の true、およびゼロ以外の整数。これら以外はすべて false と見なされる。
- `1` - 引数の値が論理 false であるか、または `false` という文字列を返す。

例

次の式は 1 を返します。

```
<isFalse>
   <s>false</s>
</isFalse>
```

isnull 関数

1 つの引数をとります。

この関数から返される値は次のとおりです。

- `0` - ステートメントが NULL 以外となる
- `1` - ステートメントが NULL になる

例 1

次の式は `1` を返します。

```
<isnull> <null/> </isnull>
```

例 2

次の式は `0` を返します。

```
<isnull> <i>0</i> </isnull>
```

isTrue 関数

0 と 1 の数値ではなく、`true` と `false` の文字列で表現されるブール型の値を参照するときに使用されます。1 つの引数をとります。

この関数から返される値は次のとおりです。

- `0` - 引数の値が論理 false となる。以下の場合に true と見なされる : 文字列 `true`、ブール型の true、およびゼロ以外の整数。これら以外はすべて false と見なされる。
- `1` - 引数の値が論理 true となる。

例

次の式は 0 を返します。

```
<isTrue>
   <s>false</s>
</isTrue>
```

lt 関数

2 つの引数をとります。

この関数から返される値は次のとおりです。

- 0 - 最初の引数の値が、数値的に 2 番目の引数の値より大きいか、または等しい
- 1 - 最初の引数の値が、数値的に 2 番目の引数の値より小さい

例 1

次の式は 0 (ゼロ) を返します。

```
<lt>
   <i>10</i>
   <i>5</i>
</lt>
```

例 2

次の式は 1 を返します。

```
<lt>
   <i>5</i>
   <i>10</i>
</lt>
```

lte 関数

2 つの引数をとります。

この関数から返される値は次のとおりです。

- 0 - 最初の引数の値が、数値的に 2 番目の引数の値より大きい
- 1 - 最初の引数の値が、数値的に 2 番目の引数の値より小さいか、または等しい

例

```
<lte>
   <ref>age</ref>
   <i>42</i>
</lte>
```

ncmp 関数

大文字と小文字を区別せずに、2 つの文字列値を比較します。

この関数から返される値は次のとおりです。

- 負の値 - 最初の引数の値が、語彙的に 2 番目の引数の値より小さい
- 正の値 - 最初の引数の値が、語彙的に 2 番目の引数の値より大きい
- `0` (ゼロ) - 2 つの引数の値が等しい

引数の値は、必要に応じて、強制的に文字列として扱われます。

例

次の式は 0 を返します。

```
<ncmp>
   <s>Daryl</s>
   <s>daryl</s>
</ncmp>
```

neq 関数

非等価性を検証します。この関数は、`eq` 関数で行われる等価性検証の否定形のように機能します。

この関数から返される値は次のとおりです。

- `0` - 2 つの引数の値が等しい
- `1` - 2 つの引数の値が等しくない

例

```
<neq>
   <ref>role</ref>
   <s>management</s>
</neq>
```

not 関数

入れ子構造の式のロジックを反転させます。

この関数から返される値は次のとおりです。

- `1` - 引数の値が論理 false となる
- `0` - 引数の値が論理 true となる

例

次の例は `1` を返します。

```
<not> <eq> <i>42</i> <i>24</i> </eq> </not>
```

or 関数

複数の引数をとります。

この関数から返される値は次のとおりです。

0 - すべての引数の値が論理 false となる

式の中で最初に論理 true と評価される引数の値

例 1

次の式は論理 false となり、0 を返します。

```
<or> <i>0</i> <i>0</i> </or>
```

例 2

次の式は、論理 true となる cat という文字列を返します。

```
<or> <i>0</i> <s>cat</s> </or>
```

notnull 関数

1 つの引数をとります。

この関数から返される値は次のとおりです。

0 - 引数の値が NULL である

1 - 引数の値が NULL 以外である

例 1

次の式は、firstname に値が設定されている場合は 1 を返し、firstname が NULL の場合は 0 (ゼロ) を返します。

```
<notnull>
   <ref>firstname</ref>
</notnull>
```

例 2

次の式は、値が NULL であるため、0 を返します。

```
<notnull><null/></notnull>
```

文字列操作式

式で文字列操作を行うには、次の関数を使用します。

indexOf 関数

別の文字列に含まれる、指定文字列の位置を返します。

例

次の関数は 3 を返します。

```
<indexOf>
   <s>abcabc</s>
   <s>abc</s>

   <s>l</s>
</indexOf>
```

concat 関数

複数の文字列値を連結します。

例

次の式は、`<s>Now is the time</s>` という文字列を返します。

```
<concat>
   <s>Now </s><s>is </s><s>the </s><s>time</s>
</concat>
```

downcase 関数

強制的に文字列として扱われる、1 つの引数をとります。この関数は、すべての大文字を小文字に変換した状態で、引数の値を返します。

例

次の式は `<s>abc</s>` を返します。

```
<downcase><s>ABC</s></downcase>
```

length 関数

リスト内の要素の数を返します。この関数を使用して、文字列の長さを取得することもできます。

最初の引数 - リストまたは文字列

例 1

次の式は 2 を返します。

```
<length>
   <list>
      <s>apples</s>
      <s>oranges</s>
   </list>
</length>
```

例 2

```
<length>
   <s>Hello world!</s>
</length>
```

この式は、11 という値を返します。

ltrim 関数

強制的に文字列として扱われる、1 つの引数をとります。

この関数は、先頭の空白文字を削除した状態で、引数の値を返します。

例

次の式は <s>hello</s> を返します。

```
<ltrim><s>   hello</s></ltrim>
```

match 関数

非推奨です。代わりに、contains 関数を使用してください。

message 関数

メッセージカタログキーによってメッセージをフォーマットします。

例

```
<message severity-'ok' name='DEFAULT_MESSAGE'>
   <!-- メッセージパラメータ 0-->
   <!-- メッセージパラメータ n-->
</message>
```

pad 関数

文字列が指定の長さになるように、文字列に空白文字を挿入します。

最初の引数 - pad 文字が挿入される文字列

2 番目の引数 - 指定する文字列の長さ

3 番目の引数 - (省略可能) pad 文字 (デフォルトは空白文字)

例

次の式は <s> email　　</s> を返します。

```
<pad>
   <s> email</s>
   <i>10</i>
</pad>
```

rtrim 関数

強制的に文字列として扱われる、1 つの引数をとります。この関数は、最後に続く空白文字を削除した状態で、引数の値を返します。

例

次の例は 0 (ゼロ) を返します。

```
<cmp>
   <s>hello</s><rtrim><s>hello   </s></rtrim>
</cmp>
```

split 関数

文字列を文字列のリストに分割します。

最初の引数 - 分割する文字列

2 番目の引数 - 1 つまたは複数の区切り文字のセット。ここに指定された文字が文字列に含まれると、そのたびに文字列が分割されます。

区切り文字の間の各下位文字列から構成されるリストが作成されます。

例 1

```
<split>
   <s>Austin City Limits</s>
   <s> </s>
</split>
```

この式は、次のリストを返します。

```
<list>
   <s>Austin</s>
   <s>City</s>
   <s>Limits</s>
</list>
```

例 2

次の式では、複数の区切り文字が使用されます。

```
<split>
   <s>(512)338-1818</s>
   <s>()-</s>
</split>
```

この式は、次のリストを返します。

```
<list>
   <s>512</s>
   <s>338</s>
   <s>1818</s>
</list>
```

substr 関数

文字列から文字の範囲を抽出します。

この関数には、次の 2 つの形式があります。

- *start* と *length* が引数として指定される (`substr` 要素の子ノード)。
- *start* と *length* が substr ノードの属性として指定され、*s* が *start*、`l` が *length* を表す。

たとえば、次の 2 つは同じ呼び出しを表します。

```
<substr>
   <s>Hello World</s>
     <i>3</i>
     <i>4</i>
<substr>
```

および

```
<substr s='3' l='4'>
   <s>Hello World</s>
<substr>
```

どちらの関数が返す文字列も `lo W` です。

```
<block>
   <substr s='3' l='4'>
      <s>Hello World</s> --> Hello World
   </substr> --> lo W
</block> --> lo W
```

start パラメータと *length* パラメータの指定は省略可能です。次のように文字列が substr ノードの子として文字列のみが指定され、*start* 引数を省略した場合を考えてみましょう。

```
<substr>
   <s>Hello World</s>
<substr>
```

この場合、substr ノードの *s* 属性も省略され、開始位置 (start) は文字列の先頭と見なされます。つまり、明示しなかった場合は、デフォルト値のゼロが適用されます。

最初の引数 - 文字列

2 番目の引数 - 開始位置

3 番目の引数 - 抽出する文字の数

例

次の式は `<s>Now</s>` を返します。

```
<substr>
   <s>Now is the time</s>
   <i>0</i>
   <i>3</i>
</substr>
```

次の例では、*start* 属性が省略されているため、0 と見なされます。

```
<block>
   <substr l='4'>
       <s>Hello World</s> --> Hello World
   </substr> --> Hell
</block> --> Hell
```

length 引数も省略できます。*length* 引数を省略した場合は、この関数は残りの文字列を返します。次のように、substr の子ノードとして *string* 引数と *start* 引数のみを指定する場合は、*length* を省略できます。

```
<substr>
   <s>Hello World</s>
       <i>3</i>
<substr>
```

または、substr ノードで *l* 属性が省略されている場合にも、省略できます。次の例には *length* 引数が指定されていませんが、指定されている開始位置より後の残りの文字列が返されます。

```
<block>
   <substr s='3'>
      <s>Hello World</s> --> Hello World
   </substr> --> lo World
</block> --> lo World
```

trim 関数

強制的に文字列として扱われる、1 つの引数をとります。

この関数は、先頭の空白文字と最後に続く空白文字を削除した状態で、引数の値を返します。

例

次の式は `<s>hello</s>` を返します。

```
<trim><s> hello </s></trim>
```

upcase 関数

強制的に文字列として扱われる、1 つの引数をとります。

この関数は、すべての小文字を大文字に変換した状態で、引数の値を返します。

例

次の式は `<s>ABC</s>` を返します。

```
<upcase><s>abc</s></upcase>
```

ztrim 関数

先頭のゼロを削除した状態で、部分式の文字列値を返します。

例

```
<ztrim>
   <s>00000sample</s>
</ztrim>
```

この関数は `<s>sample</s>` と評価されます。

リスト操作式

ほとんどのリスト操作関数には、関数要素に name 属性が含まれているかどうかに応じて 2 つの形式があります。

- 関数要素に含まれている場合は、その名前はリストを格納した変数に解決されます。この場合、参照された変数は完全に変更されます。次の例は、someList 変数に格納されているリストを変更して、2 つの要素を追加します。

```
<append name='someList'>
   <s>Hello</s>
   <s>World</s>
</append>
```

関数要素に名前が含まれない場合は、新しいリストが生成されます。次の例では、someList 変数に格納されているリストの要素と 2 つの追加要素を組み合わせて、新しいリストを作成します。someList 変数の値自体は変更されません。

```
<append>
   <ref>someList</ref>
   <s>Hello</s>
   <s>World</s>
</append>
```

リスト要素を操作するときは、次の関数を使用します。

append 関数

リストに値を追加します。*name* 属性が指定されているかどうかに応じて、引数のリストには 2 つの形式があります。*name* が指定されていない場合は、最初の引数がリスト名と見なされ、残りの引数はリストに追加する要素と見なされます。返されるのは、要素が追加されたリストのコピーであり、元のリストは変更されません。*name* 引数が指定されている場合は、すべての引数が追加オブジェクトと見なされ、その名前の変数に格納されているリストに追加されます。この場合は、リストが直接変更されます。

例 1

次の式は、*srclist* 変数に格納されているリストをコピーし、1 つの要素を追加します。

```
<append>
   <ref>srclist</ref>
   <s>oranges</s>
</append>
```

例 2

次の式は、値を追加して既存のリストを変更します。

```
<set name= 'somelist'>
  <List>
    <s>We</s>
    <s>say</s>
  </List>
</set>

<append name= 'somelist'>
  <s>Hello</s>
  <s>World</s>
</append>
<ref>someList</ref>
```

appendAll 関数

複数のリストの要素をマージします。name 属性が指定されている場合は、既存のリストが変更されます。指定されていない場合は、新しいリストが作成されます。

例 1

次の式は、*srclist* リストの要素と 3 つの追加要素を組み合わせて、新しいリストを作成します。

```
<appendAll>
   <ref>srclist</ref>
   <list>
      <s>apples</s>
      <s>oranges</s>
      <s>peaches</s>
   </list>
</appendAll>
```

例 2

次の式は、*srclist* 変数に格納されているリストに 3 つの要素を追加します。

```
<appendAll name='srclist'>
   <list>
      <s>apples</s>
      <s>oranges</s>
      <s>peaches</s>
   </list>
</appendAll>
```

contains 関数

最初の引数 - リストまたは文字列

2 番目の引数 - リストから検索する任意のオブジェクトまたは文字列から検索する下位文字列

この関数から返される値は次のとおりです。

1 -- 指定した値がリストに含まれる、または指定した下位文字列が文字列に含まれる

例 1

次の式は 1 を返します。

```
<contains>
   <list>
      <s>apples</s>
      <s>oranges</s>
   </list>
   <s>apples</s>
</contains>
```

例 2

次の式は 1 を返します。

```
<contains>

   <s>foobar</s>

   <s>foo</s>

</contains>
```

containsAll 関数

2 つのリスト引数をとります。

この関数から返される値は次のとおりです。

1 -- 指定したリストに、もう一方のリストのすべての要素が含まれる

0 (ゼロ) -- 指定したリストに、2 番目のリストの一部またはすべての要素が含まれない

例

次の式は 0 を返します。

```
<containsAll>
   <ref>fruitlist</ref>
   <list>
      <s>oranges</s>
      <s>wiper blades</s>
   </list>
</containsAll>
```

containsAny 関数

最初の引数 - 検索される側のリスト

2 番目の引数 - 最初のリストから検索する要素または要素リスト

この関数から返される値は次のとおりです。

1 -- 2 番目のリストの一部またはすべての要素が、最初のリストに含まれる

0 (ゼロ) -- 2 番目のリストのどの要素も、最初のリストに含まれない

例

次の式は 1 を返します。

```
<containsAny>
   <ref>fruitlist</ref>
   <list>
      <s>oranges</s>
      <s>wiper blades</s>
   </list>
</containsAny>
```

filterdup 関数

リストの重複要素をフィルタリングします。リストを指定すると、重複要素が削除された新しいリストが返されます。

例 1

```
<filterdup>
   <list>
      <s>apples</s>
      <s>oranges</s>
      <s>apples</s>
   </list>
</filterdup>
```

この式は、次のリストを返します。

```
<list>
   <s>apples</s>
   <s>oranges</s>
</list>
```

例 2

この関数を使用して、新しいリストを作成する代わりに既存のリストを操作することもできます。

```
<filterdup name = 'namedlist'/>
```

filternull 関数

リストの NULL 要素をフィルタリングします。

この関数に 1 つのリストを指定すると、すべての NULL 要素が削除された 1 つのリストが返されます。

例

```
<filternull>
   <list>
      <s>apples</s>
      <null>
         <s>oranges</s>
      <null/>
   </list>
</filternull>
```

この式は、次のリストを返します。

```
<list>
   <s>apples</s>
   <s>oranges</s>
</list>
```

例 2

この関数を使用して、新しいリストを作成する代わりに既存のリストを操作することもできます。

```
<filternull name = 'namedlist'/>
```

expand 関数

$() 変数参照を展開した状態で、部分式の文字列値を返します。

例

```
<expand><s>$(sample)</s></expand>
```

get 関数

リスト内の n 番目の要素の値を取得します。リストのインデックスは、ゼロ (0) カウントから始まります。引数はリストと整数です。

例

```
<get>
   <list>
      <s>apples</s>
      <s>oranges</s>
   </list>
   <i>1</i>
</get>
```

