CA Identity Manager™

Connector Xpress Guide

12.6.4

CA Technologies 製品リファレンス

このマニュアルが参照している CA Technologies の製品は以下のとおりです。

- CA CloudMinder™ Identity Management
- CA ディレクトリ
- CA Identity Manager ™
- CA Identity Governance（旧 CA GovernanceMinder）
- CA SiteMinder®
- CA User Activity Reporting
- CA AuthMinder ™

CA への連絡先

テクニカル サポートの詳細については、弊社テクニカル サポートの Web サイト（http://www.ca.com/jp/support/）をご覧ください。

目次

第 1 章: Connector Xpress

このセクションには、以下のトピックが含まれています。

Connector Xpress の概要

Connector Xpress は、動的コネクタの管理、エンドポイントへの動的コネクタのマッピング、およびエンドポイントのルーティング ルールの確立に使用する CA Identity Manager ユーティリティです。 Connector Xpress を使用すると、動的なコネクタが SQL データベースおよび LDAP ディレクトリのプロビジョニングや管理を行うように設定できます。

Connector Xpress を使用すると、専門知識がなくてもカスタム コネクタを作成および展開できます。

また、Connector Xpress を使用すると、コネクタ サーバ設定をセットアップ、編集、削除することができます。

Connector Xpress への主要入力はエンドポイント システムのネイティブ スキーマです。 たとえば、RDBMS への接続、およびデータベースの SQL スキーマの取得に Connector Xpress を使用できます。 ID 管理とプロビジョニングに関連するネイティブ スキーマの一部からマッピングを構築する場合も Connector Xpress を使用できます。 マッピングには、プロビジョニング レイヤでネイティブ スキーマの要素が表現される方法が記述されます。

Connector Xpress は、動的なコネクタ用に、ターゲット システムへの実行時マッピングを記述するメタデータを生成します。

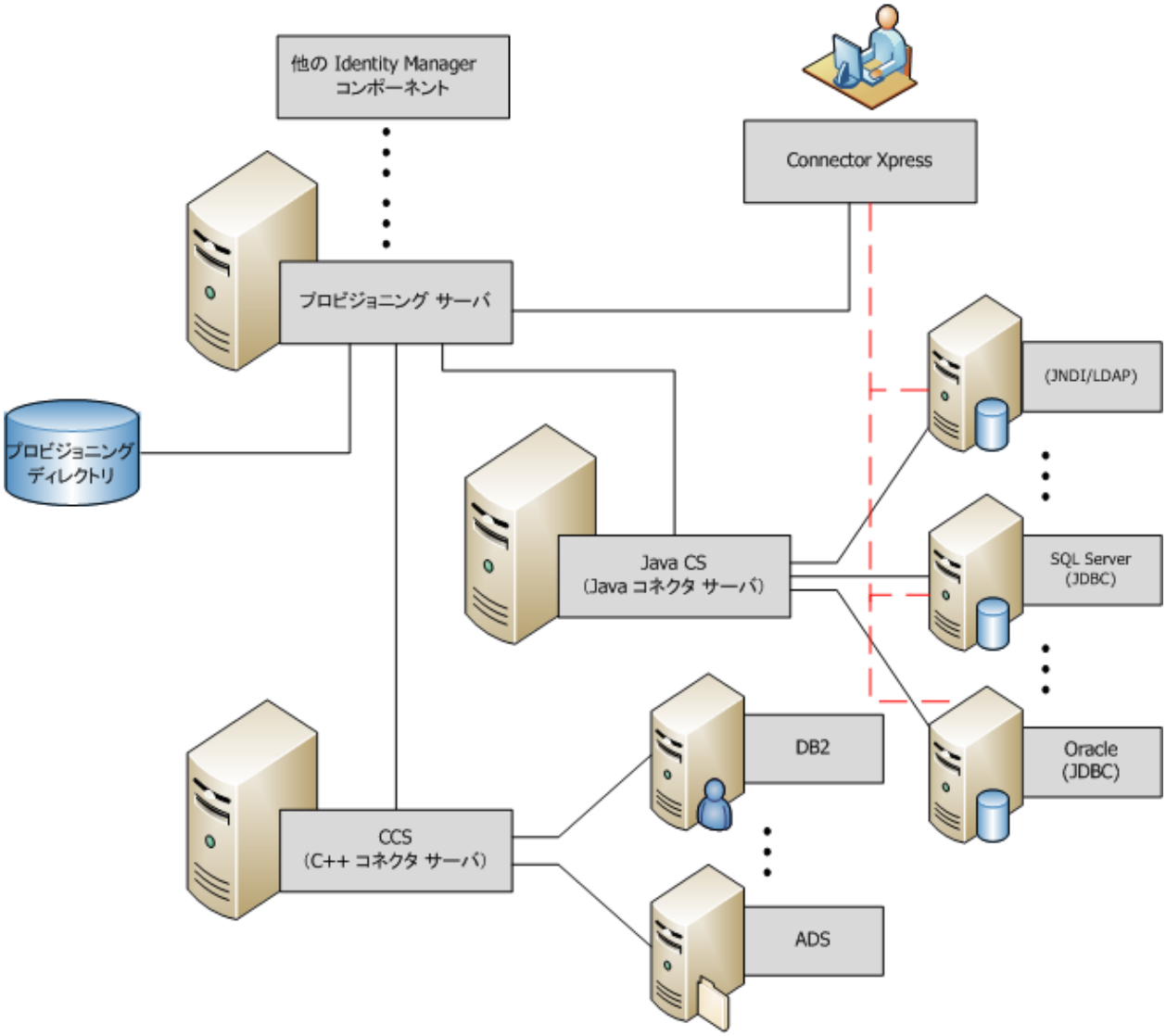
Connector Xpress の出力は、マッピングの完了時に生成されるメタデータ ドキュメントです。 メタデータは、CA IAM CS に対するコネクタの構造を記述する XML ファイルです。

出力結果は、プロビジョニング サーバのクラスと属性、およびそれらがネイティブ スキーマにどのようにマップされるかを記述します。

メタデータは CA Identity Manager で管理できる動的なエンドポイント タイプを作成するために使用されます。

他のコンポーネントとの通信

以下の図では、Connector Xpress と分散システムの他のコンポーネント間の直接通信および間接通信を示します。



注: 赤い破線は、Connector Xpress との直通通信を示し、エンドポイントは読み取り専用であることを示します。データ ソースのセットアップ時に直通通信が発生します。

サポートされているデータベースとディレクトリ

Connector Xpress がサポートしているディレクトリおよびデータベースのリストを参照するには、プラットフォーム サポート マトリックスを参照してください。CERTIFIED CONNECTOR XPRESS ENDPOINT TYPES という名前のテーブルを参照します。

パスワードのキャッシュ

アプリケーションが起動している場合にのみ、Connector Xpress によって、各プロビジョニング サーバと共に使用されるパスワードがキャッシュされます。アプリケーションが起動していないと、パスワードは永続的に記録されません。Connector Xpress でパスワードが必要な場合に、パスワードがキャッシュされていない、または、キャッシュされたパスワードが認証に失敗すると、パスワードを入力するプロンプトが表示されます。

[Connector Xpress]ウィンドウ

［Connector Xpress］ウィンドウは、以下の 3 つのペインから構成されています。

Mapping tree

エンドポイントのマッピングの概要を表示します。 ノードおよび子ノードはそれぞれ、ユーザのエンドポイント マッピングの要素を表します。 たとえば、エンドポイントのマッピングした属性、クラス間の関連付け、および操作バインディングなどです。 ノードをクリックすると、ノードのパラメータを編集できるダイアログ ボックスが表示されます。

Editor panel

選択したノードのパラメータを編集できるダイアログ ボックスが表示されます。

Provisioning Servers tree

複数のプロビジョニング サーバのエンドポイントおよびコネクタを表示します。

Mapping Tree

マッピング ツリーでは、コネクタのマッピング、関連付けおよび操作バインディングを定義できます。 マッピング ツリーには以下のノードがあります。

Endpoint type node

エンドポイントの説明を作成できます。

Classes parent node

新しいクラスをマップしてマッピングしたクラスの読み取り専用リスト、およびそのクラスがマッピングされるネイティブ クラスを表示します。

Class node

クラスをマッピングする JDBC データベース テーブル、または JNDI ネイティブ オブジェクト クラスを指定できます。

Attributes parent node

プロビジョニング属性を JDBC データベース テーブル列またはエンドポイントの JNDI オブジェクト クラス属性にマッピングできます。

Attributes node

選択された属性の詳細情報を設定できます。

Account screens node

CA Identity Manager ユーザ コンソールのアカウント画面のタブおよびページとして表示されるグループおよびサブグループのマッピングした属性を変更できます。

Associations parent node

直接関連付けおよび間接関連付けを作成するクラスを指定できます。

Containers parent node

コネクタでコンテナとして使用するオブジェクト クラスを指定します。

Custom Types

属性マッピングでメタデータ タイプとして使用する任意の文字列（flexi 文字列）を定義できます。

Operation Bindings

操作バインディング情報を指定し、既存の操作バインディングの概要を表示できます。

Scripts parent node

マッピングした操作バインディングにバインドするスクリプトを指定できます。

Scripts node

マッピングした操作バインディングにバインドするスクリプトのパラメータを指定できます。

Provisioning Servers Tree

プロビジョニング サーバ ツリー（左下のペイン）では、複数のプロビジョニング サーバのエンドポイントおよびコネクタを管理できます。プロビジョニング サーバ ツリーには以下のフィールドがあります。

プロビジョニング サーバの選択されたノードをリフレッシュします。

注: ［リフレッシュ］ボタンは、変更できる子ノードのリストがあるツリー ノードを選択した場合にのみ利用できます。 たとえば、ボタンは特定のエンドポイントで利用できません。このようなノードには必ず Policies および Endpoint の子ノードのみがあります。 特定のエンドポイントのエンドポイント ノードでは、ボタンを利用できます。

プロビジョニング サーバで選択されているオブジェクトで実行できるアクションを表示します。このボタンは、プロビジョニング サーバで選択されているオブジェクトを右クリックした場合と同じ効果があります。

プロビジョニング サーバ ノード

プロビジョニング サーバの接続詳細を追加および設定します。プロビジョニング サーバを追加すると、Connector Xpress によってプロビジョニング サーバの詳細が保持されるので、Connector Xpress を実行するたびに詳細にアクセスできます。

サーバ オブジェクト ノード

ネットワーク上に登録されたプロビジョニング サーバをそれぞれ表示します。このノードを使用して、プロビジョニング サーバの接続詳細を編集または削除できます。

Domains Node

プロビジョニング サーバに対して設定されたドメインを表示します。

エンドポイント タイプ ノード

エンドポイント タイプのリストを表示します。このノードを使用して、エンドポイント タイプを作成できます。

エンドポイント オブジェクト ノード

以下を実行できます。

- メタデータの展開および編集
- エンドポイント タイプの削除

- DYN エンドポイント タイプのエンドポイント タイプの消去
- DYN エンドポイント タイプ以外の管理コネクタ サーバの設定

エンドポイント ノード

特定のエンドポイント タイプのエンドポイントをグループ分けします。

エンドポイント名ノード

プロビジョニング サーバ上で取得したエンドポイントを表示します。このノードは、以下の目的で使用します。

- DYN エンドポイントの検索または関連付け
- エンドポイントの削除
- DYN 以外のエンドポイントにのみ管理コネクタ サーバを設定する

ポリシー ノード

特定のエンドポイント タイプのグループのポリシーを表示します。

注: エンドポイント タイプすべてに対してデフォルト ポリシーが用意されています。

CS Config Object ノード

CS 設定のリストを表示します。

このノードは、以下の目的で使用します。

- サーバのブランチ DN ルーティング ルールの既存設定を編集する
- 設定エントリの削除
- 管理コネクタ サーバのパスワードを設定する

[CS Configs]ノード

CA IAM CS 設定の論理グループを表示します。 これらの設定には、ブランチ DN を管理するための DSFConfig スタイルのルーティング ルールが含まれます。 このノードを使用して、C++ コネクタ サーバまたは CA IAM CS サービスの新しいルーティング ルールの設定オブジェクトを追加できます。