この式は `<s>oranges</s>` を返します。

indexOf 関数

最初の引数 - 値が検索されるリスト

2 番目の引数 - 検索する値

3 番目の引数 - (省略可能) 開始インデックス

この関数は、指定した値がリストに存在する場合はその要素の位置番号を返し、指定した値がリストに存在しない場合は `-1` を返します。

例 1

次の式は `1` を返します。

```
<indexOf>
   <list>
      <s>apples</s>
      <s>oranges</s>
   </list>
   <s>oranges</s>
</indexOf>
```

例 2

次の式は `3` を返します。

```
<indexOf>
   <list
      <s>apples</s>
      <s>oranges</s>
   </list>
```

```
   <s>oranges</s>
<i>2</i>
</indexOf>
```

insert 関数

リストに値を挿入します。新しい要素を追加するスペースを作るために、挿入インデックスより下の要素は下に移動されます。

最初の引数 - 要素が挿入されるリスト

2 番目の引数 - 新しい要素を挿入するリスト内の位置を示す整数

3 番目の引数 - リストに挿入される値

例 1

```
<insert>
   <list>
      <s>apples</s>
      <s>oranges</s>
   </list>
   <i>1</i>
   <s>wiper blades</s>
</insert>
```

この式は、次のリストを返します。

```
<list>
   <s>apples</s>
   <s>wiper blades</s>
   <s>oranges</s>
</list>
```

この関数は名前付きリストをとることもできます。

```
<insert name='name_of_list'>
<! -- リストを挿入する位置 >
<! -- 挿入する値 >
</insert>
```

例 2

```
<insert name='variable name of list'>
   <!-- 挿入位置 -->
   <--! 挿入する値 -->
</insert>
```

length 関数

リスト内の要素の数を返します。この関数を使用して、文字列の長さを取得することもできます。

最初の引数 - リストまたは文字列

例 1

次の式は 2 を返します。

```
<length>
   <list>
      <s>apples</s>
      <s>oranges</s>
   </list>
</length>
```

例 2

```
<length>
   <s>Hello world!</s>
</length>
```

この式は、11 という値を返します。

remove 関数

リストから 1 つまたは複数の要素を削除します。name 属性が指定されているかどうかに応じて、引数のリストには 2 つの形式があります。name が指定されていない場合は、最初の引数がリスト名と見なされ、残りの引数はリストから削除する要素と見なされます。返されるのは、要素が削除されたリストのコピーです。元のリストは変更されません。name 引数が指定されている場合は、すべての引数が削除オブジェクトと見なされ、その名前の変数に格納されているリストから削除されます。この場合は、リストが直接変更されます。

例 1

次の式は、*srclist* 変数に格納されているリストをコピーして、1 つの要素を削除し、結果リストを返します。

```
<remove>
   <ref>srclist</ref>
   <s>oranges</s>
</remove>
```

例 2

次の式は、値を削除して既存のリストを変更します。

```
<set name= 'somelist'>
  <List>
    <s>We</s>
    <s>say</s>
  </List>
</set>

<remove name= 'somelist'>
  <s>say</s>
  <s>say</s>
</remove>
<ref>someList</ref>
```

removeAll 関数

指定したリストに含まれるすべての要素を、もう一方のリストから削除します。name 属性が指定されている場合は、既存のリストが変更されます。指定されていない場合は、新しいリストが作成されます。

例 1

次の式は、*srclist* リストの要素から 3 つの要素を削除して、新しいリストを作成します。

```
<removeAll>
   <ref>srclist</ref>
   <list>
      <s>apples</s>
      <s>oranges</s>
      <s>peaches</s>
   </list>
</removeAll>
```

例 2

次の式は、*srclist* 変数に格納されているリストから 3 つの要素を削除します。

```
<removeAll name='srclist'>
   <list>
      <s>apples</s>
      <s>oranges</s>
      <s>peaches</s>
   </list>
</removeAll>
```

この式によって、リストは次のように変更されます。

```
<list>
   <s>wiper blades</s>
</list>
```

retainAll 関数

2 つのリストの共通部分を計算し、両方のリストに含まれる要素を返します。

この関数には 2 つの形式があります。

例 1

名前付きリストを、このリストともう一方のリストの共通部分のリストにします。

```
<retainAll name='variable name of list'>

<!-- もう一方のリスト -->

</retainAll>
```

例 2

2 つのリストの共通部分を返します。

```
<retainAll>

   <!-- 最初のリスト >

  <!-- 2 番目のリスト -->

</retainAll>
```

setlist 関数

リスト内の指定位置に値を割り当て、現在の値を上書きします。インデックスが付けられた要素を含めるために必要となる場合は、リストが拡張されます。リストの拡張時に作成される新しい要素は、NULL になります。

最初の引数 - リスト

2 番目の引数 - 新しい要素を挿入するリスト内の位置を示す整数 (要素の位置番号はゼロから開始)

3 番目の引数 - 要素

例 1

```
<setlist>
   <list>
      <s>apples</s>
      <s>oranges</s>
      <s>wiper blades</s>
   </list>
   <i>2</i>
   <s>bassoons</s>
</setlist>
```

この式によって、リストは次のように変更され、NULL が返されます。

```
<list>
   <s>apples</s>
   <s>oranges</s>
   <s>bassoons</s>
</list>
```

例 2

```
<setlist>
   <list>
      <s>apples</s>
      <s>oranges</s>
      <s>wiper blades</s>
   </list>
   <i>5</i>
   <s>bassoons</s>
</setlist>
```

この式によって、リストは次のように変更され、NULL が返されます。

```
<list>
   <s>apples</s>
   <s>oranges</s>
   <s>wiper</>
   </null>
   </null>
   <s>bassoons</s>
</list>
```

条件、繰り返し、およびブロックの式

式で条件処理とブロック処理を行うには、次の関数を使用します。

block 関数

複数の式を 1 つの式にグループ化します。block 関数の値は、最後の引数の値となります。

注 `<set>` 関数は値を返しません。block ステートメントの最後の行で set の操作が行われている場合、その block ステートメントは値を返しません。block ステートメントが変数の値を返すようにするには、block ステートメントの最後の行に `<ref>variable_name</ref>` を追加してください。

例

```
<block>
   <s>Hello there!</s>
   <add> <i>100</i> <i>2</i> </add>
   <i>42</i>
</block>
```

このブロックは、最後の引数の値である `42` を返します。

trace ステートメントでの block の使用例については、「デバッグと検証の式」を参照してください。

break 関数

式の早期終了を強制します。break を使用できる式は、`block`、`dolist`、`while`、`and`、`or` です。`break` 式の値は、それを含む式の値となります。オプションとしてブロック名を指定すると、break を使用して、式のいくつかのレベルを終了できます。

例 1

次の式には、ループの単純な break 終了が含まれます。

```
<dolist name='el'>
<ref>list</ref>
   <cond><eq><ref>el</ref><s>000</s></eq>
      <break>
         <ref>el</ref>
      </break>
   </cond>
<null/>
</dolist>
```

この例では、リストに含まれる要素に対して dolist 関数が繰り返され、000 という値が検索されます。dolist 関数の値は、各繰り返しの最後の部分式から返された値を連結して構成されたリストとなります。

例 2

次の式は、ブロック名を使用して複数のレベルを終了する例を示しています。

```
<block name='outer block'>
  <dolist name='el'>
    <ref>listOfLists</ref>
    <dolist name='el2'>
      <ref>el</ref>
      <cond><eq><ref>el</ref><s>000</s></eq>
        <break name='outer block'>
          <ref>el</ref>
        </break>
      </cond>
    </dolist>
    <null/>
  </dolist>
</block>
```

ループが 2 つ存在する以外は、この式は前の例と同じです。外側のループは、それ自体がリストである要素を持つリストに対して繰り返されます。内側のループは、各要素リストに対して繰り返されます。000 という値が検出されると、両方のループが終了し、break 式の outer block というブロック名が参照されます。

cond 関数

この関数は、2 つの式の値を条件付きで選択する機能を持ちます。これは、C と Java の ternary 条件演算子 (a?b:c) と類似した機能です。

例

cond 関数は 3 つの引数をとることができます。最初の引数は条件です。条件の値が論理 true であれば、*cond* の値は、2 番目の引数の値となります。条件の値が false の場合は、*cond* の値は、3 番目の引数の値となります。条件の値が false で、3 番目の引数が指定されていない場合は、*cond* の値は NULL となります。

```
<cond>
   <gt>
      <ref>age</ref>
      <i>40</i>
   </gt>
   <s>old</s>
   <s>young</s>
</cond>
```

dolist 関数

リストの要素に対して繰り返されます。name 属性の値は、ループ内で参照できる変数の名前となります。

この変数の値は、以後のリスト要素の値となります。

最初の部分式は、ループを適用するリストを返します。残りの部分式は、リスト内の要素ごとに 1 回繰り返されます。

dolist 関数の値は、各繰り返しの最後の部分式から返された値を連結して構成されたリストとなります。

例

次の式は、subset というリストを作成します。このリストには、*srclist* に格納されている要素の中で、値が 10 を超える要素のサブセットが含まれます。

```
<set name='subset'>
   <dolist name='el'>
     <cond>
       <gt>
         <ref>el</ref>
         <i>10</i>
       </gt>
       <ref>el</ref>
     </cond>
   </dolist>
</set>
```

switch 関数

最初の引数 - 任意の XPRESS 式

2 番目の引数 - 一連の <case> 要素

最初の引数が評価されてから、一致が見つかるまで各 **<case>** 要素と比較されます。<switch> 関数は、一致が存在する最初の <case> に評価されます。一致が見つからない場合は、<switch> は default='true' である <case> 要素に評価されます。

例

次の式は apples を返します。

```
<switch>
   <s>A</s>
   <case default='true'>
      <s>unknown</s>
   </case>
   <case>
```

```
      <s>A</s>
      <s>apples</s>
   </case>
   <case>
      <s>B</s>
      <s>oranges</s>
   </case>
</switch>
```

select 関数

リスト内の最初の NULL 以外 (およびゼロ以外) の値を返します。

例

```
<select>
   <ref>:display.session</ref>
   <ref>context</ref>
</select>
```

次のステートメントがあるとします。

```
<select>
   <ref>first/ref>
   <ref>second</ref>
   <ref>third</ref.
</select>
```

このステートメントはまず、`first` が NULL かどうかをチェックします。NULL でない場合は、`first` の値を返します。NULL の場合は次の項目に移り、true を返すかすべての項目をチェックするまで繰り返します。

この関数は、ワークフローなどから正しいコンテキストを取得する必要があるとき、または formUtil メソッドを呼び出すときに使用できます。

このように select を使用することで、Lighthouse コンテキストがどの変数にあるかがわからなくても、Identity Manager 内の任意の場所から formUtil メソッドを呼び出すことができます。フォームでは、`<ref>:display.session</ref>` を使ってコンテキストを指定します。ただしワークフローでは、同じ FormUtil 呼び出しで `<ref>context</ref>` を代わりに使用する必要があります。

while 関数

条件と一致するまで一連の式を繰り返します。最初の部分式は条件で、ループのたびに評価されます。条件が論理 false になると、ループは終了します。`while` 式の値は、最後の繰り返しでの、ループの最後の式の値となります。

例

次の式は NULL を返します。

```
<while>
   <gt>
      <ref>counter</ref>
      <i>0</i>
   </gt>

   <set name='counter'>
      <sub> <ref>counter</ref>
         <i>1</i>
      </sub>
   </set>
</while>
```

変数と関数の定義式

式で変数と関数を参照、定義するときは、次の関数を使用します。

ref 関数

変数の値を参照します。変数には、ホストアプリケーションがサポートする外部変数と、<defvar> を使用して定義される内部変数があります。

例 1

```
<ref>waveset.role</ref>
```

例 2

```
<defvar name='milk'><s>milkvalue</s></defvar>

<defvar name='shake'><s>milk</s></defvar>

<ref><ref>shake</ref>
```

<s>milkshake</s> と評価される

defvar 関数

新しい変数を定義します。定義した変数は、その変数を定義した式とそれ以降の式で自由に参照できます。変数名の指定には、XML 属性 name を使用します。

defvar ステートメント自体がその変数を参照することはできません。このような参照を設定すると、ループが生じます。

注 次のような構造は使用できません。

```
<defvar name='fullname'>
   <ref>fullname</ref>
</defvar>
```

または

```
<defvar name='counter'/>
   <add><ref>counter</ref>
      <i>0</i>
   </add>
</defvar>
```

例 1

次の式は変数を定義し、2 つの要素を持つリストとしてその値を初期化します。

```
<defvar name='theList'>
  <list>
     <s>apples</s>
     <s>oranges</s>
  </list>
</defvar>
```

例 2

次の式は変数を定義し、その値を整数ゼロに初期化します。

```
<defvar name='counter'>
   <i>0</i>
</defvar>
```

defarg 関数

<defun> を使用して定義された関数に、引数を定義します。引数は変数に似ていますが、関数に渡す順に定義する必要があります。

例

```
<defarg name='arg1'/>
<defarg name='arg2'/>
```

defun 関数

新しい関数を定義します。関数の引数を宣言するときは、<defarg> 関数を使用します。関数の実行には、<call> 関数を使用します。関数は、通常はフォーム内で定義されます。

例

```
<defun name='add100'>
   <defarg name='input'/>
   <add>
      <ref>input</ref>
      <i>100</i>
   </add>
</defun>
```

call 関数

ユーザー定義の関数を呼び出します。呼び出しの引数には、<defarg> 関数で定義した引数が適用されます。call 関数の引数の順序は、<defarg> の順序と同じである必要があります。以前のリリースでは、call 関数を使用して規則を呼び出すことができました。今後は、rule 関数を使用してください。

例

次の式は 142 を返します。

```
<call name='add100'>
   <i>42</i>
</call>
```

rule 関数

規則を呼び出します。規則の引数は、argment 要素を使用して、名前で指定して割り当てます。引数の値は、単純な文字列であれば、value 属性を使用して設定できます。value 属性を省略し、argment 要素の本文に式を記述することで、引数の値をその式で計算することもできます。

呼び出す別の規則の名前を動的に計算することで、<rule> 要素は別の規則も呼び出すことができます。

フォームやワークフロー内での規則の作成または呼び出しの詳細については、「規則」の章を参照してください。

例

次の式は、指定したユーザーの従業員 ID を返します。

```
<rule name='getEmployeeId'>
   <argument name='accountId' value='maurelius'/>
</rule>


<rule name='getEmployeeId'>
   <argument name='accountId'>
      <ref>username</ref>
   </argument>
</rule>
```

次の式は、返される値を計算する別の規則を呼び出します。

```
<rule>
   <cond>
     <eq><ref>var2</ref><s>specialCase</s></eq>
     <s>Rule2</s>
     <s>Rule1</s>
```

```
    </cond>
    <argument name='arg1'>
      <ref>variable</ref>
    </argument>
</rule>
```

オブジェクト操作式

式に含まれる任意のオブジェクトの値を操作するときは、次の関数を使用します。

get 関数

オブジェクト内から値を取得します。

最初の引数 - リスト、マップ、またはオブジェクトである必要があります。

2 番目の引数 - 文字列または整数である必要があります。最初の引数がリストの場合は、2 番目の引数は強制的に整数と扱われ、リストのインデックスとして使用されます。最初の引数が GenericObject の場合は、2 番目の引数は JavaBean プロパティーの名前と見なされます。

最初の引数のデータ型が list である場合、この関数の動作は異なります。最初の引数が list である場合は、2 番目の引数は整数のリストインデックスと見なされます。そのインデックスに該当する要素が返されます。

例

次の式は、ユーザーに現在割り当てられているロールの名前の文字列を返します。

```
<get>
   <!-- リスト、マップ、またはオブジェクト -->
   <!-- 文字列 -->
</get>
```

この式は、Java コードで `userView.getRole()` を呼び出した場合と同じ機能を持ちます。

putmap 関数

オブジェクトにマップ要素を割り当てます。

`map` - マップを指定します。

`key` - マップキーを指定します。

`value` - マップキーに割り当てる値を指定します。

例

```
<putmap>
   <ref>userView</ref>
   <s>waveset.role</s>
   <s>engineering</s>
</putmap>
```

setlist 関数

オブジェクトにリスト要素を割り当てます。

`list` - リストを指定します

`index` - リスト内の要素の順序を指定します

`value` - リスト要素に割り当てる値を指定します

例

```
<setlist>
   <ref>myList</ref>
   <i>s</i>
   <s>accounts</s>
</setlist>
```

setvar 関数

変数の値を設定します。この関数は、静的な変数名を受け付けます。

`name` - 変数の名前を指定します

`value` - 変数に割り当てる値を指定します

例

```
<setvar>
   <ref>var</ref>
   <s>text</s>
</setvar>
```

instanceof

オブジェクトが、`name` パラメータに指定されたタイプのインスタンスであるかどうかを判断します。

`name` - 調べるオブジェクトタイプを指定します。

この関数は、部分式のオブジェクトが `name` パラメータに指定されたタイプのインスタンスであるかどうかに応じて、1 または 0 (true または false) を返します。

例

ArrayList のデータ型は List であるため、次の式は 1 を返します。

```
<instanceof name='List'>
   <new class='java.util.ArrayList'/>
</instanceof>
```

Java および JavaScript 式

式で Java クラスまたは JavaScript の呼び出しと操作を行うには、次の関数を使用します。

invoke 関数

Java オブジェクトまたはクラスでメソッドを呼び出します。

この関数には、次の 2 つの形式があります。

静的メソッド

```
<invoke class='class name' name='method name'>
   <!-- メソッド引数 0 -->
   <!-- メソッド引数 n-->
</invoke>
```

インスタンスメソッド

```
<invoke class='method name'>
   <!-- メソッドが呼び出されるオブジェクト -->
   <!-- メソッド引数 0 -->
   <!-- メソッド引数 n-->
</invoke>
```

この関数を使用するには、呼び出すクラスとメソッドの名前、それぞれがとる引数、およびメソッドの機能を理解する必要があります。この関数は、次の Identity Manager クラスの呼び出しに頻繁に使用されます。

- FormUtil
- LighthouseContext
- WorkflowContext
- WavesetResult

詳細については、各クラスのドキュメントを参照してください。

new 関数

Java クラスのインスタンスを作成します。XML 属性 `class` を使用して指定されるクラス名は、クラスのパッケージ名で完全に修飾してください。

この関数を使用して新しいオブジェクトを作成し、メソッドを呼び出さずに、式または規則の値として返すこともできます。

例

```
<new class='classname'/>

  <!-- コンストラクタ引数 0 -->

  <!-- コンストラクタ引数 n -->

</new>
```

script 関数

JavaScript のフラグメントをカプセル化します。この式の評価時に、JavaScript インタプリタが呼び出され、スクリプトが処理されます。この式の値は、最後の JavaScript ステートメントの値です。スクリプト内では、`env` オブジェクトを使用して、ホストアプリケーション内の変数にアクセスできます。

フォームの `<Disable>` 式のように、パフォーマンスに大きく影響する式で JavaScript を使用することは避けてください。組み込まれているトレース機能を使用すると、短い XPRESS 式でより簡単にデバッグできます。ワークフローアクションの複雑なロジックには JavaScript を使用します。

例

```
<script>
   var arg1 = env.get('arg1');
   arg1 + 100;
</script>


<script>
   importPackage(Packages.java.util);
   var cal Now = Calendar.getInstance();
   cal Now.getTime()
</script>
```

デバッグと検証の式

トレースを有効にすると、豊富なトレースデータが生成されます。

式の問題の診断に役立つ式のトレース、またはテキストの出力を有効にするには、次の関数を使用します。

注	トレースをグローバルに有効化すると、出力されるトレースデータが膨大になる可能性があります。ブロックレベルでのトレースを有効にしたほうが便利なときは、ブロック要素の trace 属性を true に設定します。

trace 関数

式のトレースを有効または無効にします。引数の評価が `true` になると、トレースが有効になります。

トレースを有効にすると、トレースデータが標準出力に出力されます。

例 1

```
<trace><i>1</i></trace>
```

例 2

```
<trace><i>0</i></trace>
```

print 関数

各部分式の値を標準出力に出力します。

例

```
<print>
   <s>Ashley World!</s>
</print>
```

データ型

すべての関数は、次の表に示されるいずれかのデータ型の値を返します。

表 4-1 戻り値のデータ型

データ型	定義
`integer`	符号が付けられた整数値を表します。値の精度は、32 ビット以上です。
`list`	順序が付けられた、その他の値のリストを表します。リストに含まれる値は、要素と呼ばれます。NULL もリスト要素になることができます。要素を持たないリストの値は、NULL 値として認識されません。
`null`	データの欠落を表します。副次的な影響のみを目的として呼び出された場合や、指定された引数から意味のある値を計算できない場合、関数は NULL を返すことがあります。NULL 値の処理方法は、NULL 引数が渡された関数によって異なります。一般的には、NULL 値は論理 false と見なされ、演算式では無視されます。
`object`	XPRESS 言語以外で定義された任意のオブジェクトへの参照を表します。
`string`	文字列を表します。XML 構文では、文字列は常に Unicode 文字セットを使用します。string 値に文字を含めないこともできます。このような文字列は空と見なされますが、これは NULL ではありません。

一部の関数は、引数の値を論理 true または false と見なします。XPRESS では、ブール型は使用されません。その代わりに、NULL 値または整数のゼロが false と見なされます。それ以外の値は、true と解釈されます。

ブール型の値を返す `eq` などの論理関数は、false を表すために整数のゼロを返し、true を表すために整数の 1 を返します。

第 5 章

XML オブジェクト言語

XML オブジェクト言語は、ストリング、リスト、マップなどの一般的な Java オブジェクトを表現できる XML 要素の集合です。

この章の内容

関連する章

- 「XPRESS 言語」- 式を使用して、フォームにロジックを設定します。

XML オブジェクト言語について

XML オブジェクトはフォームで頻繁に使用されますが、ワークフローと規則で使用することもできます。もっとも一般的な用途は、フォームの `Select` フィールドまたは `MultiSelect` フィールドで許可される値のリストの作成です。次に例を示します。

例

```
<Field name='global.state'>
    <Display class='Select'>
       <Property name='title' value='State'/>
       <Property name='allowedValues'>
         <List>
            <String>Alabama</String>
            <String>Alaska</String>
            <String>Arizona</String>
            <String>Arkansas</String>
            <String>California</String>
            <String>Washington</String>
            <String>Washington D.C.</String>
            <String>West Virginia</String>
            <String>Wisconsin</String>
            <String>Wyoming</String>
         </List>
       </Property>
    </Display>
</Field>
```

XML オブジェクト言語の要素は、XPRESS 言語の要素に似ていますが、値が静的な場合は、XML オブジェクト言語を使用するほうが効率的です。

この 2 つの言語の主な相違点は、XML オブジェクト言語では、オブジェクトの内容を、式を使用して計算できないことです。この制限があるために、システムはオブジェクトをより効率的に作成することができ、オブジェクトが大きい場合の処理が迅速になります。

XML オブジェクト言語を使用してリストを定義すると、リポジトリからフォームが読み込まれるときにリストが作成され、以後はそれが再利用されます。XPRESS でリストを定義した場合は、フォームを表示するたびに新しいリストが作成されます。

XML オブジェクト言語とそれに対応する XPRESS

次の表は、基本的な XML オブジェクトと、それに対応する XPRESS 式 (存在する場合) を示しています。

表 5-1 基本的な XML オブジェクトとそれに対応する XPRESS 式

XML オブジェクト言語	XPRESS 言語
<String>cat</String>	<s>cat</s>
<Integer>10</Integer>	10
<Boolean>true</Boolean>	1
<Boolean>false</Boolean	0
<null/>	<null/>
<Map>	<map>
<MapEntry key='name' value='neko'/>	<s>name</s>
<MapEntry key='ID' value='123'/>	<s>neko</s>
</Map>	<s>ID</s>
	123
	</map>
<List>	<list>
<String>cat</String>	<s>cat</s>
<String>dog</String>	<s>dog</s>
<integer>673</Integer>	673
</List>	</list>
<Long>123456789</Long>	なし
<Date>20020911 09:15:00</Date>	なし

XML オブジェクト内で XPRESS ステートメントを使用することはできません。

XPRESS での XML オブジェクトの使用

式の記述が許可される場所であれば、XPRESS 内で XML オブジェクトを使用できます。次の例では、呼び出されたメソッドの引数としてマップが渡されます。

```
<invoke name='printTheMap'>
    <ref>mapPrinter</ref>
       <Map>
       </Map>
</invoke>
```

2.0 より前のリリースの XPRESS では、すべての XML オブジェクトを <o> 要素で囲む必要がありました。この制約は適用されなくなりましたが、XPRESS を含む古いファイルでは、このような設定をまだ見かけることがあります。

XPRESS の代わりに使用される XML オブジェクト言語

XML オブジェクト言語と XPRESS はどちらもフォーム内のリストを表現する方法を提供しますが、リストが長く、静的なデータを含んでいる場合は、XML オブジェクト構文を使用するほうが XPRESS より効率的です。リストはメモリ内に 1 回作成され、参照のたびにそれが再利用されます。一方、XPRESS のリスト構文は、参照のたびに評価され、毎回新しいリストが作成されます。

次の表に示される情報のリストを作成するときは、通常、XML オブジェクト言語が使用されます。

表 5-2 情報リストでの XML の用途

情報リストの種類	使用される場所
マシン名	フォーム
ビジネスサイト	フォーム
承認者名	ワークフロー

XML オブジェクト言語と XPRESS でのリストの表現

XML オブジェクト言語と XPRESS は、どちらもフォーム内のリストを表現する方法を提供します。

XPRESS によるリストの表現

XPRESS でリストを表現するときは、<list> 要素を使用します。<list> 要素には、任意の XPRESS 式を含めることができます。

注 計算される要素を含むリストで使用できるのは、<list> XPRESS 要素のみです。<list> 要素を使用すると、その要素が含まれるフォームの実行速度が低下する場合があります。リストに大量の要素が含まれる場合を除き、このパフォーマンス低下に気が付くことはほとんどありません。フォームでの <list> の使用は許容範囲内であり、よく使用されます。

次の例では、XPRESS リストで文字列定数 <s> を使用していますが、<invoke> 要素または <concat> 要素を使用して、リスト要素を動的に作成することもできます。

例

```
<list>
   <s>cat</s>
   <s>dog</s>
</list>
```

XML オブジェクト言語によるリストの表現

XML オブジェクト言語でリストを表現するときは、<List> 要素を使用します。<List> 要素に含めることができるのは、その他の XML オブジェクトのみです。次の例の <List> 要素の内容は、<String> 要素です。

例

```
<List>
   <String>cat</String>
   <String>dog</String>
</List>
```

両タイプの構文を使用したフォームの例

次のフォームには、XML オブジェクト構文と XPRESS の両方によって定義されたリストを含むフィールドが組み込まれています。

```
<Form>

    <Field name='department'>
      <Display class='Select'>
        <Property name='allowedValues'>
          <List>
           <String>Engineering</String>
           <String>Marketing</String>
           <String>Sales</String>
          </List>
        </Property>
      </Display>
    </Field>


    <Field name='department2'>
      <Display class='Select'>
        <Property name='allowedValues'>
          <expression>
            <list>
              <s>Engineering</s>
              <s>Marketing</s>
              <s>Sales</s>
            </list>
          </expression>
        </Property>
      </Display>
    </Field>

</Form>
```

department フィールドの allowedValues リストは、<List> を使用して作成される静的なリストとして定義されています。このフォームを何回使用する場合でも、作成されるリストは 1 つのみです。反対に、department2 フィールドの allowedValues リストは、<list> 式を使用して定義されています。このフォームを使用するたびに、新しいリストが作成されます。

XML オブジェクト構文と XPRESS によるマップオブジェクトの定義

XML オブジェクト構文と XPRESS のどちらを使用しても、マップオブジェクトを動的に作成できます。XPRESS の <map> 要素の使用は、XML オブジェクト言語の <Map> 要素と <MapEntry> 要素の使用に似ています。これらの要素の違いは、<map> の内容を式を使用して計算できることです。一方、<Map> 要素を使用して定義できるのは、静的なマップのみです。

注　マップは、<invoke> 式を使用して呼び出すメソッドの引数として使用される場合があります。たとえば、FormUtil クラスの一部のメソッドは、引数としてマップを必要とします。

XPRESS によるマップの表現

XPRESS の <map> 要素の内容は、名前と値のペアの式です。偶数番号の式はマップキーを定義し、奇数番号の式はマップ値を定義します。キー式の評価が NULL となる場合は、そのエントリは無視されます。

XPRESS の <map> 要素を使用することで、java.util.HashMap オブジェクトを動的に作成することができます。

```
<map>
   <s>name</s>
   <s>Jeff</s>
   <s>phone</s>
   <s>338-1818</s>
</map>
```

XML オブジェクト構文によるマップオブジェクトの定義

XML オブジェクト構文を使用して、マップオブジェクトを次のように定義できます。

```
<Map>
   <MapEntry key='name' value='Jeff'/>
   <MapEntry key='phone' value='338-1818'/>
</Map>
```

第 6 章

HTML 表示コンポーネント

この章では、Identity Manager の HTML 表示コンポーネントライブラリについて説明します。HTML 表示コンポーネントは、フォームのカスタマイズで使用されます。フォームのカスタマイズについては、61 ページの第 2 章「Identity Manager フォーム」を参照してください。

この章の内容

この節で説明する内容は次のとおりです。

HTML 表示コンポーネント

フォームを設計するときは、ここで説明する HTML コンポーネントを使用します。フォームを作成するには、Identity Manager のフォーム XML 言語 (フォーム) を使用して、HTML 表示コンポーネントを表現します。この言語は実行時に解釈され、必要なコンポーネントが作成されます。これを使用することで、追加の Java 開発をほとんど、または一切行わずに、新しいページを動的に作成できるので、カスタマイズが大幅に簡略化されます。

HTML コンポーネントとは

HTML 表示コンポーネントは、HTML テキストの文字列を生成する Java クラスのインスタンスです。各表示コンポーネントは次の要素を持ちます。

- クラス名 (`Display` 要素のクラス属性によってフィールドに定義)。この名前は、コンポーネントクラスを識別します。コンポーネントクラスは、コンポーネントの基本動作を決定し、コンポーネントが認識するプロパティーセットを定義します。
- 1 つまたは複数のプロパティー (`Property` 要素によってフィールドに定義)。プロパティーは、フィールドの動作と外観をさらに定義します。

表示コンポーネントの指定

表示コンポーネントは、次のように指定します。

```
<Field name='Name'>
   <Display class='Class'>
      <Property name='Name' value='Value'/>
   </Display>
</Field>
```

HTML コンポーネントに関するページプロセッサの要件

HTML コンポーネントを実装するフォームには、次のページプロセッサ要件が適用されます。

非表示パラメータ

ほとんどのコンポーネントは、HTML フォームから渡されるパラメータの名前に対応した名前を持ちます。Identity Manager では、一般的な用途のために、いくつかのパラメータ名が予約されています。これらの名前をコンポーネント名に使用しないでください。

表 6-1　非表示パラメータ

予約されている名前	説明
id	編集するオブジェクトの ID が格納されます
command	フォームの送信に使用されるボタンの値が格納されます
activeControl	フォームで最後にアクティブであったコンポーネントの名前を格納します
message	ページの上部に表示される情報メッセージを格納できます
error	ページの上部に表示されるエラーメッセージを格納できます

コンポーネントクラス

HTML コンポーネントは、さまざまな方法で組み合わせることができる独立オブジェクトです。関連するコンポーネントはクラスにまとめられます。コンポーネントクラスには、次の 2 つの主要グループがあります。

- **基本コンポーネントクラス** - 1 つの値を表示、編集するためのコンポーネント
- **コンテナクラス** - 1 つまたは複数のコンポーネントを格納できるコンポーネント

基本コンポーネントクラス

共通コンポーネントクラスには、1 つの値を表示、編集するためのコンポーネントが含まれます。これらのコンポーネントについては、「基本コンポーネント」の節を参照してください。

コンテナクラス

コンテナクラスは、特定の方法で視覚的に構成されたコンポーネントの集合を定義します。コンテナクラスを作成すると、通常は HTML の `table` タグが生成されます。単純なコンポーネントは、コンポーネントを縦または横に並べることができます。より複雑なコンテナは、より複雑にコンポーネントを配置でき、コンポーネントの周囲に装飾を加えることができる場合もあります。