Connector Servers Node

選択されたコネクタ サーバによって管理されるエンドポイント タイプ、エンドポイント、および動的エンドポイント タイプが表示されます。

Connector Xpress のユーザ設定格納場所

Java Preferences API を使用して Connector Xpress によってユーザ設定が格納されます。

Windows では、以下のレジストリ キーにデータが格納されます。

HKEY_CURRENT_USER¥Software¥JavaSoft¥Prefs

UNIX では、ユーザの .java/.userPrefs フォルダ下のホーム ディレクトリにデータが格納されます。

注: アンインストーラではこのデータは削除されないので、今後インストールしてもユーザ設定は保持されます。

第 2 章: Connector Xpress のインストール

Connector Xpress によって、動的なコネクタを作成できるようになります。

このセクションには、以下のトピックが含まれています。

インストールの前提条件

Connector Xpress のインストールの前提条件は以下のとおりです。

- CA Identity Manager サーバ
- CA Identity Manager プロビジョニング サーバ

Connector Xpress のインストール

Connector Xpress は、プロビジョニング サーバを含む CA IAM CS または他のサーバ コンポーネントと同じコンピュータにインストールする必要はありません。

重要: CA IAM CS、CA IAM CS SDK および Connector Xpress をインストールする前に、アンチウイルス ソフトウェアをすべて無効にすることをお勧めします。

次の手順に従ってください:

1. ダウンロードした CA Identity Manager インストール ファイルまたは他のメディアを見つけます。
2. 製品ファイル（ZIP または TAR）を抽出します。
3. 以下のサブフォルダに移動し、セットアップ ファイルをダブルクリックします。
   `Provisioning¥ConnectorXpress`
4. 画面の指示に従って、インストールを完了してください。

Connector Xpress のサイレント インストール

Connector Xpress では、サイレント モードのインストールがサポートされます。サイレント インストールを実行するには、応答ファイルを作成する必要があります。

次の手順に従ってください:

1. コマンド ウィンドウで、Connector Xpress ファイルを抽出した場所に移動し、以下のサブフォルダに移動します。

   `Provisioning/ConnectorXpress`

2. 応答ファイルの作成時にコンポーネントをインストールするか、後でサイレント インストール時に使用する、応答ファイル テンプレートを作成するかどうかに応じて、以下のいずれかの手順に従います。

   - 応答ファイルを作成し、同時にコンポーネントをインストールするには、以下のコマンドを入力してから、インストールを実行します。

     `setup -options-record` *`install_response_file`*

   - コンポーネントをインストールせずに応答ファイルを作成するには、以下のコマンドを入力し、必要な値をテンプレートに入力します。

     `setup –options-template` *`install_response_file`*

   注: 応答ファイルを生成および実行する際、完全修飾パス名を使用します。たとえば、responsefile.txt は有効ではありませんが、C:¥responsefile.txt は有効です。

3. 以下のコマンドを実行して、サイレント インストールを開始します。

   `setup –options` *`install_response_file`* `-silent`

Connector Xpress のサイレント アンインストール

Connector Xpress では、サイレント モードのアンインストールがサポートされます。サイレント アンインストールを実行するには、応答ファイルを作成する必要があります。

次の手順に従ってください:

1. コマンド ウィンドウで、以下のいずれかのアンインストール ディレクトリに移動します。
   - Windows

     `C:¥Program Files¥CA¥Identity Manager¥Connector Xpress¥_uninst`
   - UNIX

     `/opt/CA/IdentityManager/ConnectorXpress/_uninst`
2. 応答ファイルの作成時にコンポーネントをアンインストールするか、後でサイレント アンインストール時に使用するか、応答ファイル テンプレートを作成するかどうかに応じて以下のいずれかの手順に従います。
   - 応答ファイルを作成し、同時にコンポーネントをアンインストールするには、以下のコマンドを入力してから、アンインストールを実行します。

     `uninstaller -options-record` *`uninstall_response_file`*
   - コンポーネントをインストールせずに応答ファイルを作成するには、以下のコマンドを入力し、必要な値をテンプレートに入力します。

     `uninstaller –options-template` *`uninstall_response_file`*

   注: 応答ファイルを生成および実行する際、完全修飾パス名を使用します。たとえば、responsefile.txt は有効ではありませんが、C:¥responsefile.txt は有効です。
3. 以下のコマンドを実行して、サイレント インストールを開始します。

   `uninstaller –options` *`uninstall_response_file`* `-silent`

メタデータのアップグレード

Connector Xpress の古いプロジェクトを新バージョンで開くとき、Connector Xpress によって自動的に既存のメタデータがアップグレードされます。 このリリースの新機能を活用すると、既存のメタデータに自動的に追加されます。

注: Connector Xpress で初めて古いプロジェクトを開くとき、メタデータを再展開し (P. 119)、CA Identity Manager ユーザ コンソールのアカウント画面を生成 (P. 80) してください。

ユーザ設定の格納場所

Java Preferences API を使用して Connector Xpress によってユーザ設定が格納されます。

Windows では、Java Preferences API によって以下のレジストリ キーにデータが格納されます。

HKEY_CURRENT_USER¥Software¥JavaSoft¥Prefs

UNIX では、Java Preferences API によって以下のフォルダ下にあるユーザのホーム ディレクトリにデータが格納されます。

.java/.userPrefs

注: アンインストーラではこのデータは削除されないので、今後インストールしてもユーザ設定は保持されます。

環境設定の設定

ボックス を使用して、Connector Xpress のルック アンド フィール、検索制限、コマンド パラメータなどの、外見および動作のさまざまな要素を変更できます。

次の手順に従ってください:

1. ［Tools］ - ［Preferences］をクリックします。

   ［Connector Xpress Preference］ダイアログ ボックスが表示されます。
2. ダイアログ ボックスのフィールドに入力して、［OK］をクリックします。

   Connector Xpress によってユーザの基本設定が適用および保存されます。

詳細情報:

［Preferences］ダイアログ ボックス (P. 197)

設定の格納場所

設定は Java Preferences API を使用して格納されます。 API によって、Windows 上のレジストリ、および Solaris 上にあるユーザのホーム ディレクトリ下のファイルに値が格納されます。

Windows では、以下のレジストリ キーに値が格納されます。

HKEY_CURRENT_USER¥Software¥JavaSoft¥Prefs¥com¥ca¥iam¥conman

JNDI データ ソース用 TLS/SSL のセットアップ

Connector Xpress では、TLS/SSL を有効にして JNDI データ ソースを定義できます。

JNDI データ ソースの TLS/SSL が有効な場合、JNDI データ ソース証明書を Connector Xpress の信頼された証明書ストアにインポートします。

次の手順に従ってください:

1. ターゲット証明書チェーンの各発行者用に PEM ファイルを作成します。
2. 各ファイルに対して以下のコマンドを入力します。

   ```
   keytool –import –trustcacerts –keystore
   "conxp-home/conf/ssl.keystore" –file trusted-certificate.pem
   ```

   JNDI データ ソース証明書が ssl.keystore にインポートされます

   各 PEM ファイルに対してこのコマンドを繰り返します。
3. Connector Xpress を再起動します。
4. Connector Xpress の TLS/SSL 接続をテストします。

データ ソースの設定

データ ソースとは、Connector Xpress がコネクタを組み立てることができるサーバまたはスキーマ情報の他のソースへの参照です。

Connector Xpress は、サポートされているベンダーの以下のデータ ソースをサポートします。

- **JDBC** -- JDBC API を使用してアクセスできるサービス。一般的にはリレーショナル データベース サーバ。
- **JNDI** -- 通常は LDAP サーバなど、JNDI API からアクセスできる任意のサービス。

データ ソースに接続して、そのスキーマを読み取り、Connector Xpress を使用して、新しいエンドポイント タイプにスキーマをマッピングできます。 Connector Xpress は、設定するデータ ソースのリストを保持します。 JNDI または JDBC データ ソースをセットアップできます。

注: Connector Xpress プロジェクトを作成すると、データ ソースのセットアップを要求するメッセージが表示されます。

次の手順に従ってください:

1. ［Tools］ - ［Data Sources］をクリックします。

   ［New Project］ダイアログ ボックスに［Select Data Source］が表示されます。

2. ［Add］をクリックします。

   ［Source Types］ダイアログ ボックスが表示されます。

3. リストから利用可能なデータ ソース タイプを選択し、［OK］をクリックします。

   追加しているデータ ソース タイプに固有の［Edit Source］ダイアログ ボックスが表示されます。

4. ダイアログ ボックスのフィールドに入力し、［Test］をクリックして、設定と認証の詳細を確認します。

   選択した認証方法に応じて、さらに認証用のダイアログ ボックスが表示される場合があります。

   ダイアログ ボックス上のフィールドにすべて入力し、テストを続行します。

5. テストが正常に完了したら、［OK］をクリックします。

   選択したデータ ソース タイプが正常に設定されました。

スタンドアロン CA IAM CS と通信できるように CA IAM CS ホストを設定する

スタンドアロン CA IAM CS との通信には、クライアントが使用できる 2 つのポート（non tls（ssl）および ssl）があります。 コネクタ サーバ マネージャまたは Connector Xpress からスタンドアロン CA IAM CS にアクセスできるようにするには、CA IAM CS サーバ ホストのファイアウォールを設定してポートの通信を許可します。

Connector Xpress への Sybase および DB2 for z/OS 用ライセンス ファイルの指定

Sybase および DB2 for z/OS の両方が JDBC を使用します。いずれの場合もエンドポイントへの接続にライセンス ファイルが必要です。

インストール ガイドの以下のセクションには、これらのライセンス ファイルのセットアップ方法が説明されています。

- DB2 for z/OS コネクタ用ライセンス ファイルのセットアップ
- Sybase コネクタ用ライセンス ファイルのセットアップ

第 3 章: コネクタの作成

このセクションには、以下のトピックが含まれています。

プロジェクト

Connector Xpress では、コネクタの設定およびプロビジョニング サーバに展開する方法を指定するプロジェクトを作成できます。

Connector Xpress で、プロジェクトはコネクタを表します。 プロジェクトには、以下が含まれます。

- データ ソースから取得したスキーマ。
- Connector Xpress によってスキーマがエンドポイントにマッピングされる方法を説明するメタデータの設定。

プロジェクトは保存および編集して、再度開くことで後から使用できます。

また、既存のメタデータに基づいてプロジェクトを作成し、次にクラス マッピングおよび操作バインディングを変更して、エンドポイントにコネクタを再展開できます。

Connector Xpress プロジェクトの作成方法

下記の方法でプロジェクトを作成できます。

- JNDI または JDBC データ ソースを指定するプロジェクトの作成 (P. 34)

  コネクタの設定およびプロビジョニング サーバに展開する方法を指定するプロジェクトを作成できます。

- データ ソースを指定せずにプロジェクトを作成する

  これにより、Connector Xpress を汎用メタデータ エディタとして使用 (P. 121) し、新しい属性を作成できます。

- テンプレートに基づいたプロジェクトの作成 (P. 35)

  この手順では、共通のエンドポイント スキーマのマップの基礎としてテンプレートを使用します。

- ウィザード モードを使用したテンプレートに基づいたプロジェクトの作成 (P. 36)

  ウィザードは、ユーザがテンプレートに基づいて新しいプロジェクトを開始し、アカウントとグループのクラスをマッピングする基本的な手順をサポートします。

テンプレート

テンプレートは、共通のエンドポイント スキーマのマップの基礎として使用できます。 たとえば、RFC などの規格が広く使用されるスキーマを定義して、完全なカスタム スキーマが一般的ではない LDAP のプロジェクトでテンプレートから開始すると便利です。 テンプレートには、アカウント管理画面を描画するのに CA Identity Manager が使用する情報が含まれています。

以下のテーブルでは、特に一般的に用いられているテンプレートの一部を取り上げます。テンプレートの完全なリストについては、［Project］-［Create New from Template in Connector Xpress］を選択し、各テンプレートをクリックして簡単な説明を表示します。

重要: 正しいパフォーマンスを得られるように、特定のベンダーのマッピングを作成する場合は、Connector Xpress で提供されるベンダー固有のテンプレートを基礎として使用することをお勧めします。

プロジェクト名の設定	エンドポイント タイプ	説明	メタデータ ファイル名
JNDI NIS NetGroup	JNDI	NIS Netgroup Schema をサポートする LDAP エンドポイントで使用します。 このテンプレートは、高度な関連付け処理を説明します。	jndi_assoc_nisnetgroup_metadata
JNDI inetOrgPerson（共通）	JNDI	LDAP inetOrgPerson。 ベンダー固有のテンプレートが不要な場合に、このテンプレートを使用します。	jndi_inetorgperson_common_metadata
Lotus Notes Domino	JNDI	Lotus Notes Domino サーバ。このテンプレートは、eTLNDCustomAttribute* および eTLNDCustomCapabilityAttribute* 属性（後のセットはアカウント テンプレート同期に関連します）を簡単にマッピングできます。	lnd_metadata