コンテナ自体がコンポーネントであるため、どのコンテナも別のコンテナに含めることができます。このメカニズムを利用することで、複雑なページレイアウトを作成できます。たとえば、多くのページはタイトルを持ち、その後に編集フィールドのリスト、さらにフォーム送信ボタンの行を持ちます。このようなレイアウトを作成するには、垂直整列を使用して `Panel` コンポーネントを作成し、`Label`、`EditForm`、および `ButtonRow` コンポーネントを含めます。`EditForm` コンポーネント自体にも、多数のサブコンポーネントが含まれます。`ButtonRow` は、垂直整列を使用する `Panel` コンポーネントで、`Button` コンポーネントのリストがここに含まれます。

BorderedPanel

項目を配置するための 5 つの領域 (東西南北と中央) を定義します。北と南の領域に配置されたコンポーネントは、水平に並べられます。それ以外の領域に配置されたコンポーネントは、垂直に並べられます。

次のプロパティーがあります。

- `eastWidth` - 東の領域の幅を指定します
- `westWidth` - 西の領域の幅を指定します

ButtonRow

ボタン配置用のデフォルトオプションを設定します。Panel コンポーネントを拡張します。

- buttonDivStyle - ボタンを囲む div に適用する CSS クラスを指定します
- defaultAlign - その行のボタンのデフォルト配置を指定します。align プロパティーがその行に明示的に設定されていない場合には、Identity Manager はこのプロパティーを参照します。デフォルトは left です。
- defaultDivider - そのボタン行の上下に境界バーを表示するかどうかを指定します。divider プロパティーがその行に明示的に設定されていない場合には、Identity Manager はこのプロパティーを参照します。デフォルトは false です。
- divider - 境界バーを水平の線で表示するか、空白行として表示するかを指定します。true に設定すると、境界バーは水平線として表示されます (たとえば、<hr>) (ブール型)。
- dividerStyle - 境界バーを表示する場合に、そのスタイルを設定するために使用する CSS クラスを指定します。このプロパティーが設定されていない場合は、Identity Manager は横罫線を表示します。デフォルトは unset です。
- pad - ボタン行と隣接コンポーネントの間のどこに空白を挿入するかを指定します。指定できるのは、上下の間隔 (top、bottom) です。値が NULL の場合は、間隔はなくなります。デフォルト値は top です。

EditForm

この表示コンポーネントは、フォームをブラウザに表示するときのデフォルトの表示クラスです。

フォームコンポーネントは 2 つの列に配置され、左がタイトル、右がコンポーネントとなります。タイトルには吹き出しヘルプを含めることができます。複数のコンポーネントを 1 つの行に連結できます。

よく編集されるプロパティーは、title、subTitle、および adjacentTitleWidth です。

```
<Form name='Default User Form' help='account/modify-help.xml'>
   <Display class='EditForm'>
     <Property name='titleWidth' value='120'>
     <Property name='adjacentTitleWidth' value='60'>
   </Display>
```

EditForm には、これ以外に次のプロパティーがあります。

- adjacentTitleWidth - 隣接するフィールドのタイトルの幅を指定します。このプロパティーが定義されていない場合は、デフォルト値のゼロが適用されます。adjacentTitleWidth の値をゼロに設定すると、列のタイトルの大きさは自動的に変更されます。ゼロ以外の値を設定した場合は、隣接する列 (たとえば、2 番目と 3 番目の列) のタイトルの幅は adjacentTitleWidth の値に設定されます。
- border - EditForm コンポーネントを含むテーブルの幅をピクセル単位で指定します。デフォルトは 0 で、境界線が表示されないことを示します。
- cellpadding - EditForm コンポーネントを含むテーブルのセルパディングをピクセル単位で指定します。デフォルトは 5 です。
- cellspacing - EditForm コンポーネントを含むテーブルのセル間隔をピクセル単位で指定します。デフォルトは 0 です。
- componentTableWidth - EditForm の幅を指定します (ピクセル単位)。指定しない場合は、デフォルト値の 400 ピクセル、または EditForm の defaultComponentTableWidth グローバルプロパティーの値が適用されます。
- defaultComponentTableWidth - Identity Manager が各コンポーネントを表示するためのテーブルの幅をピクセル単位で指定します。componentTableWidth プロパティーが EditForm に明示的に設定されていない場合には、Identity Manager はこのプロパティーを参照します。このコンポーネントが設定されていない場合は、コンポーネントテーブルの幅は指定されません。
- defaultRequiredAnnotationLocation - requiredAnnotation のデフォルトの表示位置をコンポーネントとの相対位置 (left、right、または none) で指定します。requiredMarkerLocation プロパティーが EditForm に明示的に設定されていない場合には、Identity Manager はこのプロパティーを参照します。デフォルトは right です。
- evenRowClass - EditForm テーブルの偶数行のスタイルを設定するために使用する CSS クラスを指定します (noAlternatingRowColors プロパティーが true に設定されていない場合)。デフォルトは formevenrow です。
- helpIcon - コンポーネントの吹き出しヘルプメッセージに対して表示するアイコンを指定します。デフォルトは images/helpi_gold.gif です。
- noAlternatingRowColors - EditForm の行を同じ色で表示するかどうかを指定します。noAlternatingRowColors を true に設定すると、EditForm のすべての行が同じ色で表示されます。指定しない場合は、デフォルト値の false が適用されます。
- oddRowClass - EditForm テーブルの奇数行のスタイルを設定するために使用する CSS クラスを指定します (noAlternatingRowColors プロパティーが true に設定されていない場合)。デフォルトは formoddrow です。
- requiredAnnotation 必須フィールドの横に表示する注釈を指定します。デフォルトの画像は、赤色のアスタリスクです。

- requiredClass - 必須フィールドの凡例のスタイルを設定するために使用する CSS クラスを指定します。デフォルトは errortxt です。
- requiredLegendLocation - フォームに必須フィールドがある場合に、必須凡例を表示する位置 (top または bottom) を指定します。デフォルトは bottom です。
- rowPolarity - テーブル内で灰色の行と白色の行を交互に表示するときの極性を指定します。デフォルトは true です。値を false にすると、極性が反転して、最初のフォームフィールドの背景が白色になります。次のコード例では、テーブルの最初のフォームフィールドの背景が白色になります。

```
<Display class='EditForm'>
    <Property name='componentTableWidth' value='100%'/>
    <Property name='rowPolarity' value='false'/>
    <Property name='requiredMarkerLocation' value='left'/>
    <Property name='messages'>
      <ref>msgList</ref>
    </Property>
</Display>
```

- tableClass - EditForm コンポーネントを含むテーブルのスタイルを設定するために使用する CSS クラスを指定します。
- tableWidth - Identity Manager が EditForm コンポーネントを表示するためのテーブルの幅をピクセル単位で指定します。デフォルトは 400 です。
- titleClass - コンポーネントのヘルプメッセージのスタイルを設定するために使用する CSS クラスを指定します。

Menu

Menu、MenuBar、および MenuItem の 3 つのクラスで構成されます。

- Menu はコンポーネント全体のことです。
- MenuItem はリーフ (ノード) で、第 1 レベルまたは第 2 レベルのタブに相当します。
- MenuBar は、MenuBars または MenuItems を含むタブに相当します。

Menu には次のプロパティーが含まれます。

- layout - 値は horizontal または vertical です (String)。値 horizontal では、タブ付きの水平方向ナビゲーションバーが生成されます。値 vertical では、一般的なノードレイアウトの垂直方向ツリーメニューとして、メニューが表示されます。

- `stylePrefix` - CSS クラス名用のプレフィックス (String)。Identity Manager エンドユーザーページの場合は、この値は `User` になります。

`MenuBar` には次のプロパティーが含まれます。

- `default` - URL パス (String) で、MenuBar の MenuItem URL プロパティーのいずれかに相当します。`MenuBar` タブがクリックされたときにデフォルトで `selected` として表示されるサブタブを制御します。

`MenuItem` には次のプロパティーが含まれます。

- `containedUrls` - MenuItem に関連付けられた JSP への URL パスのリスト。`containedUrls` JSP が表示される場合は、現在の MenuItem が「selected」として表示されます。ワークフローが要求起動ページから起動されたあとに表示される要求起動結果ページはこの例です。

`MenuBar` または `MenuItem` には次のプロパティーを設定できます。

- `title` - タブまたはツリーリーフにハイパーリンクとして表示されるテキスト (String) を指定します。
- `URL` - タイトルハイパーリンクの URL パス (String) を指定します。

次の XPRESS の例では、2 つのタブを持つメニューが作成されます。2 番目のタブには 2 つのサブタブがあります。

コード例 6-1 Menu、MenuItem、および MenuBar コンポーネントの実装

```
<Display class='Menu'/>
<Field>
   <Display class='MenuItem'>
     <Property name='URL' value='user/main.jsp'/>
     <Property name='title' value='Home' />
   </Display>
</Field>
<Field>
    <Display class='MenuBar' >
      <Property name='title' value='Work Items' />
     <Property name='URL' value='user/workItemListExt.jsp' />
   </Display>
    <Field>
       <Display class='MenuItem'>
         <Property name='URL' value='user/workItemListExt.jsp'/>
         <Property name='title' value='Approvals' />
        </Display>
     </Field>
    <Field>
      <Display class='MenuItem'>
         <Property name='URL' value='user/otherWorkItems/listOtherWorkItems.jsp'/>
          <Property name='title' value='Other' />
       </Display>
    </Field>
 </Field>
```

Identity Manager ユーザーインタフェースで、水平方向ナビゲーションバーは enduser.xml 内のエンドユーザーナビゲーションユーザーフォームによって呼び出されます。

userHeader.jsp は、すべての Identity Manager ユーザーインタフェースページに含まれますが、これには menuStart.jsp という名前の別の JSP が含まれています。この JSP から、2 つのシステム設定オブジェクトが呼び出されます。

- ui.web.user.showMenu - ナビゲーションメニューの表示のオン / オフを切り替えます (デフォルトは true)。
- ui.web.user.menuLayout - メニューをタブ付きの水平方向ナビゲーションバー (デフォルト値は horizontal) として表示するか、または垂直方向ツリーメニュー (vertical) として表示するかを決定します。

style.css には、メニューの表示方法を決定する CSS スタイルクラスが含まれています。

Panel

もっとも基本的なコンテナを定義します。Panel は、単純な直線リストに子要素を表示します。

次のプロパティーがあります。

- horizontal - true に設定すると、コンポーネントが水平に配置されます (ブール型)。
- horizontalPad - 水平配置コンポーネントの周囲の、テーブルのセルパディングの属性を、ピクセル単位で指定します。
- verticalPad - コンポーネント間に挿入する空白行の数を指定します。(ブール型)

デフォルトの配置は垂直ですが、水平に設定することができます。

Selector

単一値または複数値のフィールド (それぞれ Text、ListEditor コンポーネントに相当) と、その下の検索フィールドを設定します。検索を実行すると、Identity Manager で検索フィールドの下に結果が表示され、値フィールドに結果が取り込まれます。

その他のコンテナコンポーネントとは異なり、Selector は値を持ちます (search の結果が反映されるフィールド)。このコンポーネントには、通常、検索条件フィールドが含まれます。Selector は、検索結果の内容を表示するためのプロパティーを実装します。

次のプロパティーがあります。

- `fixedWidth` - コンポーネントの幅を固定にするかどうかを指定します (Multiselect と同じ機能) (ブール型)。
- `multivalued` - 値のデータ型が List であるか、String であるかを指定します。このプロパティーの値は、ListEditor または Text フィールドのどちらで値を表示するかを決定します (ブール型)。
- `allowTextEntry` - 用意されているリストから値を選択しなければならないか、手動で入力できるかを指定します (ブール型)。
- `valueTitle` - `value` コンポーネントに使用するラベルを指定します (String)。
- `pickListTitle` - `picklist` コンポーネントに使用するラベルを指定します (String)。
- `pickValues` - picklist コンポーネントで使用する値を指定します。NULL に設定した場合は、選択リストは表示されません (List)。
- `pickValueMap` - 選択リスト内の値の表示ラベルのマップ (Map または List)。
- `searchLabel` - 入力テキストフィールドの横のボタンに、指定のラベルを付けます。値を指定しない場合は、デフォルト値の「...」が適用されます。
- `sorted` - 選択リストの値をソートするかどうかを指定します。multivalued であり ordered でない場合は、値リストもソートされます (ブール型)。
- `clearFields` -「クリア」ボタンをクリックしたときに、リセットするフィールドのリストを指定します (List)。

次のプロパティーは、複数値のコンポーネントでのみ有効です。

- `ordered` - 値の順序を保持するかどうかを指定します (ブール型)。
- `allowDuplicates` - リストに重複値を含めることができるかどうかを指定します (ブール型)。
- `valueMap` - リスト内の値の表示ラベルのマップを指定します (Map)。

次のプロパティーは、単一値のコンポーネントでのみ有効です。

- `nullLabel` - NULL 値として解釈される値のラベルを指定します (String)。

SimpleTable

コンポーネントをグリッドに配置します。オプションとして、上部に列のタイトル行も設定できます。

次のプロパティーがあります。

- `columns` - 列の見出しを定義します。通常はメッセージキーのリストですが、単純な文字列を定義することもできます (List)。
- `rows` - テーブルのセルを定義します。各セルは、コンポーネントでなければなりません (List)。

- columnCount - 列見出しのリストが指定されていない場合に、列の数を指定します。
- border - テーブルの境界線の幅を指定します。値を 0 に設定すると、境界線は非表示になります。
- noItemsMessage - 行がない場合に、テーブルに表示するメッセージを指定します。

TabPanel

次に示されるタブの行を表示する、タブ付きパネルの表示に使用されます。デフォルトでは、タブは水平に並べられます。

次のプロパティーがあります。

- leftTabs - true に設定すると、上部マージンにではなく、左マージンにそろえてタブが配置されます (ブール型)。
- border - true に設定すると、タブの下のメインパネルの周囲に境界線を表示します (ブール型)。
- renderTabsAsSelect - true に設定すると、タブの代わりに選択ドロップダウンとしてタブを表示します。これは、ブラウザの水平スクロールが必要になるほどタブの数が多いフォームで便利です。タブの左マージン配置と組み合わせて使用しないでください。
- tabAlignment - ページ内容からの相対位置としてタブの位置を決定します。有効な値は、left (左、デフォルト設定)、top (上)、right (右)、bottom (下)、center (中央)、および middle (中間) です。
- validatePerTab -- true に設定すると、ユーザーが別のタブに切り替えた直後に Identity Manager によって検証式が実行されます。

```
<Field name='MainTabs'>
   <Display class='TabPanel'>
     <Property name='leftTabs' value='false'/>
     <Property name='tabAlignment' value='left'/>
</Field>
```

Row

水平配置に対応したパネルの作成に使用されます。

SortingTable

列の見出しを指定して内容をソートできるテーブルの作成に使用されます。このテーブルの内容は、子コンポーネントによって定義されます。列ごとに 1 つの子コンポーネントを作成してください (columns プロパティーで定義)。通常は、列は FieldLoop に格納されます。

テーブルセルの表示時に、このコンポーネントには、子コンポーネントの align、valign、および width プロパティーが適用されます。

次のプロパティーがあります。

- emptyMessage - テーブルに行がないときにテーブルに表示する文字列またはメッセージキーを指定します。このプロパティーを省略すると、Identity Manager は汎用メッセージを表示します。
- pageButtonAlign - ページ内容からの相対位置としてボタンの位置を決定します。有効な値は、left (左)、right (右)、bottom (下)、および center (中央) です。デフォルト値は、right です。
- sortEnable - true に設定すると、列のソートが有効になります (ブール型)。
- sortURL - 列のソートを選択したときに Identity Manager が送信する URL を指定します。列のソートを設定しない場合は、HtmlPage の _postURL が使用されます (String)。
- sortURLParams - sortURL とともに渡されるパラメータを指定します (String)。
- sortColumn - 現在ソート基準として使用されている列の番号を指定します。デフォルトでは、この値は最初の列に設定されます (Integer)。
- sortOrder - ソート順序を指定します。指定できる値には、asc (昇順) と desc (降順) があります。デフォルト値は asc です (String)。
- linkEnable - 最初の列をリンク付きに指定して、テーブルを生成するかどうかを指定します (ブール型)。
- linkURL - リンクの生成時に Identity Manager がリンクする URL を指定します。指定しない場合は、そのコンポーネントが使用される HtmlPage の送信 URL がデフォルト値として適用されます (String)。
- linkURLArguments - リンク URL に含める引数を指定します。
- linkColumn - linkURL 属性によって指定されるリンク生成で使用される列の番号を指定します (Integer)。
- linkParameter - リンク行 id の値を持つ送信データパラメータの名前を指定します。デフォルト値は id です。
- selectEnable - チェックボックスの列を、複数選択テーブルの左マージンにそろえて表示するかどうかを指定します。true に設定すると、Identity Manager はチェックボックスの列を表示します (ブール型)。