プロジェクト名の設定	エンドポイント タイプ	説明	メタデータ ファイル名
SDK DYN Compound	任意	SDKDYN と類似していますが、複合値の使用方法を説明しています。このテンプレートは、複合値を使用して複雑なデータを JSON 構文の単一文字列で表します。たとえば '{"attr1": 42, "attr2": ["a", "b"], attr3: { "objName" : "jack" } }' は属性が 3 つある最上位オブジェクトを表します。1 番目は整数（42）で、次は文字列の配列です。最後は入れ子のオブジェクトです。	sdkcompound_metadata
SDK DYN	任意	ソフトウェア開発キットのデモ版コネクタ。 このテンプレートは、推奨された eTDYN* スキーマを使用して、CA IAM CS ホスト コンピュータ上のローカル ファイルへのプロビジョニング情報を保存する、フラットな（つまり、階層がない）大文字と小文字を区別するコネクタです。 フラットなので、コンテナはエンドポイント上に実際格納されない仮想コンテナです。	sdkdyn_metadata
SDK DYN Script	任意	SDKDYN と類似していますが、JavaScript に実装されます。このテンプレートは、最上位の名前空間にある connectorXML メタデータを使用して、JavaScript でコネクタ全体（すべての操作バインディング）を実装する方法と通常は connector.xml ファイルにある設定情報を説明します。	sdkscript_metadata

プロジェクト名の設定	エンドポイント タイプ	説明	メタデータ ファイル名
SDK DYN UPO Script	任意	SDK DYN Script と類似していますが、ローカル ファイルに書き込む代わりに電子メールを送信します。このコネクタには、情報をローカル ファイルに書き込む代わりに電子メールを送信するという点を除いて、非推奨の C++ UPO コネクタと類似した機能があります。	sdkuposcript_metadata

テンプレートのロケーション

テンプレート ファイルはすべて、*Connector Xpress インストール ディレクトリ*/conf/templates に格納されています。

適切なテンプレートをすべて検索するため、Connector Xpress によって、このディレクトリが再帰的に検索されます。

プロジェクトの作成

コネクタを作成するには、コネクタの設定およびプロビジョニング サーバに展開する方法を指定するプロジェクトを作成できます。

次の手順に従ってください:

1. ［Project］ - ［New］をクリックします。

   ［Select Data Source for new Project］ダイアログ ボックスが表示されます。

2. 使用するデータ ソースがリストに表示される場合は、手順 4 に進みます。
3. 使用するデータ ソースがリストに表示されない場合は、以下の手順に従います。

   a. ［Add］をクリックします。

      ［Source Types］ダイアログ ボックスが表示されます。

   b. リストから利用可能なデータ ソース タイプを選択し、［OK］をクリックします。

      追加するデータ ソースのタイプに固有の［Edit Source］ダイアログ ボックスが表示されます。

   c. 選択したデータ ソース タイプを設定するには、ダイアログ ボックスのフィールドに入力し、［OK］をクリックします。

      ［Select Data Source for new Project］ダイアログ ボックスが表示され、追加したデータ ソースがリストに表示されます。

4. リストから使用するデータ ソースを選択し、［OK］をクリックします。

   ［Enter Password for Data Source］ダイアログ ボックスが表示されます。

5. 選択したデータ ソース用のパスワードを入力し、［OK］をクリックします。

   Connector Xpress によって以下が実行されます。

   - 最上位エンドポイント タイプ ノードが含まれるマッピング ツリーが表示されます。
   - マッピング ツリーに「User Account」という名前のクラスを作成します（デフォルトの場合）。
   - ［Endpoint Type Details (P. 166)］ダイアログ ボックスが表示されます。

6. 適切なエンドポイント データを作成するマッピングを作成します。
7. ［Project］、［Save］または［Save As］をクリックします。

   ［Save Project As］ダイアログ ボックスが表示されます。
8. コネクタ プロジェクトを保存するフォルダ、ファイル名および保存されたコネクタのファイル タイプを指定し、［Save］をクリックします。

   プロジェクトが保存されます。

   注: デフォルト ファイル拡張子（.con）を使用し、プロジェクトのコピーを保管することをお勧めします。 コネクタを展開後、バインディングが格納されている Connector Xpress プロジェクト ファイルを開かないと、ストアド プロシージャ バインディングを表示できません。

詳細情報:

［Select Data Source for new project］ダイアログ ボックス (P. 203)
［Source Types］ダイアログ ボックス (P. 205)
プロジェクト (P. 29)

テンプレートを使用したプロジェクトの作成

テンプレートは、共通のエンドポイント スキーマのマップの基礎として使用して、新しいプロジェクトを開始できます。

次の手順に従ってください:

1. ［Project］メニューで、［Create New from Template］をクリックします。

   ［Select Template (P. 204)］ダイアログ ボックスが表示されます。
2. プロジェクトの基礎に使用するテンプレート (P. 31) を選択します。

   ［Select Data Source for a new project］ダイアログ ボックスが表示されます。
3. ［Select Data Source for a new project］ダイアログ ボックスの詳細をすべて入力し、［OK］をクリックします。

   ［Endpoint Type Details (P. 166)］ダイアログ ボックスが表示されます。
4. 必要な属性マッピングをすべて実行します。
5. プロジェクトを保存します (P. 40)。

ウィザード モード

ウィザード モードからテンプレートを開いてプロジェクトを開始することもできます。ウィザードを使用してプロジェクトを開始すると、選択したテンプレートに応じてアカウント クラスのみ、またはアカウント クラスおよびグループ クラスをマッピングする基本的な手順をステップ バイ ステップでサポートします。マッピングには、プロジェクトの開始時にウィザードを使用して、後で他のダイアログ ボックスを使用することもできます。

ウィザードでは、以下のマッピング情報を入力できます。

1. エンドポイント タイプ名および説明。
2. アカウント マッピングのネイティブ クラス。
3. アカウント属性のマッピング。
4. グループ クラスが含まれるテンプレート プロジェクトについては、以下のマッピング情報を入力できます。
    a. グループ マッピングのネイティブ クラス。
    b. グループ属性のマッピング。
    c. アカウント クラスとグループ クラスの関連付け。

ウィザードを使用したプロジェクトの作成

ウィザードを使用して、アカウントおよびグループ属性のマッピングなどのマッピングの基本的な情報を入力できます。マッピングには、プロジェクトの開始時にウィザードを使用して、後で他のダイアログ ボックスを使用することもできます。

次の手順に従ってください:

1. ［Project］メニューで、［Create New from Template］をクリックします。

   ［Select template］ダイアログ ボックスが表示されます。

2. プロジェクトの基礎に使用するテンプレート (P. 31) を選択します。

3. ［Open in Wizard Mode］をクリックします。

   ［Select Data Source for a new project］ダイアログ ボックスが表示されます。

4. ［Select Data Source for a new project］ダイアログ ボックスの詳細をすべて入力し、［OK］をクリックします。

   ［Endpoint Type Details］ダイアログ ボックスが表示されます。

5. ウィザードの指示に従い、マッピングを完了します。

   注: ウィザードがアクティブの場合、マッピング ツリーは読み取り専用になります。ウィザードの手順を進めると、マッピング ツリーの関連するノードが展開され強調表示されます。

6. マッピングの概要を確認してから、［完了］をクリックします。

7. 必要な属性マッピングをすべて実行します。

既存メタデータに基づいたプロジェクト

動的コネクタおよび静的コネクタに設定された eTMetadata および eToperation バインディング セットのコンテンツに基づいてプロジェクトを作成できます。既存のメタデータに基づいてプロジェクトを作成すること、以下を実行できます。

- Connector Xpress で、コネクタ タイプのクラス マッピングおよび操作バインディングを確認および変更できます。
- 変更をエンドポイントに展開 (P. 44)できます。

既存のメタデータに基づいてプロジェクトを作成すること、以下の場合に役立ちます。

- コネクタの作成に使用された元のプロジェクト ファイルが利用可能できなくなった場合。
- 静的コネクタ タイプなど、コネクタのプロジェクト ファイルが存在しない場合。
- コネクタ タイプの最新の変更が反映されたプロジェクト ファイルが必要な場合。

既存のメタデータに基づいたプロジェクトの作成

コネクタの元のプロジェクトが存在しない場合に、静的コネクタおよび動的コネクタのクラス マッピングおよび操作バインディングを確認または変更するには、既存のメタデータに基づいたプロジェクトを作成します。

次の手順に従ってください:

1. プロビジョニング サーバ ツリーでプロジェクトの基礎になるエンドポイントを右クリックし、［Create Project］をクリックします。

   進捗ダイアログ ボックスが表示されます。 動的コネクタを選択した場合、［Select Data Source for New project］ダイアログ ボックスが表示されます。
2. プロンプト画面が表示された場合、プロジェクトのデータ ソースを選択し、［OK］をクリックします。

   プロジェクトが作成され、マッピング ツリーにメタデータを入力します。
3. 関連するプロビジョニング属性に Account クラスのネイティブ属性をマッピングします。
4. Connector Xpress プロジェクトを保存します。(P. 40)
5. 必要に応じてメタデータを再展開します。(P. 119)

既存プロジェクトを開く

既存のプロジェクトを開くには、［Project］-［Open］をクリックし［Open Project］ダイアログ ボックスで目的のファイルをダブルクリックします。

プロジェクトが開かれ、［Connector Xpress］ウィンドウのマッピング ツリーにプロジェクトのマッピングが表示されます。

最近使用したプロジェクトを再度開く

最近使用したプロジェクトを再度開くには、［Project］-［Open Recent］をクリックし、目的のプロジェクトを選択します。

プロジェクトが開かれ、［Connector Xpress］ウィンドウのマッピング ツリーにプロジェクトのマッピングが表示されます。

プロジェクトの編集

プロジェクト情報を変更するには、保存されたプロジェクトを開き、メタデータを編集します。

次の手順に従ってください:

1. 既存のプロジェクトを開きます (P. 39)。

   ［Select Data Source for new Project］ダイアログ ボックスが表示されます。

2. ダイアログ ボックスに情報を入力し、［OK］をクリックします。

   プロジェクトが開かれ、［Connector Xpress］ウィンドウのマッピング ツリーでプロジェクトのマッピングが表示されます。

   ［Enter Password for Data Source］ダイアログ ボックスが表示されます。

3. ダイアログ ボックスのフィールドに入力して、［OK］をクリックします。
4. メタデータを編集します。
5. コネクタを保存します。 (P. 40)
6. 必要に応じてコネクタを展開 (P. 44) します。

詳細情報:

［Enter Password for Data Source］ダイアログ ボックス (P. 165)
プロジェクト (P. 29)

プロジェクトの保存

新しい動的コネクタを作成後、将来の展開で使用できるようにコネクタ プロジェクトを保存できます。

次の手順に従ってください:

1. ［Project］、［Save］または［Save As］をクリックします。

   ［Save Project As］ダイアログ ボックスが表示されます。

2. コネクタ プロジェクトを保存するフォルダ、ファイル名および保存されたコネクタのファイル タイプを指定し、［Save］をクリックします。

   プロジェクトが保存されます。

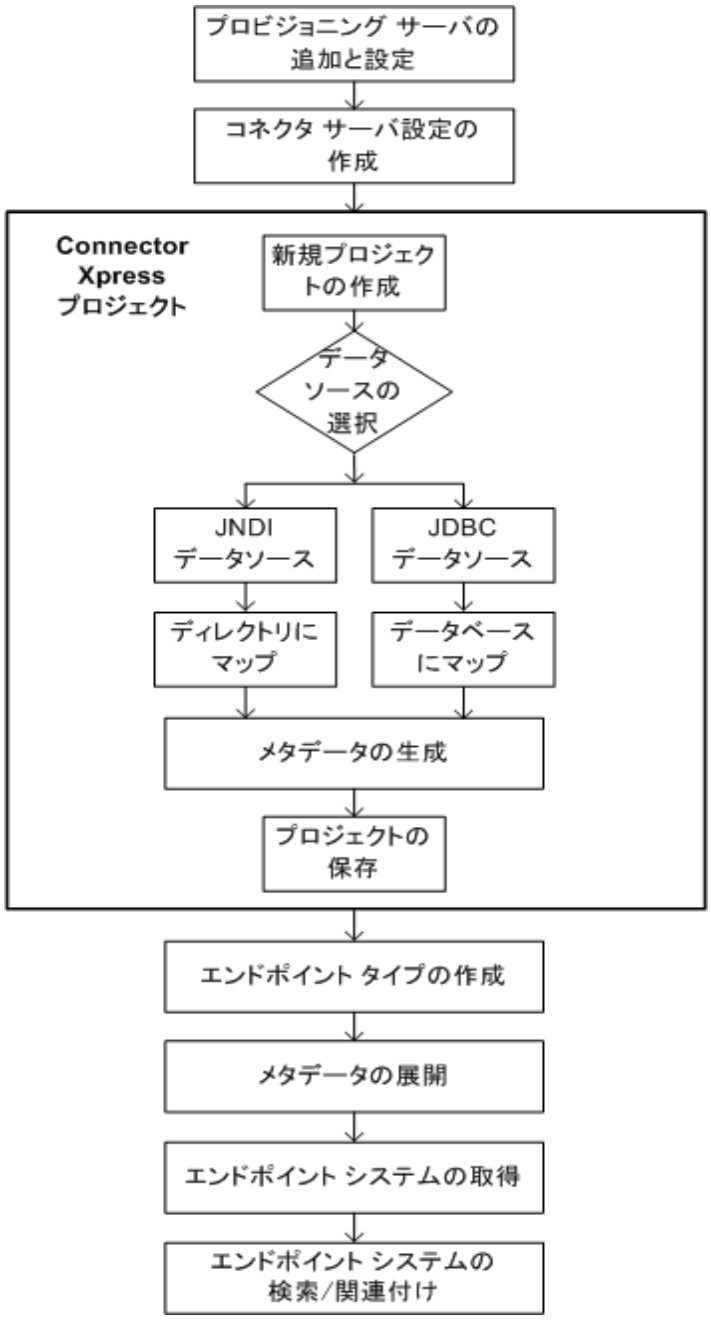
   注: デフォルト ファイル拡張子（.con）を使用し、プロジェクトのコピーを保管することをお勧めします。 コネクタを展開後、バインディングが格納されている Connector Xpress プロジェクト ファイルを開かないと、ストアド プロシージャ バインディングを表示できません。