- columns - テーブル列の見出しのリストを指定します (文字列の List)。
- pageSize - テーブルに同時に表示する _pageSize の最大エントリ数を指定します。_pageSize より多くのエントリが存在する場合は、インタフェース要素で結果のページを切り替えることができます。_pageSize に 1 (デフォルト設定) を下回る値を設定すると、すべてのエントリが一度に表示されます (Integer)。
- useSavedPage - pageSize の値が 0 より大きい場合にテーブルをソートすると、現在の HTTP セッションのソートテーブルページが <fieldName>_currentPage 属性に保存されます。_useSavedPage プロパティーは、現在のページを HTTP セッションから取得して表示するかどうかを指定します。このプロパティーの値を XPRESS 式の結果にすることで、SortingTable コンポーネントが含まれる JSP に戻るときに、現在のページをいつ呼び出しなおすかをフォームまたはビューで制御できます (ブール型)。

たとえば、SortingTable コンポーネントが編集可能な項目を含むクエリー結果を表示する場合、ユーザーが結果テーブルの項目を編集したあとに、編集後の項目を含む結果ページが Identity Manager で必ず表示されるようにするには、0 より大きな値を指定します。

WizardPanel

いくつかの子コンポーネントの 1 つ (通常は EditForms) の表示に使用されます。コンポーネント間の移動には、ウィザード形式の「次へ」および「戻る」ボタンが使用されます。

次のプロパティーがあります。

- button - 子コンポーネントの値をボタン行に表示する、子コンポーネントの location プロパティーの値を指定します (String)。
- nextLabel -「次へ」ボタンに表示するラベルを指定します。デフォルト値は、Next です (String)。
- prevLabel -「戻る」ボタンに表示するラベルを指定します (String)。
- cancelLabel -「キャンセル」ボタンに表示するラベルを指定します (String)。
- okLabel -「OK」ボタンに表示するラベルを指定します (String)。
- noOk -「OK」ボタンを非表示にするかどうかを指定します (ブール型)。
- alwaysOk - true に設定すると、「OK」ボタンが表示されます (ブール型)。
- noCancel - true に設定すると、「キャンセル」ボタンは表示されません (ブール型)。
- topButtons - true に設定すると、ページの下部ではなく、ページの上部にボタンが表示されます (ブール型)。
- noButtons - true に設定すると、すべてのボタンが非表示になります (ブール型)。

コンポーネントサブクラス

すべてのコンポーネントは、ほとんどのコンポーネントに共通するプロパティーを定義した Component クラスを拡張します。また、一部のコンポーネントは Container クラスを拡張し、別のコンポーネントを格納できるようにします。

各 Component サブクラスは、コンポーネントの基本クラスより詳細にコンポーネントの特性を指定する、数多くのプロパティーを定義します。たとえば、Label コンポーネントは font プロパティーをサポートします。このプロパティーは、ラベルの表示に適用されるフォントを指定します。

表記上の規則

プロパティー名は、常に小文字から始まり、隣接する単語との区切りにはキャメルケース (2 番目以降の複合単語の頭文字を大文字で表記) が使用されます。アクセスメソッド名はプロパティー名の大文字表記で、get または set のプレフィックスが付けられます。たとえば、font というプロパティーには、getfont および setfont メソッドを利用して、Java からアクセスできます。

各プロパティーのデータ型は多様です。各プロパティーの説明を参照してください。次の表は、プロパティー値のデータ型を表現する用語を示しています。

データ型

次の表は、コンポーネントのプロパティーで使用できるデータ型を示しています。

表 6-2　HTML コンポーネントプロパティーのデータ型

データ型	説明
null	プロパティーが値を持たないことを表します
String	もっとも一般的なデータ型です。String 値は、通常は Java の String クラスのインスタンスで表現されます。一部のコンポーネントは、任意のクラスの値です。これは、強制的に、文字列に toString メソッドを適用することを意味します。 特に明記されていない限り、すべてのプロパティーのデータ型は String であると考えてください。 例 : <String>Hello World</String>

表 6-2　HTML コンポーネントプロパティーのデータ型 (続き)

データ型	説明
文字列の List	値が、1 つまたは複数の文字列のリストであることを表します。Java では、この値は常に List クラスのインスタンスとして実装されます。また、リストを構成する要素は、String クラスのインスタンスです。 例 : <List> <String>choice one</String> <String>choice two</String> </List>

基本コンポーネントクラス

Component クラスは、すべての HTML コンポーネントの基本クラスです。このクラスには、ほとんどのコンポーネントに共通するプロパティーが含まれます。Component クラスのすべてのプロパティーが、すべてのサブクラスに関連するわけではありません。たとえば、Component は、値の制約リストを含めることができる allowedValues プロパティーを定義します。このプロパティーは、Select や MultiSelect のように、値を編集できるサブクラスにのみ関連します。一方で、Container クラスは、編集可能な値を直接表すことは、ほとんどありません。このため、コンポーネントの値に関連するプロパティーとは無関係です。一部のプロパティーは、コンポーネントが特定の Container クラスに含まれる場合にのみ関係します。

name

フィールドの内部名を指定します。すべての編集コンポーネントは名前を持ちます。この名前は、通常、ページに表示されるその他すべてのコンポーネント名と重複しません。多くの場合は、name はビュー属性へのパスを表す文字列です。

コンテナコンポーネントは名前を必要としません。また、割り当てられているすべての名前は無視されます。コンポーネントを Java で構築する場合、コンポーネント名がアプリケーションによって定義されます。XML フォームからコンポーネントを構築する場合は、コンポーネント名はフォームの Field 要素の名前から継承されます。フィールド名は、フォームで使用されるビューオブジェクト内のパス式です。

例

```
<Field name='global.firstname'>
```

name 属性がユーザービューの特定の属性をどのように参照するかについての詳細は、「Identity Manager のビュー」を参照してください。

title

(省略可能) フィールドの外部名を指定します。タイトルは、1 つの列にタイトル、別の列にコンポーネントを持つ HTML テーブルを作成する EditForm コンテナで頻繁に使用されます。

コンポーネントは、独自のタイトルを表示しません。タイトルの表示はコンテナによって制御されます。多くのコンテナはタイトルを無視します。

例

```
<Property name='title' value='FirstName'/>
<Property name='title'>
   <expression>
      <concat>
         <s>Edit User: </s>
         <ref>waveset.accountId</ref>
      </concat>
   </expression>
</Property>
```

この例では、フィールドのタイトルの一部が、ユーザーの Identity Manager アカウント ID から動的に取得されます。

value

編集コンポーネントは、NULL 値にも設定できる値を持ちます。多くの場合、値はビューの属性に基づいて、Identity Manager によって自動的に設定されます。一部のコンポーネントでは、現在のビューの内容を無視して値を設定できます。この値は、NULL 値にすることもできます。

Component クラスでは、任意の Java オブジェクトを値として指定できます。サブクラスは、割り当て時、または HTML の生成時に、強制的に特定のデータ型を値に適用します。コンポーネントの値は、ほとんどの場合、常に String オブジェクト、または文字列を含む List オブジェクトです。コンポーネントのデータ型の詳細については、「データ型」の節を参照してください。

ほとんどのコンテナクラスは値を持ちません。割り当てた値は無視されます。ただし、一部のコンテナには値を設定できます (たとえば、TabPanel、WizardPanel)。

XML フォームからコンポーネントを構築するときは、値は、通常、基本となるビューオブジェクトへのパスを示す、コンポーネントの name から継承されます。そのビューオブジェクトには、編集の対象となるすべての値が含まれます。

例

```
<Property name= 'value' value='false '/>
```

allowedValues

コンポーネントで使用できる値のオプションリストを指定します。指定した場合、そのコンポーネントで選択できる値は、このリストに含まれる値に限定されます。コンポーネントが値の制限をサポートする場合は、使用できる値のリストはここに格納されます。この値は常にリストであり、その値は通常、文字列です。XML フォームからプロパティーを設定するときに便利なように、使用できる値をコンマ区切りのリストして指定することもできます。

例

```
<Property name='allowedValues' value= 'Mon, Tue, Wed, Thurs, Fri'/>
<Property name='allowedValues'/>
   <expression>
      <call name='DaysoftheWeek'/>
   </expression>
</Property>
```

primaryKey

このプロパティーは、SortingTable コンテナでのみ認識されます。SortingTable コンテナは、コンポーネントをテーブル形式にまとめます。各列には、同じクラスのコンポーネントが配置されます。SortingTable を使用することで、任意の列の値に基づいて行をソートできます。通常、ソート順序は、列の各コンポーネントの value によって決定されます。しかし、コンポーネントの値がソートに適さない場合や、比較に適さない場合もあります。このようなときは、それに代わる数値ソートキーを指定できます。

required

true に設定した場合は、フォームの送信前に、フィールドに値を指定しなければなりません。このコンポーネントを EditForm で使用する場合は、保存前にフィールドに値を指定しなければならないことをユーザーに示すために、コンポーネントの後に赤のアスタリスク (*) が表示されます。必須スキーママップ属性を選択した場合は (つまり、値を true に設定した場合は)、そのフィールドは常に必須フィールドとなります。

プロパティーの値は、true または false です。

例

```
<Property name='required' value='true'/>
```

noNewRow

true に設定すると、Identity Manager ページの前のフィールドの横にフィールドが表示されます。設定しない場合、または false に設定した場合は、フィールドは前のフィールド直下の新規行に表示されます。デフォルト値は false です。

このブール型プロパティーは、フィールドが EditForm 表示クラスを使用するフォームに指定されている場合にのみ認識されます。通常、EditForm は各コンポーネントを新しい行に表示し、左の列にタイトル、右の列にコンポーネントを示します。複数のコンポーネントを同じ行に連結することで、表示領域を節約できます。コンポーネントにタイトルも設定されている場合は、そのコンポーネントと前のコンポーネントの間に、強調表示されないテキストとしてタイトルが表示されます。

次の値があります。

```
value='true ' | 'false '
```

例

```
<Property name='noNewRow' value='true'/>
```

location

コンテナが複数の表示領域を定義し、特定の領域にコンポーネントを追加しなければならない場合に使用されます。一部のコンテナでは、location プロパティーに値を割り当てることでコンポーネントの配置を制御できます。たとえば、BorderedPanel コンテナは、東西南北と中央の 5 つの異なる表示領域をサポートします。

location プロパティーで認識される値は、コンテナによって定義されます。場所を割り当てない場合、または認識されない場所名を割り当てた場合は、コンテナはコンポーネントをデフォルトの場所に配置します。

help

ユーザーがフィールドの目的を理解できるように、画面に表示する説明テキストを指定します。ほとんどの Identity Manager ページでは、このコンポーネントを設定すると、コンポーネントタイトルの横に <icon> アイコンが表示されます。マウスカーソルをこのアイコンに重ねると、左マージンにヘルプテキストが表示されます。

プロパティーの値には、表示するリテラルテキスト、またはメッセージカタログキーを指定できます。リテラルテキストには HTML マークアップを含めることができます。

カスタムフォームにヘルプを追加する方法の詳細については、167 ページの「フォームへのガイダンスヘルプの追加」を参照してください。

inlineHelp

Identity Manager ページのコンポーネントの下に表示されるテキストを指定します。

プロパティーの値には、表示するリテラルテキスト、またはメッセージカタログキーを指定できます。リテラルテキストには HTML マークアップを含めることができます。

command

コンポーネントの変更時に送信するコマンドを指定します。ユーザーが値を変更すると、フォーム出力が再計算されます。

このプロパティーは、通常、Button コンポーネントで使用されます。一部のコンポーネントでは、変更が加えられた場合に、アプリケーションがその変更に応じてページを再生成できるように、そのコンポーネントが含まれる HTML フォームをただちに送信する必要があります。command プロパティーに NULL 以外の値を指定することで、この動作を設定できます。

command プロパティーを設定してコンポーネントを変更すると、フォームが送信され、command という追加の非表示パラメータも送信されます。この非表示パラメータの値は、command プロパティーの値です。

command は、ビューに加えられた変更をシステムがどのように処理するかを指定します。command プロパティーは、次のいずれかの値をとります。

表 6-3 command プロパティーの値

値	説明
Save	編集内容を保存します。
Cancel	編集内容を破棄します。
Recalculate	ページを再生成します。

表 6-3　command プロパティーの値 (続き)

値	説明
SaveNoValidate	編集内容を保存しますが、フォームの検証は行いません。

フォームでは、command はよく Recalculate に設定されるため、この設定のために短い構文が用意されています。Display 要素は action という属性を持ちますが、これを true に設定した場合、command プロパティーを **Recalculate** に設定した場合と同じ効果が得られます。

```
<Display class='Select' action='true'>"
```

onClick

値を設定すると、その値は、このコンポーネントで生成される input 要素の onClick 属性の値として割り当てられる JavaScript を持つものと見なされます。すべてのコンポーネントが onClick プロパティーをサポートするわけではありません。

このプロパティーが使用されることはあまりありません。使用する場合は、生成される HTML に関する詳しい知識が必要です。このプロパティーを使用する場合は、通常、onClick の値から呼び出す JavaScript 関数を定義する Javascript コンポーネントが、ページに必要です。

例

```
<Property name='onClick' value="Uncheck(this.form,
'resourceAccounts.selectAll');"/>
```

注　フォームがリポジトリに格納されると、Identity Manager は常に引用符 (一重) で属性値を囲みます。属性値内で引用符を使用するときは、それを ' に置き換えます。このエスケープを使用しない場合は、XPRESS の s 式で文字列を表現します。

```
<Property name='onClick'>

   <s>Uncheck(this.form, 'resourceAccounts.selectAll');
</s>

</Property>"
```

onChange

これは command コンポーネントに似ています。このコンポーネントの値には、フィールドの変更時に実行する、任意の JavaScript ステートメントを指定できます。

すべてのコンポーネントが onChange プロパティーをサポートするわけではありません。

このプロパティーが使用されることはあまりありません。使用する場合は、生成される HTML に関する詳しい知識が必要です。このプロパティーを使用する場合は、通常、onChange の値から呼び出す JavaScript 関数を定義する Javascript コンポーネントが、ページに必要です。

nowrap、align、width、valign、および colspan

ほとんどのコンテナは、HTML の table タグで囲んで、サブコンポーネントを配置しています。各コンポーネントに生成される HTML は、通常は td タグに含まれます。一部のコンテナは、nowrap、align、width、および colspan の各プロパティーを認識し、テーブルセルのタグを生成するときにそれらを使用します。これらのコンポーネントを使用することで、コンテナ内のコンポーネントの配置や大きさを調整できます。

- nowrap - 一部のコンポーネントについて、長い文字列のテキストが含まれる場合にどのように表示するかを指定します。nowrap の値を false に設定した場合、または値を指定しなかった場合は、ブラウザはコンポーネントの表示時にテキストを複数の行に分割することがあります。noWrap を true に設定すると、ブラウザはコンポーネントテキストの 1 行での表示を試みます。
- align - あまり使用されません。フォーム上の要素の横方向の位置を調整します。指定できる値は、left (左)、right (右)、および center (中央) です。
- valign - あまり使用されません。コンポーネントの縦方向の位置を指定します。指定できる値は、top (上)、bottom (下)、および middle (中間) です。
- colspan - 非推奨です。

例

```
<Property name= 'width' value='3'/>

<Field name='Start Day' prompt='Day' nowrap='true'/>
```

htmlFormName

コンポーネントが表示される HTML の <FORM> タグの名前属性を設定するときに使用されます。これにより、コンポーネントで使用される JavaScript 関数が、適切な HTML フォームを確実に参照します。デフォルト値が mainform であるため、このプロパティーは、mainform 以外のフォームにコンポーネントを表示する場合にのみ使用されます。

例

```
<Property name='htmlFormName' value='endUserNavigation'>
```

基本コンポーネント

BackLink

前のページに戻るリンクを表示します。このコンポーネントは、ブラウザの「**戻る**」ボタンと同じように機能します。ただし、このリンクはページ上の使いやすい場所に配置できます。

この表示コンポーネントには、次のプロパティーがあります。

- `text` - リンクのテキストを指定します。テキストを指定しない場合は、デフォルト値の **Back** が適用されます。

例

```
<Field name='back'>
   <Display class='BackLink'>
      <Property name='value' value='previous page'/>
   </Display>
</Field>
```

Button

ボタンを表示します。ボタンは通常、そのボタンが含まれるフォームの送信に使用されますが、任意の JavaScript を実行するように定義することもできます。

この表示コンポーネントには、次のプロパティーがあります。

- `class` - 有効になっているボタンに使用する CSS クラスを指定します。デフォルトは `formbutton` です。
- `command` - `name` パラメータとともに送信するオプション値 (たとえば、Save、Cancel、Recalculate) を指定します。
- `disabledclass` - 無効になっているボタンに使用する CSS クラスを指定します。デフォルトは `formbutton` です。
- `hiddenID` - フォーム送信データに含まれる *id* パラメータのオプション値を指定します。
- `label` - ボタンに表示されるテキストを指定します。
- `linkClass` - ボタンがリンクとして表示されるときに使用する CSS クラスを指定します。
- `name` - ユーザーがこのボタンをクリックしたときに送信されるパラメータの名前を指定します。このプロパティーは省略可能です。指定しなかった場合は、デフォルト値の `command` が適用されます。

- onMouseOver - ボタンの onMouseOver イベントで実行する Javascript を指定します。このプロパティーを使用して、ボタンの上にマウスがあるときのボタンのスタイルを変更できます。
- onMouseOut - ボタンの onMouseOut イベントで実行する Javascript を指定します。このプロパティーを使用して、ボタンからマウスが離れたときのボタンのスタイルを変更できます。
- onFocus - ボタンの onFocus イベントで実行する Javascript を指定します。このプロパティーを使用して、ボタンにフォーカスがあるときのボタンのスタイルを変更できます。
- onBlur - ボタンの onBlur イベントで実行する Javascript を指定します。このプロパティーを使用して、ボタンがフォーカスを失ったときのボタンのスタイルを変更できます。
- postURL - フォームの代替送信先 URL を指定します。この値は、JSP に指定されている URL に優先して適用されます。
- value - ユーザーがこのボタンをクリックしたときに送信されるパラメータの値を指定します。

例

```
<Display class='Button'>
   <Property name='label' value='Change Password'/>
   <Property name='value' value='Recalculate'/>
</Display>
```

Checkbox

チェックボックスを表示します。チェックボックスを選択すると、そのボックスの値は true となります。選択されていないボックスの値は false となります。

この表示コンポーネントには、次のプロパティーがあります。

- label - (省略可能) チェックボックスの右に表示されるラベルを指定します。これはコンポーネントの横に表示されますが、タイトル列には表示されません。
- leftLabel - ラベルがチェックボックスの左に表示されるように指定します。
- checkAll - このチェックボックスを「すべてを選択」チェックボックスとして機能させる場合に使用されます。この値は、その他のチェックボックスにも反映されます。プロパティーの値は、HTML ページのその他のチェックボックスの名前と一致する正規表現です。

- uncheck - 同期された一連のチェックボックスの「すべてを選択」チェックボックスを表す、別のチェックボックスフィールドの名前を指定します。これを設定すると、このチェックボックスの選択状態が変わるたびに、「すべてを選択」チェックボックスの選択が解除されます。
- syncCheck - このプロパティーが設定されているチェックボックスフィールドの値との同期を保つ、別のチェックボックスフィールドの名前を指定します。これを設定すると、このチェックボックスの値が変更されるたびに、同期されているチェックボックスに同じ値が設定されます。
- syncUncheck - このプロパティーが設定されているチェックボックスフィールドの値が false に変更された場合に同期を保つ、別のチェックボックスフィールドの名前を指定します。これを設定すると、このチェックボックスの値が false に変更されるたびに、同期されているチェックボックスの値も false (未選択) に設定されます。
- syncCheckAllTo - このプロパティーが設定されているチェックボックスフィールドの値が false に変更された場合に、正規表現と一致するすべての「すべてを選択」チェックボックスとの同期が維持されます。このプロパティーの値は、1 つまたは複数の「すべてを選択」チェックボックスを表す正規表現です。
- syncUncheckAll - このプロパティーが設定されているチェックボックスフィールドの値が false に変更された場合に同期を保つ、別のチェックボックスフィールドの名前を指定します。これを設定すると、このチェックボックスの値が false に変更されるたびに、同期されているチェックボックスの値も false (未選択) に設定されます。
- syncCheckTo - このプロパティーが設定されているチェックボックスフィールドの値は、正規表現と一致するすべてのチェックボックスと同期を保ちます。このプロパティーが設定されているチェックボックスフィールドの値が変更されるたびに、同期されているチェックボックスに同じ値が設定されます。このプロパティーの値は正規表現です。
- value - チェックボックスの状態を決定します。値が論理 true の場合は、チェックマークが表示されます。

例

```
<Field name='accounts[AD].passwordExpired'>
   <Display class='Checkbox'>
      <Property name='title 'value='Password is Expired'/>
   </Display>
</Field>
```

DatePicker

ユーザーが、カレンダが表示されるポップアップウィンドウを使用して日付を指定できるようにします。Identity Manager のフォームには、カレンダアイコンとしてフィールドが表示されます。ユーザーがアイコンをクリックすると、Identity Manager に別のポップアップウィンドウとしてカレンダが開きます。

このコンポーネントを使ってユーザーは日付値を入力できます。コンポーネントプロパティーの設定に応じて、ユーザーは選択メニュー、テキストフィールド、またはカレンダポップアップウィンドウを使用して日付値を入力できます。デフォルトでは、テキストフィールドとアイコンが表示され、アイコンをクリックするとカレンダポップアップが表示されます。

次のプロパティーがあります。

- `command` - 送信するオプション値 (たとえば、Save、Cancel、または Recalculate) を指定します。このプロパティーを Recalculate に設定することは、`action` プロパティーを `true` に設定することと同じで、更新またはその他の処理が呼び出されます。更新処理は次のいずれかの方法で呼び出すことができます。

  - 日付ウィジェットで日付を選択する

    または

  - テキスト領域で日付を変更し、Tab キーで別の画面領域に移動するか別の画面領域をクリックして、画面を更新する

  詳細については、「基本コンポーネントクラス」の `command` プロパティーの説明を参照してください。

- `disableTextInput` - (ブール型) `true` に設定すると、テキスト入力ボックスの代わりに日付テキスト文字列が表示されます。テキスト入力ボックスが表示されないのでユーザーはこのフィールドを編集できません。ユーザーが日付文字列の値を変更するには、カレンダアイコンをクリックして、ポップアップウィンドウで日付を選択する必要があります。Identity Manager には、新しく選択した日付がカレンダアイコンの横にプレーンテキストとして表示されます。

  このプロパティーが表示されない場合、または `false` に設定されている場合は、入力テキストフィールドが通常どおりに表示されます。

- `displayFormatHint` - そのテキストフィールドに入力する日付フォーマットのヒントを表示するかどうかを決定します。`true` に設定すると、対応する日付フォーマットのヒントが Identity Manager に表示されます。フォーマット文字列の値は、コンポーネントの `format` プロパティーによって決まります。次のようなときは、Identity Manager にヒントは表示されません。

  - このプロパティーが `false` に設定されている
  - このプロパティーが表示されない
  - `multiField` プロパティーが `true` に設定されている

- disableTextInput が true である

- format - 日付を表示するための日付フォーマットを指定します。値には、y、M、または d の任意のフォーマット文字を使用する、Java 形式の日付フォーマット文字列を指定できます。また、ISO フォーマット (yyyy-MM-dd) を指定する値 iso、またはロケールを反映したフォーマット (ロケール用の Java デフォルト) を指定する値 local も指定できます。指定しない場合は、Identity Manager では「MM/dd/yyyy」フォーマットが使用されます。
- multiField - 日付要素ごとに、独立した入力フィールドを表示するかどうかを指定します。値を設定しなかった場合、または false に設定した場合は、Identity Manager では適切にフォーマットされた日付テキストを入力するための 1 つのテキストフィールドが使用されます。
- value - カレンダで、現在の日付として強調表示する日付を指定します。日付は、Date オブジェクトまたは String オブジェクトのいずれかから解析できます。

例

```
<Field name='ExpireDate'>
   <Display class='DatePicker'>
      <Property name='title' value='Set Password Expire date'/>
      <Property name='format'= value='iso'/>
   </Display>
</Field>
```

FileUpload

ユーザーがファイルを選択し、それをサーバーにアップロードするためのテキストフィールドと「**参照**」ボタンを表示します。ファイル (ユーザーオブジェクトや設定オブジェクトなど) から Identity Manager にデータをインポートするときは、このコンポーネントを使用します。このコンポーネントは、Text コンポーネントがサポートするすべてのプロパティーをサポートします。

Html

このコンポーネントを使用することで、HTML ページに格納されているフォームフィールドやその他のコンポーネントに、JavaScript など任意の HTML マークアップを挿入できます。

このコンポーネントには、html という 1 つのプロパティーがあります。これは、ページに表示する文字列の指定に使用されます。

例

```
<Display class='Html'>
   <Property name='html'>
       <concat>
         <s><![CDATA[<div class="DashAlrtMsgTxt">]]></s>
         <ref>loginWarning</ref>
         <s><![CDATA[ <a href=']]></s>
         <s>user/changePassword.jsp</s>
         <s><![CDATA['>]]></s>
         <message name='UI_USER_MAIN_CLICK_HERE_INTRO'/>
         <s><![CDATA[</a>]]></s>
         <message name='UI_USER_MAIN_CLICK_HERE_REMAINDER'/>
         <s><![CDATA[</div>]]></s>
       </concat>
   </Property>
</Display>
```

HtmlPage

ここでは、ルート HTML ページについて説明します。このコンポーネントには、任意の HTML とブラウザ JavaScript を含めることができます。次のプロパティーがあります。

- `commentScripts` - JavaScript 用に出力される `<script>` タグを、コメント内に入れるかどうかを指定します。
- `title` - ページのタイトルを指定します。String または Message のデータ型でも指定できますが、通常は String として指定されます。
- `postUrl` - メインフォームの送信時に Identity Manager が送信する URL を指定します。
- `messages` - 表示する情報メッセージを指定します。
- `comments` - ページに含める特別なコメントを指定します。このプロパティーは、通常、GenericEditForm および FormConverter メソッドで例外が検出された場合に使用されます。
- `focussedFieldName` - 最初に選択されるフィールドの名前を指定します。通常は NULL に設定されます。このプロパティーの値は、最初のテキストフィールドとして、またはテキストフィールドが存在しない場合に最初のフィールドとして計算されます。
- `activeControl` - 最後にアクティブであると認識されるフォームフィールドの名前を指定します (String)。

InlineAlert

エラー、警告、成功、または情報のアラートボックスを表示します。このコンポーネントは、通常はページの上部に配置されます。タイプが InlineAlert$AlertItem の子コンポーネントを定義することで、1 つのアラートボックスに複数のアラートを表示できます。

この表示コンポーネントには、次のプロパティーがあります。

- alertType - 表示するアラートの種類を指定します。このプロパティーは、適用するスタイルと使用する画像を決定します。指定できる値は、error (エラー)、warning (警告)、success (成功)、および info (情報) です。このプロパティーのデフォルト値は info です。このプロパティーは、InlineAlert でのみ有効です。
- header - アラートボックスに表示するタイトルを指定します。これは、文字列またはメッセージのいずれかのオブジェクトです。このプロパティーは、InlineAlert または InlineAlert$AlertItem で有効です。
- value - 表示するアラートメッセージを指定します。この値は、文字列またはメッセージのいずれかのオブジェクトです。このプロパティーは、InlineAlert または InlineAlert$AlertItem で有効です。
- linkURL - アラートの下部に表示するオプション URL を指定します。このプロパティーは、InlineAlert または InlineAlert$AlertItem で有効です。
- linkText - linkURL のテキストを指定します。これは、文字列またはメッセージのいずれかのオブジェクトです。このプロパティーは、InlineAlert または InlineAlert$AlertItem で有効です。
- linkTitle - linkURL のタイトルを指定します。これは、文字列またはメッセージのいずれかのオブジェクトです。このプロパティーは、InlineAlert または InlineAlert$AlertItem で有効です。

単一のアラートメッセージの例

```
<Field>
   <Display class='InlineAlert'>
      <Property name='alertType' value='warning'/>
      <Property name='header' value='Data not Saved'/>
      <Property name='value' value='The data entered is not yet saved.
Please click Save to save the information.'/>
   </Display>
</Field>
```

複数のアラートメッセージの例

alertType は、InlineAlert プロパティー内のみに定義します。その他のプロパティーは InlineAlert$AlertItems に定義できます。

```
<Field>
   <Display class='InlineAlert'>
      <Property name='alertType' value='error'/>
   </Display>
   <Field>
     <Display class='InlineAlert$AlertItem'>
        <Property name='header' value='Server Unreachable'/>
        <Property name='value' value='The specified server could not
       be contacted.Please view the logs for more information.'/>

        <Property name='linkURL' value='viewLogs.jsp'/>
        <Property name='linkText' value='View logs'/>
        <Property name='linkTitle' value='Open a new window with
           the server logs'/>

     </Display>
  </Field>
  <Field>
     <Display class='InlineAlert$AlertItem'>
        <Property name='header' value='Invalid IP Address'/>
        <Property name='value' value='The IP address entered is in an
invalid subnet. Please use the 192.168.0.x subnet.'/>
      </Display>
  </Field>
</Field>
```

Javascript

ページに事前にフォーマットされている JavaScript を挿入するときに使用されます。これは、コンポーネントで onClick または onChange プロパティーを使用し、カスタム JavaScript 関数を呼び出す場合に便利です。

name プロパティーの設定は必須ではありませんが、XML フォームからコンポーネントを作成する場合は、設定するようにしてください。フィールドをループさせたり、含めたりする機能などを使用するときに、同じスクリプトを含む複数の JavaScript コンポーネントをページに追加できます。同じ名前の JavaScript コンポーネントは、HTML の生成時に 1 回だけ取り込まれます。

例

```
<Display class='Javascript'>
 <Property name='script'>
 <String>
function setTextFromSelect(sel, textFieldName) {
 if ( sel == null || sel.inchange ) return;
 sel.inchange = true;
 var textField = sel.form.elements[textFieldName];
 if ( textField == null ) return;
 textField.value = sel.value;
 sel.selectedIndex = 0;
 sel.inchange = false;
}  // setTextFromSelect(sel, textFieldName)
 </String>
 </Property>
 <Property name='noNewRow' value='true'/>
 </Display>
```

コンポーネントには、JavaScript テキストを格納できる `script` という拡張プロパティーがあります。

Label

テキストの文字列を表示します。

この表示コンポーネントには、次のプロパティーがあります。

- `value` - 表示するテキストを定義します。この値は、文字列、または文字列のリストのいずれかです。値がリストの場合は、リストに含まれる各文字列が別々の行に表示されます。
- `leftPad` - ラベルの左に挿入する空白文字の数を指定します。
- `pad` - ラベルの左右に挿入する空白文字の数を指定します。
- `rightPad` - ラベルの右に挿入する空白文字の数を指定します。

注 空白文字の数を指定しない場合は、デフォルト設定の leftPad=2、rightPad=2 が適用されます。

```
<Field>
   <Display class='Label'>
      <Property name='title' value='Account ID'/>
      <Property name='value'>
```

```
          <ref>waveset.accountId,/ref>
        </Property>
    </Display>
</Field>
```