詳細情報:

プロジェクト (P. 29)

コネクタを作成および展開する手順

以下の図では、Connector Xpress を使用して、コネクタを作成および展開するプロセスを示します。



例: 基本的な JNDI コネクタを作成および展開する手順

以下の手順では、基本的な JNDI コネクタを作成および展開する詳細について説明します。 基本的な JNDI コネクタを作成するには、以下の手順に従います。

1. Connector Xpress を起動します。

   Connector Xpress のメイン ウィンドウが表示されます。

2. 管理するエンドポイントがあるプロビジョニング サーバの接続詳細を追加および変更 (P. 96) します。

3. プロビジョニング サーバにコネクタ サーバが設定されていない場合は、エンドポイントの管理コネクタ サーバを作成 (P. 98) および設定 (P. 99) します。

   設定により、プロビジョニング サーバがそれぞれのエンドポイントをどのようにコネクタ サーバにルーティングするかが決定されます。

4. プロジェクトを作成します (P. 34)。

5. JNDI データ ソースをセットアップ (P. 14) し、JNDI データ ソースの接続および認証情報をテストします。

   Connector Xpress によって、この JNDI データ ソースからスキーマが取得されます。これにより、プロビジョニング サーバの新しいエンドポイントにスキーマがマッピングされます。

   注: プロジェクトを作成すると、Connector Xpress によってマッピング ツリーに User Account という名前のクラスが自動的に作成されます。

6. 関連するプロビジョニング属性にユーザ アカウント クラスのネイティブ属性をマッピング (P. 60) します。

7. プロジェクトを保存します。 (P. 40)

   将来の展開で使用するためにプロジェクトを保存できます。

8. 管理およびコネクタに展開するプロビジョニング サーバでエンドポイント タイプを作成 (P. 113) します。

9. エンドポイント タイプを管理する管理コネクタ サーバ (P. 99) を指定します。

   エンドポイント タイプを作成し管理コネクタ サーバを指定すると、以下が実行されます。

   プロビジョニング サーバがコネクタ サーバにエンドポイント タイプまたはそれぞれのエンドポイントをどのようにルーティングするかが指定されます。

10. コネクタ サーバにコネクタを展開します (P. 44)。

    メタデータを展開すると新しい動的 JNDI コネクタが作成されます。

11. エンドポイントを取得、検索、および関連付けします (P. 112)。

    Connector Xpress によって、取得したエンドポイントで検出されたアカウント、およびその他のオブジェクトがプロビジョニング サーバに入力されます。

    エンドポイントの関連付けとは、エンドポイントのアカウントを検査し、アカウント属性から自動的にユーザを作成し、グローバル ユーザ属性の設定を可能にするプロセスです。

    オブジェクトを管理するには、オブジェクトのエンドポイントを検索し、グローバル ユーザにオブジェクトを関連付けします。

12. CA Identity Manager ユーザ コンソールのアカウント画面を生成します (P. 80)。

コネクタの展開

Connector Xpress を使用して、メタデータ設定を設定し、コネクタを展開できます。Connector Xpress では、メタデータ設定をプロジェクト ファイル (P. 29)に保存し、そのファイルからコネクタを展開できます。 たとえば、テスト環境から実稼働環境にコネクタを移動させるときにプロジェクト ファイルを使用して、確実に同じ設定が使用されるようにすることができます。

コネクタの展開

コネクタを作成すると、そのコネクタのメタデータをプロビジョニング サーバに展開できます。

注: この手順では、コネクタ サーバ設定が作成済み (P. 98)であることが前提されています。

次の手順に従ってください:

1. 展開するメタデータが含まれる既存のプロジェクトを開きます (P. 39)。

   Connector Xpress がプロジェクトを開き、［Connector Xpress］ウィンドウの［Edit］ペインにメタデータが表示されます。

2. プロビジョニング サーバ ツリーで、［Provisioning Servers］ノードを展開し、コネクタを展開するサーバを選択します。

   ［Provisioning Server Password Required］ダイアログ ボックスが表示されます。

3. ダイアログ ボックスのフィールドに入力してサーバ用のパスワードを指定し、［OK］をクリックします。

4. サーバ展開し、［Endpoint Types］を右クリックして、［Create New Endpoint Type］をクリックします。

   ［Create New Endpoint Types］ダイアログ ボックスが表示されます。

5. ダイアログ ボックスのフィールドに入力して、新しいエンドポイント タイプの名前を定義し、［OK］をクリックします。

   注: エンドポイントの管理画面が利用可能になるまで遅延が発生する場合があります。 画面作成ステータスを確認するには、ユーザ コンソールで［サブミット済みタスクの表示］を使用するか、または、展開が完了したら電子メール 通知が送信されるようにワークフロー プロセスを設定します。

6. 新しいエンドポイントを右クリックし、［Acquire Endpoint］を選択します。

   ［Create New Endpoint］ダイアログ ボックスが表示されます。

7. ダイアログ ボックスのフィールドに入力して、新しいエンドポイントの名前とパスワードを指定し、［OK］をクリックします。

8. （オプション）エンドポイント ノードの下にある新しいエンドポイントを右クリックし、［Explore/Correlate endpoint］を選択します。

   ［Explore/Correlate Endpoint］ダイアログ ボックスが表示されます。

9. ダイアログ ボックスのフィールドを入力して Connector Xpress がエンドポイントを検索および参照する方法を指定し、［OK］をクリックします。

   Connector Xpress はプロビジョニング サーバにコネクタを展開します。

詳細情報:

［Create New Endpoint］ダイアログ ボックス (P. 149)
［Explore/Correlate Endpoint］ダイアログ ボックス (P. 167)
プロジェクト (P. 29)

コネクタの削除

Connector Xpress を使用して、コネクタを展開解除できます。 コネクタの展開を解除すると、プロビジョニング サーバおよび CA IAM CS からコネクタが削除されます。

コネクタを展開解除するために Connector Xpress を使用する場合、［Delete Endpoint Type］タスクが作成されます。

重要: 承認者が［Delete Endpoint Type］タスクを拒否すると、いくつかの画面は削除されません。承認者がすべてのエンドポイント削除要求を承認すること強くお勧めします。

JDBC コネクタのマルチテーブル サポート

複合クラスとは、属性のデータ型として使用できる非管理対象クラスです。

マルチテーブルの JDBC コネクタをサポートするには、エンドポイント オブジェクトに非管理対象クラスをマップし、アカウント クラス属性の新しいデータ型として複合クラスを使用します。

したがって、テーブルの単一の列からではなく、複数の列からの JDBC コネクタ値を使用して、単一の属性値に入力できます。

複合クラスは、命名属性が不要な点を除いて、通常のクラスのようにマッピングできます。

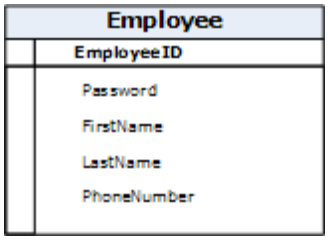
注: 複合型の詳細については、「Connector Programming Guide」を参照してください。

マルチテーブルをサポートする複合クラスの例

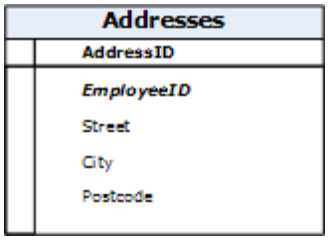
この例では、ユーザ アカウント クラスに 2 つのデータベース テーブルを組み合わせる、My_MultiTable_Connector という名前の単純なコネクタを作成する方法について説明します。

この例では、Employees テーブルおよび Address テーブルが格納された Multi_Table という名前のスキーマを使用します。

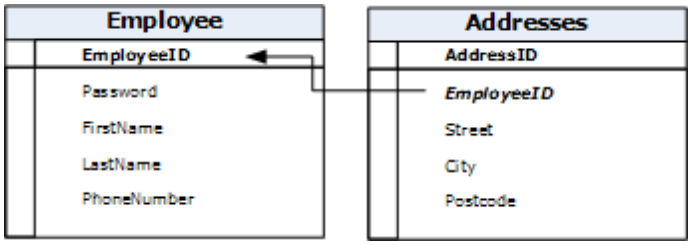
Employees テーブルは従業員レコードのメイン テーブルです。 テーブルでは以下の図で示すように、主キーに EmployeeID が使用されます。



Addresses テーブルでは以下の図で示すように、外来キーに EmployeeID が使用される住所の個別テーブルです。



以下の図では、テーブルの関係を示します。



例: マルチテーブルをサポートする複合クラスの作成

この例では、Multi_Table スキーマで Employees テーブルおよび Addresses テーブルを組み合わせます。組み合わせると、両方のテーブル内の属性は、ユーザ コンソール アカウント管理画面のエンドポイント アカウントから両方にアクセスできるようになります。

1. My_MultiTable_Connector という名前の JDBC プロジェクトを作成し、使用する JDBC データ ソースを指定します。
2. マッピング ツリーで［User Account］クラス ノードをクリックします。

   ［Map Class］ダイアログ ボックス (P. 179)が表示されます。
3. ［Schema］ドロップダウン リストから Multi_Table スキーマを選択し、［Table］ドロップダウン リストから Employees テーブルを選択します。
4. ［Name］列の属性に Employees テーブルの列をマッピングします。
5. ［Custom Types］ノードをクリックします。

   ［Custom Types］ダイアログ ボックスが表示されます。
6. Compound Type クラスの下で、［Add］をクリックし、テーブルに新しいクラスの名前を入力します（例： Address）。

   Connector Xpress によって、Address と呼ばれる複合クラスが作成され、マッピング ツリーに追加されます。 Address 複合クラスはデフォルトで［Unmanaged］に設定されます。

   Connector Xpress によって、Addresses という名前のアカウント属性が作成され、マッピング ツリーに追加されます。また、［Attribute Details］ダイアログ ボックスで複数値の属性チェック ボックスがオンされます。また、Address クラスから User Account クラスに直接関連付けが作成され、マッピング ツリーに［*with User Account*］という名前のノードも追加されます。Addresses 属性と直接関連付けにより User Account クラスに Address 複合クラスが組み込まれます。

   注: Address クラスは［Map Compound Class and Attributes］ダイアログ ボックスで *is compound value* メタデータ フラグが true に設定されています。このフラグによって、クラスが複合クラスであることが指定されます。

7. マッピング ツリーで［Address］クラス ノードをクリックします。

   ［Map Compound Class and Attributes (P. 180)］ダイアログ ボックスが表示されます。

8. ［Schema］ドロップダウン リストから Multi_Table スキーマを選択し、［Table］ドロップダウン リストから Address テーブルを選択します。

9. ［Name］列の属性に Address テーブルの列をマッピングします。

   注: 複合クラスの数値のフィールドをメタデータで［Sting］に設定します。フィールドが［Numeric］に設定されていると、複合クラスでルール文字列テンプレートがサポートされません。

10. ［Address］ノードの下の［With User Account］ノードをクリックします。

    ［Direct Association with User Account (P. 156)］ダイアログ ボックスが表示されます。

11. ［Address Attribute］フィールドで、［Employee ID］を選択します。

    Connector Xpress によって、この属性と User Account 命名属性が一致されます。

    注: Address クラスの Employee ID 属性は関連付け属性として機能します。 この属性には、この複合クラスを親クラスにリンクする属性として定義する関連タイプ メタデータ プロパティ（Assoc Type=COMPOUND_PARENT）設定されています。 ［Attribute Detail］ダイアログ ボックスにこのプロパティを表示するには、拡張メタデータ プロパティを表示 (P. 122)します。

12. ［User Account］ノードの下の、［Attributes］ノードをクリックします。

    ［Attributes Summary］ダイアログ ボックスが表示されます。

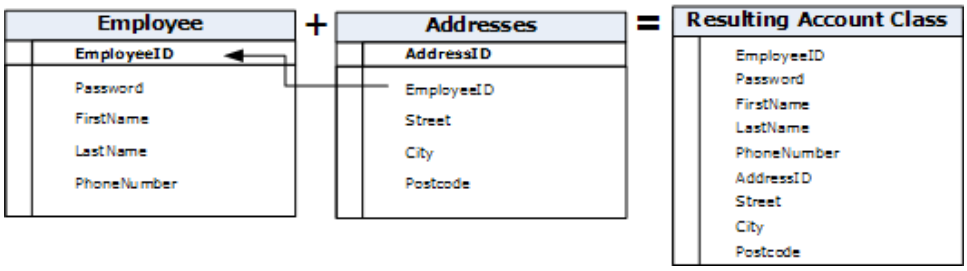
13. Connector Xpress でプレゼンテーション メタデータを作成（P. 84）することで、CA Identity Manager ユーザ コンソールのアカウント管理画面のタブおよびページとして表示されるグループおよびサブグループのマッピングされた属性をグループ化できます。

14. プロジェクトを保存し、新しいコネクタを展開します。

15. CA Identity Manager ユーザ コンソールのアカウント画面を生成します。（P. 80）

例: Employees および Addresses テーブルを組み合わせた結果

以下の図では、Employees テーブルおよび Addresses テーブルを組み合わせた結果を示します。



例: 生成されたアカウント画面

この例では、CA Identity Manager に My_MultiTable_Connector-RoleDef.Xml ファイルをインポートした後のアカウント管理画面を示しています。

アカウントおよびグループの管理

Connector Xpress の以前のリリースでは、グループ オブジェクト タイプをマッピングすると、グループとアカウントの間の関連付けが暗黙的に作成されていました。今回のリリースでは、エンドポイント システムのアカウントとグループの間の関連付けを詳細に制御できるようになりました。 グループを定義して、グループ メンバシップを定義するには、クラス間の関連付けを明示的に作成します。直接関連付け、逆の関連付け、または間接関連付けを作成できます。 クラス間の関連付けを作成することによりクラスがグループ クラスとして定義されます。

JNDI コネクタの場合、グループ クラス メンバ属性はタイプ DN（識別名）の値が含まれるように固定されています。このタイプの値はエンドポイント ディレクトリのルートと相対的に表現され、各グループに属するアカウントを列挙します。

グループ クラスのメンバ属性は仮想です。したがって、値は直接ルックアップされず group.member から計算する必要があるので、非効率的に値が取得されます。そのため、要求する場合には注意することをお勧めします。

一部の JNDI ベンダー（特に Novell eDirectory）は、スキーマで account.memberOf 属性を表示しています。ただし、すべてのベンダーで一貫した動作を保証するには、明示的にマッピングしないでください。 代わりに、CA IAM CS によって仮想属性として実装されます。

マッピング

マッピングは以下から構成されます。

- **One or more class mappings** -- クラス マッピングではそれぞれ、プロビジョニング サーバの DIT で単一のクラスのプロビジョニング オブジェクトが表現されます。 JDBC の場合、クラス マップによって単一のデータベース テーブルにマッピングされます。 LDAP の場合、クラス マッピングによって複数のネイティブ LDAP オブジェクト クラスにマッピングされます。 アカウント クラス マッピングはポリシー同期のようなプロビジョニング サーバ操作にとって重要なので、アカウント クラス マッピングにはいくつかの特殊なハンドリングがあります。
- **Multiple attribute mappings** -- クラス マッピングにはそれぞれ、複数属性マッピングを割り当てることができます。複数属性マッピングでは、プロビジョニング サーバのプロビジョニング オブジェクトの値それぞれが管理エンドポイント システムにあるオブジェクトの単一の値にマッピングされます。

複数属性マッピング

Connector Xpress では、複数属性（多数から単一）マッピングがサポートされます。つまり、ネイティブ属性を複数のプロビジョニング属性にマッピングできます。LDAP DYN テンプレート (P. 31)には、複数属性マッピングの例が格納されています。テンプレートのアカウント クラスでは、エンドポイントの *cn* 属性に Common Name および Account ID の両方がマッピングされています。Common Name はアカウント オブジェクトに含まれるべき LDAP の共通属性で、Account ID は、CA Identity Manager の共通属性セットに要求されるプロビジョニング命名属性なので便利です。

また、Account ID および *uid* は両方とも、エンドポイントのテンプレート内の *uid* 属性にあいまいにマッピングされます。

同じクラス内でネイティブ属性に重複してマッピングできません。 たとえば、*cn* がマッピングされる場合、再度 *cn* をマッピングできません。 ただし、*ambiguous* マッピングの一部として cn を再度マッピングできます。 たとえば LDAP DYN テンプレートで示されているように *uid* と一緒にマッピングします。

accountname、*cn*、*uid* に複数マッピングおよび *cn* に個別に直接マッピングするには、JNDI エンドポイントを完全に管理する必要があります。 これにより、*cn* または *uid* のいずれかを命名属性で使用できるようになり、*uid* が命名属性として使用されている場合にも、*cn* はエンドポイントに要求される条件を満たします。

同じネイティブ属性に複数属性をマッピングする

JNDI エンドポイントには *cn* または *uid* の属性を使用して指定したアカウントを含めることができるので、以下の手順を実行することをお勧めします。

- *cn* および *uid* 属性の両方に Account ID をマッピングする
- エンドポイントを管理するため、*cn* と *uid* に単一のあいまいではないマッピングを提供します。

次の手順に従ってください:

1. ［Project menu］メニューで、［New］をクリックします。

   プロジェクトを作成すると、Connector Xpress によってマッピング ツリーにユーザ アカウントのプロビジョニング クラス ノードが自動的に作成されます。

   ［Select Data Source for new Project］ダイアログ ボックスが表示されます。
2. プロジェクトに使用するデータ ソースを選択します。

   ［Endpoint Types］ダイアログ ボックスが表示されます。
3. ［Endpoint Type Details］ダイアログ ボックスで、コネクタの名前、説明およびバージョンを指定します。

   注: これらのフィールドは説明を目的としています。
4. マッピング ツリーで、［Map Attributes］ダイアログ ボックスをクリックします。

   ［Map Attributes］ダイアログ ボックスが表示されます。
5. ［Maps To］列で、マッピングするプロビジョニング属性の編集ボタンをクリックします。

   ［Attribute］リストが表示されます。
6. マッピング先のネイティブ属性を選択します。
7. ［Maps To］列で、マッピングする以下のプロビジョニング属性の編集ボタンをクリックします。

   ［Attribute］リストが表示されます。
8. 手順 5 で選択したネイティブ属性を選択します。
9. プロジェクトを保存します。（P. 40）

あいまいな属性のマッピング

あいまいな（単一から多数）マッピングをサポートするには、複数のネイティブ属性にプロビジョニング属性を 1 つマッピングします。

次の手順に従ってください:

1. ［Project menu］メニューで、［New］をクリックします。

   プロジェクトを作成すると、Connector Xpress によってマッピング ツリーにユーザ アカウントのプロビジョニング クラス ノードが自動的に作成されます。

   ［Select Data Source for new Project］ダイアログ ボックスが表示されます。

2. プロジェクトに使用するデータ ソースを選択します。

   ［Endpoint Types］ダイアログ ボックスが表示されます。

3. ［Endpoint Type Details］ダイアログ ボックスで、コネクタの名前、説明およびバージョンを指定します。

   注: これらのフィールドは説明を目的としています。

4. マッピング ツリーで、［Map Attributes］ダイアログ ボックスをクリックします。

   ［Map Attributes］ダイアログ ボックスが表示されます。

5. ［Maps To］列で、マッピングするプロビジョニング属性の編集ボタンをクリックします。

   ［Attribute］リストが表示されます。

6. ［属性］リストで Ctrl+ クリックするか Shift+ クリックして、マッピングする複数のネイティブ属性を選択します。

7. プロジェクトを保存します。（P. 40）

関連付けのタイプ

Connector Xpress では以下のタイプの関連付けを作成できます。

- （JNDI と JDBC）直接関連付け
- （JNDI と JDBC）逆の関連付け

  注: 複数テーブルをサポートする JDBC コネクタを作成する場合、複合クラスとユーザ アカウント クラス間の関連付けでは、逆の関連付けがサポートされていません。

- （JDBC のみ）間接関連付け

直接関連付け

*直接*関連付けは任意の 2 つのクラスのオブジェクト間の関連付けで、関連値がいずれかのオブジェクトに直接格納される場合を指します。

直接関連付けは、順方向（グループからアカウント）および逆方向（アカウントからグループ）が可能です。順方向で直接関連付けを作成すると、関連付けのグループ側からのグループに属するアカウントを管理できます。逆方向で直接関連付けを作成する（つまり、逆の関連付け (P. 55)）と、関係のアカウント側からの関連付けを管理させます。

順方向での直接関連付けは、グループによりアカウント メンバが格納される場所である、エンドポイント オブジェクトに関する関連付け情報の自然な表現と一致します。逆方向内の直接関連付けは、ネイティブ システムにこの情報が格納されていなくても、アカウントが属するグループを定義して、関連付けを逆に表現します。

通常は、両方向（直接関連付けと逆の関連付け）を同時に確立します。［Direct Association (P. 156)］ダイアログ ボックスを使用して、両方の関連付けを同時に作成および編集できます。

直接関連付けでは、グループによって属するアカウントのディレクトリ関連 DN が直接格納されます。そのため、inetOrgPerson.cn に groupOfNames.member をマッピングすると、クラス間に直接関連付けが作成されます。

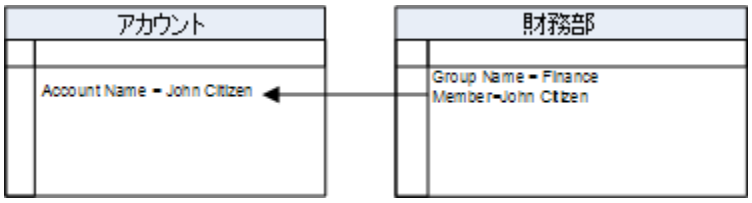
注: これらのオブジェクト クラスは両方とも inetOrgPerson スキーマの一部です。エンドポイントの実装の詳細が若干異なります。異なる点は、groupOfUniqueNames では、関連付けられたエントリが一意であることを保証するため、メンバをリストではなくセットとして格納します。

直接関連付けでは、参照はエンドポイント上の複数値属性に直接永続化されます。たとえば、LDAP では、グループのメンバ属性によって、参照に格納されたアカウントに参照を直接格納します。

JDBC コネクタのクラス間の関連付けを作成する場合、1 対 1 および 1 対複数の関係には、直接関連付けを使用することをお勧めします。 複数対複数の関係（たとえばアカウントの memberOf およびグループの members の関係）には、直接関連付けではなく間接関連付け使用します。

例: 直接関連付け

以下の例では、アカウント クラスと財務部クラスの間にマッピングされた直接関連付けを示します。 財務部クラスには、そのメンバ属性に属するアカウントが格納されます。



逆の関連付け

逆の関連付けは、Class1 から Class 2（直接関連付け）、および Class 2 から Class 1（逆の関連付け）の形式の 2 つのクラスに属するオブジェクト間の直接関連付けです。 逆の関連付けは基本的に双方向です。

通常、ほとんどのエンドポイントは、アカウントとグループ間の関連付けを関連付けのグループ側から 1 つの側しか管理できません。 たとえば、関連付けのグループ側からのグループに属するアカウントを管理できます。逆の関連付けを作成することで、関連付けのアカウント側からアカウントが属するグループをプロビジョニングおよび管理できます。