- font - フォントスタイルを指定します。指定できる値は、Identity Manager のインストールディレクトリ内の styles/style.css ファイルに定義されているいずれかのスタイル名です。
- color - ラベルの色を指定します。色の値の指定には、HTML の標準色フォーマット (#*xxxxxx*) を使用します。

Link

ページにリンクを配置します。

次のプロパティーがあります。

- URL - ターゲット URL (Uniform Resource Locator) を指定します。
- imageURL - (省略可能) リンクの右に表示されるアイコンまたは画像に URL を指定します。
- imageURL2 - (省略可能) 最初の画像の右に表示されるアイコンまたは画像に URL を指定します。
- hoverText - マウスカーソルを最初の画像または 2 番目の画像に合わせたときに表示されるテキストを指定します。
- id - (省略可能) リンクの *id* クエリー引数として含められる値を指定します。
- arguments - (省略可能) クエリー引数として含められる名前と値のペアのセットを指定します。
- extraURL - (省略可能) ベース URL と引数の後に含められる、追加の URL フラグメントを指定します。
- baseURLOption - (省略可能) 生成される URL のプレフィックスを指定します。異なるベース URL が必要な場合は、この設定は baseURL RequestState 設定よりも優先されます。

例

```
<Field>
   <Display class='Link'>
      <Property name='name' value='Request
            Group Access'/>
      <Property name='URL'
         value='user/processLaunch.jsp?newView=true'>
      <Property name='id' value='Group Request
            Process'/>
   </Display>
</Field>
```

注 Link コンポーネントは、<map> 要素を使用して名前と値のペアを渡すことができる、フォーム内の唯一の場所です。次の例の <map> 要素には、String をブール型の値に、また、String を List にマッピングするためのペアがいくつか含まれています。

```
<invoke class='com.waveset.ui.FormUtil'
name='getOrganizationsDisplayNames'>
   <ref>:display.session</ref>
     <map>
       <s>filterVirtual</s>
       <o><Boolean>true</Boolean></o>
       <s>current</s>
       <list>
         <ref>original.orgParentName</ref>
       </list>
       <s>excluded</s>
       <list><ref>orgName</ref></list>
     </map>
</invoke>
```

LinkForm

メニューと同様に、リンクの箇条書きリストを表示します。

ListEditor

編集できる文字列のリストを表示します。

プロパティー

次のプロパティーがあります。

- `listTitle` - (String) Identity Manager が ListEditor グラフィカル表示の横に表示するラベルを指定します。
- `pickListTitle` - (String) `picklist` コンポーネントに使用するラベルを指定します。
- `valueMap` - (Map) リスト内の値の表示ラベルのマップを指定します。
- `allowDuplicates` - (Boolean) 値に `true` を指定すると、Identity Manager が管理するリストで重複が許可されます。
- `allowTextEntry` - (Boolean) 値に `true` を指定すると、Identity Manager にテキスト入力ボックスと追加ボタンが表示されます。
- `fixedWidth` - (Boolean) 値に `true` を指定すると、固定幅として表示されるはずです (`Multiselect` コンポーネントと同じ動作)。
- `ordered` - (Boolean) 値に `true` を指定すると、値の順序が保持されます。
- `sorted` - (Boolean) 値に `true` を指定すると、選択リスト内の値がソートされます。値が multivalued であり ordered でない場合は、Identity Manager では値リストもソートします。
- `pickValueMap` - (List または Map) 選択リスト内の値の表示ラベルのマップを指定します。
- `pickValues` - (List) `picklist` コンポーネント内で選択できる値を指定します。NULL の場合は、`picklist` は表示されません。
- `height` - (Integer) 優先する高さを指定します。
- `width` - (Integer) 優先する幅を指定します。この項目が表示されるテーブルセルのプロパティーとしてコンテナで使用できます。

例

次の例では、ListEditor 表示クラスを使用するフィールドが表示されます (Tabbed User Form)。

```
<Field name='accounts[Sim1].Group'>
     <Display class='ListEditor' action='true'>
        <Property name='listTitle' value='stuff'/>
        <Property name='allowTextEntry'>
            <Boolean>true</Boolean>
        </Property>
        <Property name='ordered'>
            <Boolean>true</Boolean>
        </Property>
     </Display>
     <Expansion>
         <ref>accounts[Sim1].Group</ref>
     </Expansion>
</Field>
```

このコードスニペットでは、顧客がユーザーにグループを追加したりユーザーからグループを削除したりできるフィールドが作成されます。

注 この表示クラスには通常、文字列のリストが入力として必要になります。1 つの文字列を文字列のリストにするときは、次のようにします。

```
<Expansion>
   <appendAll><ref>accounts[Sim1].Group</ref></appendAll>
</Expansion>
```

NameValueTable

このコンポーネントは、名前と値のペアを 2 列の簡易テーブルに表示するときに使用されます。このコンポーネントは、含まれるデータを直接表示します。

データはいくつかの形式で指定できます。

- フラットリスト - リストには、要素 0 が名前、要素 1 が値、要素 2 が名前のような形式で、名前と値のペアが含まれます。
- マップ - マップ内のエントリは、アルファベット順に出力されます。
- GenericObject - オブジェクトはフラット化され、マップとして出力されます。

プロパティーには、`true` に設定した場合に値を含まない行を非表示にする `_hideEmptyRows` があります。

MultiSelect

複数の選択項目を持つテキストボックスを表示します。これは、1 つのボックスに含まれる定義済みの値セットを選択ボックスに移動できる、2 つの部分から構成されるオブジェクトです。左のボックスの値は `allowedValues` プロパティーで指定します。多くの場合、この値は `FormUtil.getResources` などの Java メソッドを呼び出すことで動的に取得されます。複数選択ボックスの右側部分に表示される値には、フィールド名によって特定される関連ビュー属性の現在値が適用されます。

この 2 部構成のオブジェクトのフォームタイトルは、`availabletitle` プロパティーと `selectedtitle` プロパティーによって設定されます。

`MultiSelect` コンポーネントがアプレットを使用しないように設定するときは、`noApplet` プロパティーを **true** に設定します。

関連する説明については、459 ページの「MultiSelect コンポーネントを使用しない方法」を参照してください。

注 Safari ブラウザが稼動するシステムで Identity Manager を実行する場合は、MultiSelect コンポーネントを使用するすべてのフォームで、`noApplet` オプションの設定をカスタマイズしてください。このオプションを次のように設定します。

```
<Display class='MultiSelect'>
      <Property name='noApplet' value='true'/>
 ...
```

この表示コンポーネントには、次のプロパティーがあります。

- `allowedValues` - 複数選択ボックスの左側に関連付ける値を指定します。この値は、必ず文字列のリストとして指定してください。**注** : このボックスへの値の取り込みには `<Constraints>` 要素を使用できますが、この方法はお勧めしません。
- `availableTitle` - 表示されるボックスのタイトルを指定します。
- `class` - コンポーネントをアプレットとして表示しないときに、`MultiSelect` ボタンのスタイルを設定するために使用する CSS クラスを指定します。デフォルトは `formbutton` です。
- `disabledclass` - コンポーネントをアプレットとして表示しないときに、無効になっている `MultiSelect` ボタンのスタイルを設定するために使用する CSS クラスを指定します。デフォルトは `formbutton` です。
- `displayCase` - 各 `allowedValues` を大文字または小文字の値にマップします。**upper** または **lower** の値を取ります。

- height - 選択しているボックスの高さを、ピクセル単位で指定します。デフォルト値は、400 です。
- noApplet - MultiSelect コンポーネントがアプレットとともに実装されるか、標準 HTML 選択ボックスのペアとともに実装されるかを指定します。デフォルトでは、アプレットを使用するように設定されます。これは、値の長いリストを処理する場合に効果的です。Safari ブラウザが稼動するシステムでのこのオプションの使用については、前述の注記を参照してください。
- onBlur - 複数選択ボタンの onBlur イベントで実行する Javascript。このプロパティーを使用して、ボタンがフォーカスを失ったときのボタンのスタイルを変更できます。
- onFocus - MultiSelect ボタンの onFocus イベントで実行する Javascript を指定します。これを使用して、ボタンにフォーカスがあるときのボタンのスタイルを変更できます。
- onMouseOver - MultiSelect ボタンの onMouseOver イベントで実行する Javascript を指定します。このプロパティーを使用して、ボタンの上にマウスがあるときのボタンのスタイルを変更できます。
- onMouseOut - MultiSelect ボタンの onMouseOut イベントで実行する Javascript を指定します。このプロパティーを使用して、ボタンからマウスが離れたときのボタンのスタイルを変更できます。
- ordered - テキストボックスの項目リスト内で、選択している項目を上下に移動できることを定義します。値を true に設定すると、選択している項目を上下に移動させるための追加ボタンが表示されます。
- selectedTitle - 選択されたボックスのタイトルを指定します。
- sorted - 両方のボックスの値をアルファベット順にソートすることを指定します。
- typeSelectThreshold - noApplet プロパティーが true に設定されている場合にのみ適用されます。このコンポーネントは、allowedValue リストの下に先行入力選択ボックスを表示するかどうかを制御します。左の選択ボックスのエントリの数が、このプロパティーによって定義されるしきい値に達すると、選択ボックスの下に追加のテキスト入力フィールドが表示されます。テキストフィールドに文字を入力するときに選択ボックスがスクロールされ、入力した文字と一致するエントリが存在する場合に、それが表示されます。たとえば、**w** と入力すると、名前が **w** から始まる最初のエントリまで、選択ボックスがスクロールされます。
- width - 選択しているボックスの幅を、ピクセル単位で指定します。デフォルト値は、150 です。

例

```
<Field name='accounts[LDAP].LDAPDept' type='string'>
   <Display class='MultiSelect' action='true'>
      <Property name='title' value='LDAP Department'/>
   </Display>
   <Constraints>
      <o>
         <List>
             <String>Sales</String>
             <String>Marketing</String>
             <String>International Sales</String>
      </List>
      </o>
   </Constraints>
</Field>
```

Radio

1 つまたは複数のラジオボタンの水平リストを表示します。ユーザーが一度に選択できるラジオボタンは 1 つだけです。コンポーネントの値が NULL の場合、または許可される値のいずれとも一致しない場合は、ボタンは選択されません。

この表示コンポーネントには、次のプロパティーがあります。

- `title` - すべてのラジオボタンのタイトルを指定します。
- `labels` - ボタンラベルの代替リストを指定します。`labels` リストの長さは、`allowedValues` リストの値と同じ長さにしてください。代替ラベルは、値が暗号化されている場合に使用されます。たとえば、H、M、S などの文字コードを値として指定できますが、これは時、分、および秒のボタンラベルの識別に使用されるプロパティーです。
- `allowedValues` - 各ボタンに割り当てる値を指定します。この値は、必ず文字列のリストとして指定してください。
- `value` - ボタンの値を指定します。この値は 1 つの文字列を受け付けます。設定しない場合は、ラベルと同じ値が適用されます。

例

```
<Field name='attributes.accountLockExpiry.unit'>
   <Display class='Radio'>
     <Property name='noNewRow' value='true'/>
     <Property name='labels'>
        <List>
        </List>
     </Property>
   </Display>
</Field>
```

```
<Field name='attributes.accountLockExpiry.unit'>
            <String>UI_TASKS_XML_SCHED_MINUTES</String>
            <String>UI_TASKS_XML_SCHED_HOURS</String>
            <String>UI_TASKS_XML_SCHED_DAYS</String>
            <String>UI_TASKS_XML_SCHED_WEEKS</String>
            <String>UI_TASKS_XML_SCHED_MONTHS</String>
         </List>
      </Property>
      <Property name='allowedValues'>
         <List>
            <String>minutes</String>
            <String>hours</String>
            <String>days</String>
            <String>weeks</String>
            <String>months</String>
         </List>
      </Property>
   </Display>
</Field>
```

SectionHead

text プロパティーの値によって定義される、新しいセクション見出しを表示します。これは Label クラスを拡張したもので、font プロパティーに大きな太字のテキストとなるスタイルを設定します。また、pad プロパティーはゼロに設定され、デフォルトの 2 文字分の隙間は取り除かれます。長いフォームを、目立つラベルで複数のセクションに分割する場合に使用されます。

この表示コンポーネントのプロパティーは、表示するテキストを指定する text のみです。

例

```
<Field>
   <Display class='SectionHead'>
      <Property name='text' value ='Calculated Fields'/>
   </Display>
</Field>
```

Select

1 つの項目を選択できるリストボックスを表示します。このリストボックスの値は、allowedValues プロパティーで指定します。

この表示コンポーネントには、次のプロパティーがあります。

- allowedValues - リストボックスに表示される、選択できる値のリストを表示します。
- allowedOthers - これを設定すると、allowedValues リストに指定されていない初期値が許容され、リストに追加されます。
- autoSelect - true に設定すると、フィールドの初期値が NULL の場合に、allowedValues リストの最初の値が自動的に選択されます。
- multiple - true に設定すると、複数の値を選択できるようになります。
- nullLabel - どの値も選択していない場合に、リストボックスの最上部に表示するテキストを指定します。
- optionGroupMap - セレクタが、<optgroup> タグを使用して、グループ内のオプションを表示できるようになります。マップのキーがグループラベルとなり、要素が選択可能なリスト項目になるように、マップをフォーマットしてください。値を表示するには、それらの値が allowedValues のメンバーである必要があります。
- size - (省略可能) 表示する行の最大数を指定します。行数がこの値を超える場合は、スクロールバーが追加表示されます。
- sorted - true に設定すると、リストの値がソートされます。
- valueMap - raw 値と表示値をマップします。

このコンポーネントは、command プロパティーと onChange プロパティーをサポートします。

例

```
<Field name='city' type='string'>
   <Display class='Select'>
      <Property name='title' value='City'/>
      <Property name='allowedValues'>
         <List>
            <String>Austin</String>
            <String>Portland</String>
            <String>New York</String>
         </List>
      </Property>
   </Display>
</Field>
```

Text

通常のテキスト入力ボックスを表示します。

この表示コンポーネントでよく使用されるプロパティーは次のとおりです。

- autocomplete ユーザーのクレデンシャルをコンピュータに保存するべきかどうかを指定します。デフォルトでは、このプロパティーは off に設定されているので、この情報が保存されることはありません。
- size - テキスト入力ボックスに表示される文字数を指定します。入力ボックスのサイズは、ボックス内のテキストの長さに応じて再計算されます。
- notrim - HTML フォームから送信されるテキストをトリミングするかどうかを指定します。true に設定すると、空白文字が削除されません。空白文字を維持するには、このオプションを設定します。
- noTranslate - true に設定すると、メッセージキーの値が代替値としてではなく、そのまま表示されます。デフォルトは false です。
- maxLength - テキストボックスで編集できる文字列の最大長を指定します。
- multiValued - true に設定すると、値を追加、削除できるように、「追加」ボタンと「削除」ボタンを持つテキストボックスが表示されます。
- secret - 入力したテキストの代わりに、アスタリスク (*****) が表示されます。このオプションは、パスワードフィールドで頻繁に利用されます。
- readOnly - 読み取り専用のテキストを表示します。ユーザーがこのテキストを編集することはできません。このプロパティーは、たとえば、管理者がユーザーアカウントを作成または編集するときに、リソース属性に関する必要情報を表示できるようにする場合などに使用されます。
- submitOnEnter - このプロパティーが設定され、Text フィールドが選択されると、ユーザーが Enter キーを押したときに、プロパティー値として指定されているコマンドを使用してフォームが送信されます。次の例では、ユーザーが「**保存**」ボタンをクリックした場合と同様に、フォームが送信されます。

例

```
<Field name='variables.identityID'>
   <Display class='Text'>
      <Property name='required'>
         <Boolean>true</Boolean>
      </Property>
      <Property name='title' value='Identity ID'/>
      <Property name='size' value='32'/>
      <Property name='maxLength' value='128'/>
      <Property name='submitOnEnter' value='Save'/>
   </Display>
</Field>
```

TextArea

複数行のテキスト入力ボックスを表示します。

この表示コンポーネントには、次のプロパティーがあります。

- rows - テキスト領域の行数を指定します (Integer)。
- columns - テキスト領域の列数を指定します (Integer)。
- readOnly - テキスト入力ボックスに、読み取り専用テキストを表示します。true に設定すると、このコンポーネントの境界線は表示されなくなります (ブール型)。
- format - setValue() の動作を制御し、getPostData() から返されるオブジェクトのタイプを決定します (String)。
- sorted - true に設定すると、テキスト領域の行をソートできるようになります。この機能は、自由形式のテキストではなく、選択項目のリストをテキスト領域に表示する場合に便利です (ブール型)。
- noTrim - HTML フォームから送信されるテキストをトリミングするかどうかを指定します。デフォルトの設定では、空白文字は削除されます。空白文字を維持するには、この値を true に設定します。

例

5 つの表示行を持ち、70 文字で行を折り返すテキストボックスを表示するには、次のように設定します。

```
<Field name='Description'>
   <Display class='TextArea'>
     <Property name='rows' value='5'/>
     <Property name='columns' value='70'/>
   </Display>
</Field>
```

定義されている表示行を超えてテキストを入力すると、テキスト領域にスクロールバーが追加表示されます。

MultiSelect コンポーネントを使用しない方法

MultiSelect コンポーネント (アプレット版または HTML 版) を使用して多くの管理者ロールを表示すると、使いにくくなることがあります。Identity Manager には、管理者ロールをより効率的に表示して管理できるように、objectSelector フィールドテンプレートが用意されています。

Scalable Selection Library (sample/formlib.xml 内) には、objectSelector フィールドテンプレートを使用してユーザーが選択できる管理者ロール名を検索するサンプルが入っています。

コード例 6-2　objectSelector フィールドテンプレートのサンプル

```
<Field name='scalableWaveset.adminRoles'>
  <FieldRef name='objectSelector'>
     <Property name='selectorTitle' value='_FM_ADMIN_ROLES'/>
     <Property name='selectorFieldName' value='waveset.adminRoles'/>
     <Property name='selectorObjectType' value='AdminRole'/>
     <Property name='selectorMultiValued' value='true'/>
     <Property name='selectorAllowManualEntry' value='true'/>
     <Property name='selectorFixedConditions'>
       <appendAll>
         <new class='com.waveset.object.AttributeCondition'>
           <s>hidden</s>
           <s>notEquals</s>
           <s>true</s>
         </new>
         <map>
           <s>onlyAssignedToCurrentSubject</s>
           <Boolean>true</Boolean>
         </map>
       </appendAll>
     </Property>
     <Property name='selectorFixedInclusions'>
       <appendAll>
         <ref>waveset.original.adminRoles</ref>
       </appendAll>
     </Property>
  </FieldRef>
</Field>
```

objectSelector サンプルコードを使用する方法

1. Identity Manager IDE から Administrator Library UserForm オブジェクトを開きます。
2. このフォームに次のコードを追加します。

```
<Include>
   <ObjectRef type='UserForm' name='Scalable Selection Library'/>
</Include>
```

3. `AdministratorFields` フィールド内で `accounts[Lighthouse].adminRoles` フィールドを選択します。
4. `accounts[Lighthouse].adminRoles` 全体を次の参照で置き換えます。

```
<FieldRef name='scalableWaveset.adminRoles'/>
```

5. オブジェクトを保存します。

次回以降にユーザーを編集して「セキュリティー」タブを表示すると、Identity Manager にカスタマイズされたフォームが表示されます。「...」をクリックすると Selector コンポーネントが開き、検索フィールドが表示されます。このフィールドを使用してあるテキスト文字列で始まる管理者ロールを検索すると、フィールドの値に 1 つ以上の値が設定されます。

フォームを復元するときは、「**設定**」>「**交換ファイルのインポート**」から `$WSHOME/sample/formlib.xml` をインポートします。

多数のオブジェクトを使用する環境でリソースとロールを管理するために `objectSelector` テンプレートを使用するその他のサンプルについては、`sample/formlib.xml` 内の Scalable Selection Library を参照してください。

フォームとプロセスのマッピング

この付録は、Identity Manager で使用されるフォームおよびワークフロープロセスと、それに対応するシステム名の一覧を示しています。

フォームのマッピング

次の表は、各フォームのシステム名と、製品インタフェースに表示される名前を示しています。

「**システム名**」列には、フォームのシステム名が示されます。

「**マップ対象**」列には、Identity Manager IDE でのフォームの識別に使用され、Identity Manager の「デバッグ」ページに表示される名前が示されます。

表 A-1 フォームのシステム名と製品インタフェース名

システム名	マップ対象
accessApprovalList	アクセス承認リスト
accessReviewAbortConfirmation	アクセスレビュー終了確認
accessReviewDeleteConfirmation	アクセスレビュー削除確認フォーム
accessReviewDashboard	アクセスレビューダッシュボード
accessReviewSummary	アクセスレビュー概要
accessReviewDetail	アクセスレビュー詳細
accessScanDeleteConfirmation	アクセススキャン削除確認
accessScanForm	アクセススキャンフォーム
accessScanList	アクセススキャンリスト
accountOwnerSelection	アカウント所有者選択フォーム

表 A-1　フォームのシステム名と製品インタフェース名 (続き)

システム名	マップ対象
accountSelect	アカウント選択フォーム
activeSyncWizard	リソース Active Sync ウィザード
anonymousUserMenu	匿名ユーザーメニュー
auditPolicyDeleteConfirmation	監査ポリシー削除確認フォーム
auditPolicyList	監査ポリシーリスト
auditorViewUserComplianceForm	Auditor タブ
changeAnswers	ユーザーの秘密質問の回答変更フォーム
changeCapabilities	ユーザー機能変更フォーム
changeMyPassword	マイパスワード変更フォーム
changeOrgAuditPolicies	組織監査ポリシー変更フォーム
changePassword	ユーザーパスワード変更フォーム
changePasswordSelection	ユーザー選択フォーム
changeUserAuditPolicies	ユーザー監査ポリシー変更フォーム
complianceViolationSummaryForm	コンプライアンス違反概要フォーム
conditionForm	条件ダイアログ
confirmDeletes	削除確認
conflictViolationDetailsForm	相反違反詳細フォーム
complianceViolationSummaryForm	コンプライアンス違反概要フォーム
LDAP ChangeLog ActiveSync グループ作成フォーム	LDAP グループ作成フォーム
LDAP ChangeLog ActiveSync 組織作成フォーム	LDAP 組織作成フォーム
LDAP ChangeLog ActiveSync 組織単位作成フォーム	LDAP 組織単位作成フォーム
LDAP ChangeLog ActiveSync 人物作成フォーム	LDAP 人物作成フォーム
LDAP ChangeLog ActiveSync グループ更新フォーム	LDAP グループ更新フォーム
LDAP ChangeLog ActiveSync 組織更新フォーム	LDAP 組織更新フォーム
LDAP ChangeLog ActiveSync 組織単位更新フォーム	LDAP 組織単位更新フォーム
LDAP ChangeLog ActiveSync 人物更新フォーム	LDAP 人物更新フォーム
LDAP Listener ActiveSync グループ作成フォーム	LDAP グループ作成フォーム
LDAP Listener ActiveSync 組織作成フォーム	LDAP 組織作成フォーム

表 A-1　フォームのシステム名と製品インタフェース名 (続き)

システム名	マップ対象
LDAP Listener ActiveSync 組織単位作成フォーム	LDAP 組織単位作成フォーム
LDAP Listener ActiveSync 人物作成フォーム	LDAP 人物作成フォーム
LDAP Listener ActiveSync グループ更新フォーム	LDAP グループ更新フォーム
LDAP Listener ActiveSync 組織更新フォーム	LDAP 組織更新フォーム
LDAP Listener ActiveSync 組織単位更新フォーム	LDAP 組織単位更新フォーム
LDAP Listener ActiveSync 人物更新フォーム	LDAP 人物更新フォーム
NetWare NDS ActiveSync グループ作成フォーム	NetWare NDS グループ作成フォーム
NetWare NDS ActiveSync 組織作成フォーム	NetWare NDS 組織作成フォーム
NetWare NDS ActiveSync 組織単位作成フォーム	NetWare NDS 組織単位作成フォーム
NetWare NDS ActiveSync ユーザー作成フォーム	NetWare NDS ユーザー作成フォーム
NetWare NDS ActiveSync グループ更新フォーム	NetWare NDS グループ更新フォーム
NetWare NDS ActiveSync 組織更新フォーム	LDAP 組織更新フォーム
NetWare NDS ActiveSync 組織単位更新フォーム	NetWare NDS 組織単位更新フォーム
NetWare NDS ActiveSync ユーザー更新フォーム	NetWare NDS ユーザー更新フォーム
remediationList	是正リスト
UserEntitlementForm	userEntitlementForm
userEntitlementSummaryForm	ユーザーエンタイトルメント概要フォーム
violationDetailForm	違反詳細フォーム
Windows Active Directory ActiveSync コンテナ作成フォーム	Windows Active Directory コンテナ作成フォーム
Windows Active Directory ActiveSync グループ作成フォーム	Windows Active Directory グループ作成フォーム
Windows Active Directory ActiveSync 組織単位作成フォーム	Windows Active Directory 組織単位作成フォーム
Windows Active Directory ActiveSync ユーザー作成フォーム	Windows Active Directory ユーザー作成フォーム
Windows Active Directory ActiveSync コンテナ更新フォーム	Windows Active Directory コンテナ更新フォーム
Windows Active Directory ActiveSync グループ更新フォーム	Windows Active Directory グループ更新フォーム

表 A-1　フォームのシステム名と製品インタフェース名 (続き)

システム名	マップ対象
Windows Active Directory ActiveSync 組織単位更新フォーム	Windows Active Directory 組織単位更新フォーム
Windows Active Directory ActiveSync ユーザー更新フォーム	Windows Active Directory ユーザー更新フォーム
accountOwnerSelection	アカウント所有者選択フォーム
anonymousUserMenu	匿名ユーザーメニュー
changeAnswers	ユーザーの秘密質問の回答変更フォーム
changeCapabilities	ユーザー機能変更フォーム
changeMyPassword	マイパスワード変更フォーム
changePassword	ユーザーパスワード変更フォーム
changePasswordSelection	ユーザー選択フォーム
confirmDeletes	削除確認
deprovisionUser	プロビジョニング解除フォーム
disableUser	無効化フォーム
editArgument	引数編集
editChangeLog	ChangeLog 編集
editChangeLogConfiguration	ChangeLog 設定編集
editChangeLogPolicy	ChangeLog ポリシー編集
editField	フィールド編集
editForm	フォーム編集
editRule	規則編集
enableUser	有効化フォーム
endUserAccessApprovalList	アクセス承認リスト
endUserAnonymousEnrollment	エンドユーザー匿名登録フォーム
endUserAppMenu	エンドユーザーナビゲーション
endUserChangePassword	パスワード変更フォーム
endUserForm	エンドユーザーフォーム
endUserLaunchList	エンドユーザー起動リスト
endUserMenu	エンドユーザーメニュー

表 A-1　フォームのシステム名と製品インタフェース名 (続き)

システム名	マップ対象
endUserOtherWorkItemList	エンドユーザーその他の作業項目リスト
endUserResetPassword	ユーザーパスワードリセットフォーム
endUserTaskList	エンドユーザータスクリスト
endUserTaskResults	エンドユーザータスク結果
endUserWorkItemEdit	エンドユーザー作業項目編集
endUserWorkItemList	エンドユーザー作業項目リスト
endUserWorkItemListExt	エンドユーザー承認リスト
findAccountOwner	アカウント所有者検索フォーム
findObjects	オブジェクト検索フォーム
findReconciledAccount	承認検索フォーム
findReconciledAccountResults	アカウント結果検索フォーム
findUser	ユーザー検索フォーム
findUserResults	ユーザー結果検索フォーム
listForms	リストフォーム
listRules	リスト規則
loadForm	デフォルトユーザーフォーム
loginChangeAnswers	ユーザーの秘密質問の回答変更ログインフォーム
loginChangePassword	有効期限切れログインフォーム
loginResetPassword	ユーザーパスワードリセットフォーム
lookupUserId	ユーザー ID の問い合わせ
otherWorkItemList	その他の作業項目リスト
renameUser	ユーザー名変更フォーム
reprovisionForm	デフォルトユーザーフォーム
resetPassword	ユーザーパスワードリセットフォーム
resetPasswordSelection	ユーザー選択フォーム

表 A-1　フォームのシステム名と製品インタフェース名 (続き)

システム名	マップ対象
selfDiscovery	自己検索
userForm	タブ付きユーザーフォーム
viewUserForm	タブ付きビューユーザーフォーム
enableUser	有効化フォーム
endUserChangePassword	パスワード変更フォーム
endUserForm	エンドユーザーフォーム
workItemList	作業項目リスト

プロセスのマッピング

「**システム名**」列には、プロセスのシステム名が示されます。

「**マップ対象**」列には、Identity Manager IDE でのプロセスの識別に使用され、Identity Manager の「デバッグ」ページに表示される名前が示されます。

表 A-2　プロセスのシステム名と製品インタフェース名

システム名	マップ対象
abortAccessReview	アクセスレビューアボート
accessReview	アクセスレビュー
accessReviewScan	アクセススキャン
accessReviewRescan	アクセスレビュー再スキャン
auditPolicyRescan	監査ポリシー再スキャン
changeResourceAccountPassword	アカウントリソースパスワード変更
changeUserPassword	ユーザーパスワード変更
createResourceGroup	グループリソース作成
createResourceObject	オブジェクトリソース作成
createResourceOrganization	組織リソース作成
createResourceOrganizationalUnit	組織単位リソース作成
createResourcePerson	人物リソース作成
createResourceUser	ユーザーリソース作成

表 A-2　プロセスのシステム名と製品インタフェース名 (続き)

システム名	マップ対象
createUser	ユーザー作成
delegateWorkItems	作業項目委任
deleteAccessReview	アクセスレビュー削除
deleteAccount	アカウントリソース削除
deleteResourceGroup	グループリソース削除
deleteResourceObject	オブジェクトリソース削除
deleteResourceOrganization	組織リソース削除
deleteResourceOrganizationalUnit	組織単位リソース削除
deleteResourcePerson	人物リソース削除
deleteResourceUser	ユーザーリソース削除
deleteUser	ユーザー削除
disableUser	ユーザー無効化
enableUser	ユーザー有効化
endUserAnonymousEnrollment	エンドユーザー匿名登録
endUserUpdateGroups	エンドユーザーグループ更新
endUserUpdateMyGroups	エンドユーザー自分のグループ更新
endUserUpdateMyResources	エンドユーザー自分のリソース更新
endUserUpdateMyRoles	エンドユーザー自分のロール更新
endUserUpdateResources	エンドユーザーリソース更新
endUserUpdateRoles	エンドユーザーロール更新
handleNativeChangeToAccountAttributes	アカウント属性へのネイティブ変更監査
lockUser	ユーザーロック
manageResource	リソース管理
manageRole	規則管理
passwordLogin	パスワードログイン
questionLogin	質問ログイン
recoverAccessReview	アクセスレビュー復元
renameUser	ユーザー名変更
resetUserPassword	ユーザーパスワードリセット

表 A-2　プロセスのシステム名と製品インタフェース名 (続き)

システム名	マップ対象
unlinkResourceAccountsFromUser	ユーザーとのアカウントリソースリンク解除
unlockUser	ユーザーロック解除
updateResourceGroup	グループリソース更新
updateResourceObject	オブジェクトリソース更新
updateResourceOrganization	組織リソース更新
updateResourceOrganizationalUnit	組織単位リソース更新
updateResourcePerson	人物リソース更新
updateResourceUser	リソースユーザー更新
updateUser	ユーザーテンプレート更新

注　アクセスレビュータスクはワークフローとして実装されます。それ以外のすべてのタスクは Java タスクとして実装されます。

索引

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W

X

Z

あ

う

え

お

か

き

く

け

こ

さ

し

す

せ

そ

た

ち

て

に

は

ひ

ふ

へ

ほ

ま

む

め

も

ゆ

よ

ら

り

ろ

わ