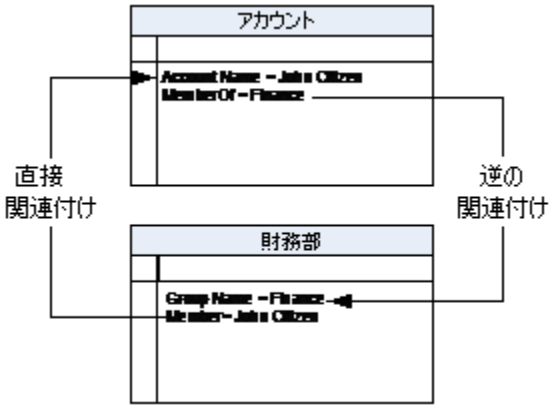
ほとんどの双方向関連付けは 1 つのクラスに物理属性があり、もう一方のクラスに仮想属性があります。最初に物理関連付け属性を定義することをお勧めします。

通常は、両方向（直接関連付けと逆の関連付け）を同時に確立します。 ［Direct Association (P. 156)］ダイアログ ボックスを使用して、両方の関連付けを同時に作成および編集できます。

逆の関連付けが、逆の関連付けを指定したクラス ノードの下のマッピング ツリーに表示されます。

例: 逆の関連付け

以下の図では、アカウントが属するグループが含まれるアカウント内の属性をマッピングするとき、グループの命名属性に作成されるアカウントおよびグループ財務クラス間の直接関連付けおよび逆の関連付け（つまり、双方向関連付け）を示します（例： *memberOf 属性*）。



間接関連付け

3 番目のエンティティによって 2 つのクラスのオブジェクト間の関連付けが定義される場合に、*間接関連付け*が発生します。 たとえば、他の 2 つのデータベーステーブルを 1 つにバインドする中間のテーブルが該当します。

間接関連付けでは、関連付けはオブジェクトの 1 つ上のプロパティとしてではなく、独立したエンティティとして格納されます。

たとえば、*メンバシップ テーブル*などの個別のアカウントとグループの間の関連付けを識別する中間のテーブルがある場合、アカウント オブジェクトとグループ オブジェクトの間に間接関連付けが存在します。

*直接*関連付けでは、オブジェクトには別のオブジェクトを直接ポイントする属性が割り当てられます。 ただし、*間接関連付け*で、関連するオブジェクトには両方とも互いにではなくメンバシップ テーブルをポイントする属性があります。

メンバシップ テーブルによって、各グループのメンバが定義されます。テーブルには関連する個別のアカウントおよびグループを識別する関連付けマッピング（アカウントが属するグループなど）、および各グループに属するアカウントのリストが含まれます。メンバシップ テーブルでは、オブジェクト間の関連付けは複数対複数のマッピング テーブルに格納されます。

メンバシップ テーブルを使用すると、アカウントとグループ オブジェクトの間の関連付けのルックアップ テーブルを個別に指定して、関連する属性をマッピングできます。

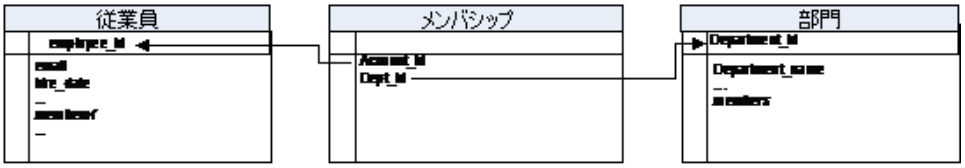
間接関連付けを定義するには、両方の関連付けオブジェクトへの参照が含まれるメンバシップ テーブルを指定します。

2 つのオブジェクト間に間接関連付けを作成すると、Connector Xpress によって逆の関連付けが自動的に作成されます。つまり、間接関連付けは基本的に双方向定義になります。

注: 間接関連付けは、JDBC マッピングでのみ作成できます。

例: 間接関連付け

以下の図では、メンバシップ テーブルを使用して、従業員クラスと部門クラスの間にマッピングされた間接関連付けを示します。以下にスキーマとデータベースの例を示します。



データベースには以下の 3 つのテーブルがあります。

- **Employees テーブル –** 組織の従業員をリスト表示します。

  仮想属性 memberOf には、従業員がメンバである部門の Group ID が含まれます。

- **Membership テーブル –** Employees と Departments の間の関連付けを定義します。 テーブルには関連する個別の従業員および部門を識別する関連付けマッピング（従業員が属する部門など）、および各部門に属する従業員のリストが含まれます。

- **Departments テーブル –** 組織の部門をリスト表示します。

  仮想属性メンバには、部門に属する従業員のリストである Account ID が含まれています。

詳細情報:

グループ メンバシップを定義する方法 (P. 59)
間接関連付けを作成する方法 (P. 65)

グループ メンバシップを定義する方法

グループ メンバシップを定義し、クラスをグループ クラスとして定義する、クラス間の関連付けを作成するには、以下の手順に従います。

1. アカウント クラスを作成し、その属性をマッピングします。

   注: プロジェクトを作成すると、デフォルトでは Connector Xpress によってユーザ アカウント クラスが作成されます。

2. グループ クラスとして定義するクラスを作成し、その属性をマッピングします。

   注: アカウント クラスとグループ クラスのマッピングにウィザードを使用すると、ウィザードによって自動的にグループ クラスおよび入力する必要がある関連付けの詳細が作成されます。

3. クラス間に直接関連付け、逆の関連付け、または間接関連付けを作成し、グループの group.member または group.uniqueMember 属性をアカウント命名属性にマッピングします。

   関連付けを作成するにより、グループ メンバシップが定義され、クラス間の関連付けが作成されます。

アカウント クラスのマッピング例

この例では、管理者が JNDI データ ソースにアカウント クラスをマッピングする場合に、従う手順について説明します。 プロビジョニング アカウント クラスにエンドポイントのアカウント クラスをマッピングする方法について説明します。ここでは、管理者が JNDI データ ソースをセットアップしたと仮定します。

次の手順に従ってください:

1. ［Project menu］メニューで、［New］をクリックします。

   ［Select Data Source for new Project］ダイアログ ボックスが表示されます。

2. プロジェクトに使用するデータ ソースを選択します。

   ［Endpoint Types］ダイアログ ボックスが表示されます。

3. ［Endpoint Type Details］ダイアログ ボックスで、コネクタの名前、説明およびバージョンを指定します。

   注: これらのフィールドは説明を目的としています。

   プロジェクトを作成すると、Connector Xpress によってマッピング ツリーにユーザ アカウントのプロビジョニング クラス ノードが自動的に作成されます。

4. マッピング ツリーで［User Account］ノードをクリックします。

   ［Map Account Class］ダイアログ ボックスが表示されます。

5. ［Add］構造クラスのリストからマッピングする inetOrgPerson などのエンドポイント タイプのオブジェクト クラスを選択します。

6. マッピング ツリーで［User Account］ノードの下の［Attributes］ノードをクリックします。

   ［Map Attributes］ダイアログ ボックスが表示されます。

7. Map Object Class Attribute マッピング テーブルのプロビジョニング属性にエンドポイントの必須属性をマッピングします。 たとえば、Last Name に cn Account ID および sn をマッピングします。

   注: テーブルのマッピング時に、テーブル リストに入力されない場合、データベースでスキーマ メタデータのトランザクションまたはロックが残存していないことを確認します。

8. 他の必要な属性（ユーザ パスワード、通り、肩書きなど）をマッピングします。

9. マッピング ツリーの［Classes］ノードをクリックします。

   ［Mapped Classes］ダイアログ ボックスには、マッピングしたクラスの概要が表示されます。このダイアログ ボックスを使用して、ネイティブ クラスを作成したプロビジョニング マッピングに変更できます。

10. マッピング ツリーで、［Attributes］ノードの下の［Account Id］ノードをクリックします。

    ［Attribute Details］ダイアログ ボックスが表示されます。

    このダイアログ ボックスでは、各フィールドに割り当てられた LDAP 属性、そのデータ型、JIAM が使用する JavaBean プロパティ名、およびフィールドが必要かどうか（ヌル値を許可する）、および長さ制約が表示されます。

11. ［Attributes］ノードの下で［Last Name］ノードをクリックします。

    デフォルトのポリシー値が設定された状態で［Attribute Details］ダイアログ ボックスが表示されます。

    注: 一般的なプロビジョニング属性に必要な属性をマッピングすると、Connector Xpress によって、デフォルトで、デフォルト アカウント テンプレート値が設定されます。

12. プロジェクトを保存します。

直接関連付けおよび逆の関連付けの作成方法

2 つのクラス（アカウント クラスなど）とグループ クラスの間に直接関連付けを作成するには、以下の手順に従います。

1. 新しいプロジェクトを開始し、プロジェクトのデータ ソースを指定します。
2. アカウント クラスを作成しマッピングします。
3. グループに定義するクラスを作成し、名前のグループのエントリを定義するエンドポイントのクラス（*groupOfNames* など）にマッピングします。
4. グループの名前およびそのメンバ属性をマッピングします。
5. グループ クラスとアカウント クラスの間に直接関連付けを作成することを指定します。
6. アカウントの命名属性にグループのメンバ属性をマッピングします。

   グループのメンバ属性をマッピングすると、グループのメンバシップ属性がアカウントの命名属性によって入力されていることが指定されます。

   これは、グループ クラスとアカウント クラスの間の関連付けを説明し、アカウント クラスおよびグループ クラスの間に直接関連付けを作成します。
7. アカウント クラスとグループ クラスの間に逆の関連付けを作成することを指定します。
8. グループの命名属性にアカウント クラス memberof 属性をマッピングします。

   注: ネイティブ アカウント クラスに memberOf 属性がない場合、仮想 memberOf 属性を作成し、グループの命名属性にマッピングします。

   Connector Xpress によって、マップしたアカウントとグループのクラス間に直接関連付けおよび逆の関連付けが作成され、［User Accounts］ノードの下に関連付けが自動的に作成および表示されます。

直接関連付けおよび逆の関連付けの例

この例では、管理者がアカウント クラスとグループ クラス間に直接関連付けおよび逆の関連付けを作成する手順を説明します。この例で管理者は、グループ クラスとユーザ アカウント クラスの関係について説明する関連付けを定義します。

ここでは、管理者が JNDI データ ソースをセットアップし、User account という名前のアカウント クラスを作成およびマッピングしたと仮定します。

アカウント クラスとグループ クラス間に直接関連付けおよび逆の関連付けを作成するには、グループ クラスとユーザ アカウント クラス間の関係を説明する関連付けを定義します。

次の手順に従ってください:

1. マッピング ツリーで、［Classes］ノードをクリックします。

   ［Mapped Classes］ダイアログ ボックスが表示されます。

2. ［Mapped Classes］ダイアログ ボックスの［追加］をクリックし、クラスの名前（例： Group of Names）を入力します。

   Connector Xpress によって、新しいクラスがマッピング ツリーに追加されます。

3. マッピング ツリーで、［Group of Names］ノードをクリックします。

   ［Map Class］ダイアログ ボックスが表示されます。

4. 構造クラス リストで、マッピングするネイティブ クラス（例: groupOfNames）を選択します。

5. マッピング ツリーで、［Attributes］ノードをクリックします。

   ［Map Attributes］ダイアログ ボックスが表示されます。

6. プロビジョニング属性にグループ名およびグループのメンバ属性をマッピングします。たとえば、プロビジョニング属性 objectname および member にネイティブ属性 *cn* および *member* をマッピングします。

7. メンバ属性の［Multivalued］チェック ボックスをオンにします。

   チェック ボックスをオンにすると、メンバ属性が複数値で複数のアカウント名を保持できることを指定します。

8. ［Group of Names］ノードの下の［Associations］ノードをクリックします。

   ［Class Associations］ダイアログ ボックスが表示されます。

9. ［*Create direct association with*］リストで、［User Account］クラスを選択します。

   Connector Xpress によって「*with User Account*」という名前のノードがマッピング ツリーに追加されます。

10. ［Group of Names］の下にある［Associations］ノードの下の［*with User Account*］ノードをクリックします。

    ［Direct Association with User Account］ダイアログ ボックスが表示されます。

    注: Connector Xpress によって、デフォルトでマッピングされる属性として、［Group of Names By Attribute］フィールドのグループの命名属性が選択されます。

11. ［Group of Names Attribute］で、メンバを選択します。

    メンバ属性を選択すると、アカウントの命名属性にグループ メンバ属性がマッピングされます。つまり、グループ メンバ属性がアカウント命名属性によって入力され、Group of names をグループ クラスとして定義したことになります。

12. ［Include a Reverse Association］チェック ボックスをオンにします。

    ［Reverse Association］ダイアログ ボックスが表示されます。

13. ［New Virtual Attribute］フィールドに「memberof」と入力します。

    この例では、ネイティブ アカウント クラスに memberOf 属性がないので、仮想 memberOf 属性を作成し、グループの命名属性にマッピングします。

    注: Connector Xpress によって、デフォルトでマッピングされる属性として、［By Attribute］フィールドのグループの命名属性が選択されます。

14. ［Project］ - ［Save］をクリックします。

    Connector Xpress によって、マップしたアカウントとグループのクラス間に直接関連付けおよび逆の関連付けが作成されます。

間接関連付けを作成する方法

この例では、メンバシップ テーブルを使用して、2 つのクラス間（従業員クラスおよび部門クラスなど）の間接関連付けを定義する手順について説明します。 間接関連付けを作成するには、以下の手順に従います。

1. プロジェクト (P. 29) を作成し、プロジェクトの JDBC データ ソース (P. 27) を指定します。
2. 従業員のリストを保持するクラスを作成しマッピングします。
3. 部門のリストを保持するクラスを作成しマッピングします。
4. 従業員クラスと部門クラスの間に間接関連付けを作成することを指定します。
5. 関連する個別の従業員と部門を識別する関連付けマッピングが含まれるメンバシップ テーブルを指定します。
6. 従業員と部門の間の関連を定義するメンバシップ テーブル列、および部門と従業員の間の関連を定義するメンバシップ テーブル列を指定します。
7. 従業員クラスに仮想 *memberof* 属性を作成して、従業員がメンバの部門のリストが含まれるメンバシップ テーブル列にマッピングします。
8. 部門クラスに仮想 *members* 属性を作成して、各部門の従業員のリストが含まれるメンバシップ テーブル列にマッピングします。
9. プロジェクトを保存します。

間接関連付けの例

この例では、管理者がメンバシップ テーブルを使用して Employees クラスと Departments クラス間に間接関連付けを作成する場合に従う手順について説明します。

この例では例スキーマ *HR* および例メンバシップ テーブル、*Membership* を使用します。メンバシップ テーブル列 *Account_Id* および *Dept_id* は、以下の例で示すように、関連する個別の従業員および部門を識別する関連付けマッピングが含まれます。

関連する個別の従業員と部門を識別するには、それらの間に間接関連付けを作成します。

注: ここでは、管理者が JDBC データ ソースをセットアップ (P. 27) し、Employees という名前のアカウント クラスを作成およびマッピングしたと仮定します。

次の手順に従ってください:

1. ［Classes］ノードをクリックします。

   ［Mapped Classes］ダイアログ ボックスが表示されます。

2. ［Add］をクリックし、クラスの名前を指定します（例： Departments）。

   Connector Xpress によって「Departments」という名前のノードがマッピング ツリーに追加されます。

3. マッピング ツリーの［Departments］ノードをクリックします。

   ［Map Class］ダイアログ ボックスが表示されます。

4. クラスの名前および説明を指定します。

   注: これらのフィールドは説明を目的としています。

5. マッピングするスキーマおよびテーブル（例： HR および Departments）を選択します。

6. マッピング ツリーで、［Attributes］ノードをクリックします。

   ［Map Attributes］ダイアログ ボックスが表示されます。

7. ［Name］列で、objectname 属性にアカウント命名属性をマッピングします。

8. ［Employees］ノードの下の［関連付け］ノードをクリックします。

   ［Class Associations］ダイアログ ボックスが表示されます。

9. Departments クラスとの間接関連付けを作成します。

   Connector Xpress によって、マッピング ツリーの Employees クラスの［Associations］ノードの下に「*with Departments*」という名前のノードが追加されます。

10. マッピング ツリーで、［Associations］ノードの下の［*with Departments*］ノードをクリックします。

    ［Indirect Association］ダイアログ ボックスが表示されます。

11. マッピングするクラスが含まれるスキーマを指定します（例： *HR*）。

12. ［Membership Table］リストで、*Membership* メンバシップ テーブルを選択します。

    このテーブルは、関連する個別の従業員および部門を識別する関連付けマッピングを指定します。

13. ［Employees Attribute］リストで、Employees クラスの命名属性を選択します。

14. ［Membership Table Columns］リストで、［AccountID］と［DeptID］をそれぞれ選択します。

    結果を以下に示します。

    - メンバシップ テーブル列［AccountID］に、Employees クラス命名属性（employee_id）をマッピングします。 これより、従業員と部門の間の関連付けが定義されます。
    - メンバシップ テーブル列［DeptId］に、Department クラス命名属性（Department_id）をマッピングします。 これより、部門と従業員の間の関連付けが定義されます。

    注: Connector Xpress ではデフォルトで、Employees および Departments Attributes リストからネーミング クラス属性を選択します。

15. ［Employees Attribute］および［Departments Attributes］フィールドに、それぞれ「*member*」および「*memberof*」と入力します。

    注: この例では、Employees クラスおよび Department クラスに *memberOf* または *member* 属性がないので、仮想の *member* および *memberOf* 属性を作成します。

    作成する仮想属性では、Employee クラスおよび Department クラスの間の関連付けについて説明します。

    これらの仮想属性は、従業員クラスおよび部門クラスの命名属性と手順 11 でマッピングしたメンバシップ テーブル列の関連付けを仮想表現したものです。 コネクタはこの仮想属性を使用して、部門で従業員を検索したり従業員が属する部門を検索したりできます。

    Connector Xpress によって以下が自動的に実行されます。

- マッピング ツリーの Employees および Department クラス ノードの下の［属性］ノードの下に *memberof* と *member* ノードを作成します。
- 自動的に、属性を複数値にし、デフォルトで［Provisioning Details］ダイアログ ボックスの複数値チェック ボックスをオンにします。
- 間接関連付けを自動的に作成し、Departments ノードの下の［関連付け］ノードの下に表示します。
- ［Employees］ノードの下に、Employees と Departments クラス間の関連付けを Departments クラスの視点から自動的に作成します。 そのため、Employees クラスの下の［Associations］ノードから表示された属性は、逆に表示されます。

16. ［Project］ - ［Save］をクリックします。

    Connector Xpress によって、マッピングした Employees と Departments クラス間の間接関連付けが作成されます。

コンテナ クラス

コンテナ クラスの指定は、マッピングできる属性が Container Name のみという点を除いて、通常のクラスをマッピングとほぼ同じです。

さらに、このコンテナ（［Contained Classes］）の子クラスを指定できます。 たとえば、コンテナ Employee Groups が個別のアカウント クラスではなく Staff Group および Executive Group クラスのみを許可できます。

入力検証

Connector Xpress では、以下のダイアログ ボックスのフィールドに入力したエントリが検証されます。

- エンドポイント タイプ
- Map Class
- Provisioning Attributes
- Defined Associations
- Indirect Associations

入力が無効なフィールドには隣に警告アイコンが表示され、マッピング ツリーの対応するノードの隣も警告アイコンが表示されます。警告アイコンにマウスを重ねると、違反の詳細が表示されます。入力を修正し、マッピング ツリーの別のノードをクリックと、警告アイコンが消えます。

Operation Bindings

操作バインディングは、その操作の処理に特化するため、特定の操作にバインドできるストアド プロシージャまたはスクリプトなどの追加のロジックです。操作に関連するストアド プロシージャによってロジックが呼び出されるタイミングを指定できます（操作の前後または代わりに）。

CA IAM CS 操作を実行すると、CA IAM CS では、その操作の前後または代わりにロジックの呼び出しを命じる操作バインディングがあるかどうかを確認します。

たとえば、ユーザをエンドポイントに追加するとします。 ユーザの名前および姓の情報があります。 しかし、エンドポイント システムのユーザのレコードでは、名前および姓の特定の組み合わせから構成された属性です。この状況を解決するには、エンドポイント形式に合わせて 2 つの名前を組み合わせるスクリプトを作成し、操作バインディングを使用して、エンドポイント システム上で実行する検索の前にこのスクリプトを実行することを指定します。

- ストアド プロシージャ（JDBC エンドポイントのみ）

  操作バインディングを使用して、Add、Modify、Delete および Rename 操作のストアド プロシージャを呼び出すことができます。

- スクリプト（JDBC と JNDI のエンドポイント）

  操作バインディングを使用して、Add、Modify、Delete および Rename 操作に加えて、Search、Lookup、Modify-Assocs、Delete-Assocs、Activate、Deactivate などの 16 追加操作のスクリプトを呼び出すことができます。

  この場合スクリプトは、グローバル スクリプトに格納された関数、またはスタンドアロン スクリプトレットと呼ばれます。他の操作バインディングに関数を再利用できるので、グローバル スクリプトに存在する関数を呼び出す方がベスト プラクティスです。

操作バインディングは、任意のオブジェクト クラスに適用できます。 たとえば、アカウント オブジェクトまたはグループ オブジェクトのいずれかへの変更操作をすべてログ ファイルに記録する場合、単一の操作バインディングを使用して、両方のオブジェクトに適用します。

単一のオブジェクトに複数の操作バインディングを同じタイミング（前後、または代わりに）で適用できます。 たとえば、特定の操作の前に 2 つのストアド プロシージャを呼び出すことができます。

注: Connector Xpress では、複合クラスに操作をバインドできません。

ストアド プロシージャ

ストアド プロシージャは JDBC エンドポイントにあります。 これは、CA IAM CS によって操作の前後または代わりに呼び出すことができるコードです。ストアド プロシージャは、そのエンドポイントに固有の言語で記述されたリレーショナル データベース エンドポイントにのみ関連します。

詳細情報:

ストアド プロシージャへのバインド操作 (P. 71)

スクリプト

スクリプトは Connector Xpress プロジェクト ファイルあり、JavaScript で記述されています。 ストアド プロシージャと同様に、CA IAM CS では、任意の CA Identity Manager 操作の前後、または代わりにスクリプトを呼び出すことができます。

グローバル スクリプトも個別のスクリプトも作成することができます。グローバル スクリプトには、操作バインディングの数を問わず、呼び出すことができる JavaScript 関数が含まれています。 グローバル スクリプトは共通関数を格納して再利用できる優れた方法です。

個別のスクリプトは、単一の操作バインディングによって使用される 1 つの JavaScript コードです。 個別のスクリプトは通常、特化した簡単な操作に使用されます。グローバル スクリプトで複数の関数をバインドする場合、選択された複数の関数を呼び出す個別のスクリプトを作成します。

詳細情報:

スクリプトへのバインド操作 (P. 73)

ストアド プロシージャへのバインド操作

操作を JDBC データベースのストアド プロシージャにバインドして、標準的なアカウント CRUD 操作の前後、または代わりに実行するアクションを指定できます。アクションには Add、Modify および Delete などがあります。選択する操作バインディングのタイプに応じて、他のタイプの操作も利用できます。

次の手順に従ってください:

1. Connector Xpress JDBC プロジェクトを作成します。
2. ユーザ アカウント クラス (P. 60)および他の必要なクラスをマッピングします。
3. ［Operations Bindings］ノードをクリックします。

   ［Operation Bindings Editor］が表示されます。
4. ［Object Classes Filtering］リストで、オブジェクト クラスを選択します。

   これで、操作バインディングを作成するオブジェクト クラスが指定されました。
5. ［Create］をクリックします。

   ［Create Operation Binding］ダイアログ ボックスが表示されます。
6. ［Available］オブジェクト クラス リストでクラスを選択し、［Added］オブジェクト クラス リストに追加します。

   これで、操作バインディングを適用するオブジェクト クラスが指定されました。
7. ［Type］から、［Stored Procedure］を選択します。
8. ［Available Operations］リストで、操作を選択します。

   これで、操作をバインディングする操作タイプが指定されました。
9. ［Timing］リストで、タイミングを選択します。

   これで、操作バインディングが実行されるタイミングが指定されました。
10. ［OK］をクリックします。

    ノードは、操作のタイプ、選択したタイミング、および操作バインディングが適用されるクラス名を表示するマッピング ツリーに追加されます。
11. 作成した操作バインディングに関する情報を表示するノードをクリックします。

    ［Stored Procedure］ダイアログ ボックスが表示されます。
12. ［Procedure］リストで、ストアド プロシージャを選択します。

    これで、手順 8 で指定した操作にプロシージャがバインドされました。

注: プロシージャ リストに入力されない場合、データベースでスキーマ メタデータのトランザクションまたはロックが残存していないことを確認します。

13. 必要に応じて、ストアド プロシージャの他の詳細を編集します。

    これで、ストアド プロシージャ スタイル操作のバインディング パラメータを指定できました。

14. プロジェクトを保存します。

詳細情報:

Operations Bindings Editor (P. 190)
Operation Bindings – Stored Procedure Editor (P. 191)

スクリプトへのバインド操作

操作をスクリプトにバインドして、Add、Modify、Delete などの標準的なアカウント CRUD 操作の前後、または代わりに実行するアクションを指定できます。 選択する操作バインディングのタイプに応じて、他のタイプの操作も利用できます。操作は、グローバル スクリプトの特定の関数にバインドすることも、個別のスクリプトにバインドすることもできます。 許可されている操作であれば、どのようなエンドポイント タイプ（JNDI や JDBC など）のスクリプトにもバインドできます。

次の手順に従ってください:

1. Connector Xpress JDBC または JNDI プロジェクトを作成します。
2. ユーザ アカウント クラス (P. 60)および他の必要なクラスをマッピングします。
3. ［Operations Bindings］ノードをクリックします。

   ［Operation Bindings Editor］が表示されます。
4. ［Object Classes Filtering］リストで、オブジェクト クラスを選択します。

   これで、スクリプト バインディングを作成するオブジェクト クラスが指定されました。
5. ［Create］をクリックします。

   ［Create Operation Binding］ダイアログ ボックスが表示されます。
6. ［Available］オブジェクト クラス リストでクラスを選択し、［Added］オブジェクト クラス リストに追加します。

   これで、スクリプト バインディングを適用するオブジェクト クラスが指定されました。
7. ［Type］から、［Script］を選択します。
8. ［Available Operations］リストで、操作を選択します。

   これで、スクリプトをバインディングする操作タイプが指定されました。
9. ［Timing］リストで、タイミングを選択します。

   これで、スクリプト バインディングが実行されるタイミングが指定されました。
10. ［OK］をクリックします。

    ノードは、操作のタイプ、選択したタイミング、およびスクリプト バインディングが適用されるクラス名を表示するマッピング ツリーに追加されます。
11. 作成したスクリプト バインディングに関する情報を表示するノードをクリックします。

    ［Script Editor］ダイアログ ボックスが表示されます。

12. グローバル スクリプトの関数を操作にバインドするには、以下の手順に従います。
    a. ［Execute a function in a global script］を選択します。
    b. ［Global Script］リストからグローバル スクリプトを選択します。
       グローバル スクリプトを選択することで、操作にバインドする関数があるスクリプトが指定されます。
    c. ［Function name］フィールドに、関数の名前を入力します。
       これで、操作にバインディングする関数が指定されました。
13. 操作を個別のスクリプトにバインドするには、以下の手順に従います。
    a. ［Execute an individual script］を選択します。
    b. ［Edit Script］をクリックします。
       ［Edit Script］ダイアログ ボックスが表示されます。
    c. 必要に応じて［Edit Script］ダイアログ ボックスにスクリプトをロードまたは貼り付けます。
    d. ［OK］をクリックします。
14. 必要に応じて、スクリプト バインディングの他の詳細を編集します。
    これで、スクリプト バインディングのスタイル操作バインディング パラメータを指定できました。
15. プロジェクトを保存します。

操作バインディングのインポート

別のプロジェクトで操作バインディングを作成し、それを再利用する場合や、バックアップから操作バインディングをリストアする場合、操作バインディングを XML ファイルからインポートできます。

操作バインディングをインポートする場合、Connector Xpress によって操作バインディングおよびストアド プロシージャへの参照がすべてインポートされます。

次の手順に従ってください:

1. ［Metadata］ - ［Import Operation Bindings］を選択します。
   ［Import Operation Bindings］ダイアログ ボックスが表示されます。
2. 操作バインディング XML ファイルに移動して、［Open］を選択します。
   インポートされた操作バインディングおよびストアド プロシージャへの関連スクリプトまたは参照はすべて、［Operation Bindings］ノードの下に表示されます。

操作バインディングのエクスポート

操作バインディングを別のプロジェクトで再利用する予定がある場合や単に操作バインディングをバックアップする場合、操作バインディングを XML ファイルにエクスポートできます。操作バインディングをエクスポートすると、Connector Xpress によって、すべての操作バインディングおよび任意のスクリプトまたは参照のコピーがストアドプロシージャに作成されます。本稼働エンドポイント タイプのコピーを作成すると、コピーには操作バインディングがすべて含まれます。

次の手順に従ってください:

1. ［Metadata］ - ［Export Operation Bindings］を選択します。

   ［Export Operations Bindings］ダイアログ ボックスが表示されます。

2. XML ファイルの保存先フォルダおよびエクスポートされた操作バインディングの名前を指定します。

   これで、操作バインディングが保存されます。

   操作バインディングをエクスポートすると、Connector Xpress によって、エクスポートされた XML ファイルの CDATA セクション内のスクリプトが自動的にカプセル化されます。"<" => "<" および "&" => "'" などの引用符に伴う XML の問題を考慮せずにエクスポートされた XML ファイルから容易に切り取りおよび貼り付けできます。

ストアド プロシージャと列の考慮事項

このセクションでは、ストアド プロシージャのスムーズな運用を保証ためのデータベース スキーマに関するいくつかの推奨事項について説明します。

ドライバによって共通の SQL タイプがレポートされない（ドライバによって java.sql.Type.OTHER = 1111 が返される）ストアド プロシージャ引数は、ドライバによって文字列が引数で求められているネイティブ タイプに変換できることを前提として、Type.VARCHAR (16) として扱われます。 たとえば、これは Oracle NVARCHAR2 の引数で機能します。これが機能しない場合、ストアド プロシージャは呼び出されず、「無効な列タイプ」で終了するエラー メッセージが表示されます。

ストアド プロシージャの引数には可能な限り、基本的なタイプ（VARCHAR に関連する基本的なタイプなど）を使用し、他の引数タイプを単一のストアド プロシージャ、使用するベンダーとバージョンで検証してから、広範囲で使用することをお勧めします。

グループ対アカウントの関連付けを作成する場合、キーに選択するネイティブ タイプのアカウント/グループ テーブル列が選択されたメンバシップ テーブルで対応する列のタイプに一致することを確認します。 タイプが一致しない場合、メンバシップ情報が正常に取得されません。できるだけ厳しい制約を設けることをお勧めします。たとえば、グループ命名属性として選択した列がタイプ NVARCHAR2 で、対応する列がメンバシップ テーブル内でタイプ VARCHAR2 の場合、アカウントが属するグループを検索しても空の（または、名前にマルチバイト文字が使用されるグループが欠落している）リストが返されます。

パーセント（%）およびアンダースコア（_）記号は、スキーマ、テーブル、テーブル列およびストアド プロシージャ引数のようなデータベース オブジェクトを検索する場合に、ワイルドカードとして機能するので、使用しないことをお勧めします。 この記号が表示される場合、引用符で囲われますが、この点はベンダーおよびバージョンによって異なります。たとえば、あるベンダーでは、一部リリースで正しく使用される引用符がレポートされません。当社の製品がサポートされているベンダーでは、%/_ に関する既知の問題は存在しません。

重要: Connector Xpress および CA IAM CS では、引数を使用してストアド プロシージャからデータが読み取り/書き込みされます。 ただし、この引数ではバインド先のストアド プロシージャのコードが妥当であるかどうかを確認できません。バインド先のストアド プロシージャのコードが妥当であるかどうかを確認することをお勧めします。

完全にスクリプト記述されたコネクタ

この SDK にバンドルされたフル機能の SDK スクリプト コネクタは、完全にスクリプト記述されたコネクタの一例です。

既製のコネクタを使用しない場合、Connector Xpress によって提供されるテンプレートを使用して完全にスクリプト記述されたコネクタを作成できます。以下の 2 つのテンプレートを使用して、完全にスクリプト記述されたコネクタを作成できます。

- SDK DYN Script
- SDK DYN UPO Script

このテンプレートをコネクタの基礎として使用して、必須操作のそれぞれでユーザ独自の JavaScript 関数を呼び出すことができます。 この機能は「代わりに」操作バインディングに依存します。 CA IAM CS では、操作が実行される前に操作の前後または代わりに、ロジックを呼び出すように命じる操作バインディングがあるかどうかが確認されます。完全にスクリプト記述されたユーザ独自のコネクタの JavaScript 操作を呼び出すには、「代わりに」操作バインディングを使用します。

Add、Delete、Modify、Search、および Lookup 必須操作それぞれに対して、「代わりに」操作バインディングを作成する必要があります。

完全にスクリプト記述されたコネクタを「ホット展開」できます。ホット展開とは、CA IAM CS を変更せずに、新しいコネクタを操作可能できるという意味です。

ホット展開したコネクタによって、メタデータ connectorXML の一部として connector.xml コンテンツが名前空間レベルで指定されます（この値に適切な XML エンコーディングに従う）。つまり、実行中の CA IAM CS を再起動したり、CA IAM CS ホストに静的な connector.xml を追加したりせずに、スクリプト コネクタをその場で作成できることを意味します。 また、通常は connector.xml で指定されるコネクタ設定（接続プール設定など）の変更もすべて、CA IAM CS を再起動せずにアクティブ化できます。

完全にスクリプト記述されたコネクタを作成する方法

テンプレートを基礎として使用すると、必須操作（Add、Delete、Modify、Search、および Lookup）それぞれに対して独自の JavaScript 関数が呼び出されるユーザ独自の完全にスクリプト記述されたコネクタを作成できます。以下の手順を実行します。

1. テンプレートからプロジェクトを作成します。（P. 35）

   完全にスクリプト記述された 2 つのコネクタ テンプレートは、SDK DYN Script および SDK DYN UPO Script です。

2. 属性をマッピングします。

   テンプレートには、出発点としていくつかの属性が用意されていますが、さらに追加することができます。

3. 必須関数それぞれに対して操作バインディングを作成します。（P. 73）

   テンプレートには 5 つの必須操作（Add、Delete、Modify、Search、Lookup）が含まれますが、さらに追加することができます。

4. JavaScript 操作を追加します。

   注: テンプレートには、基本的なスクリプトが用意されていますが、カスタマイズする必要があります。

5. マッピング ツリーの最上位で［Endpoint type］ノードをクリックします。

   ［Endpoint Type Details］ダイアログ ボックスが表示されます。

6. ［Load］をクリックし、ロードする connector.xml 設定を選択します。

   Connector Xpress によって、Connector XML フィールドに connector.xml 設定がロードされます。

7. ［Connector XML］フィールドで、以下の Connector XML 設定を設定します。

   a. 「name」の値をテンプレートのデフォルトから一意の名前に変更します。

   ```
   <Property name='name'>
   <Value>UniqueName</value>
   </Property>
   ```

   b. connectorTypeName の値をテンプレートのデフォルトから一意の名前に変更します。

   ```
   <Property name='connectorTypeName'>
   <Value>UniqueName</value>
   </Property>
   ```

   c. （オプション）Connector XML の他の設定アイテムを確認して、コネクタ設定を調整することをお勧めします。 以下に例を示します。

   ```
   <Property name='defaultConnectorConfig'>
   ```

注: 新しい完全にスクリプト記述されたコネクタのテンプレート名を維持すると、CA IAM CS で認識されます。 これで、完全にスクリプト記述されたコネクタを展開できます。 ただし、コネクタに新しい名前を付けることがベスト プラクティスですが、この場合、確実に CA IAM CS でコネクタが認識され、設定がロードされるようにするため、ホット展開する必要があります。 ホット展開の詳細については、「CA IM コネクタ サーバ プログラミング ガイド」を参照してください。

8. Connector Xpress プロジェクトを保存します。
9. コネクタを展開します。

ユーザ コンソールのアカウント画面の生成方法

CA Identity Manager では、メタデータ ベースの CA Identity Manager ユーザ コンソールのロールと画面の定義の生成がサポートされます。

CA Identity Manager ユーザ コンソールを使用して、特定の動的エンドポイント タイプのアカウント管理画面を作成できます。 アカウント管理画面では、アカウント、アカウント テンプレート、および特定のエンドポイント タイプのエンドポイントを管理できます。 アカウント管理画面を作成するには、以下の手順に従います。

1. Connector Xpress を使用して、ユーザ コンソール のアカウント管理画面のレイアウトを定義するプレゼンテーション メタデータを作成します。プレゼンテーション メタデータは、マップされた属性を論理的なグループおよびサブグループにグループ化することで作成できます。

   これらの属性は、アカウント管理画面でタブおよびページ セクションとして表示されます。Connector Xpress によって、作成されたグループがメタデータに保存されます。

   プレゼンテーション メタデータによって、以下が定義されます。

   - ユーザ コンソール アカウント管理画面で、入力コントロールが表示される場所
   - 入力コントロールのラベル

2. ロール定義ジェネレータを使用して、以下を生成します。

   - Connector Xpress で作成したプレゼンテーション メタデータからのフィールド、画面、タブ、タスク、および ロール定義。
   - ユーザ コンソールを介して特定のエンドポイント タイプでアカウント管理をするのに CA Identity Manager サーバによって要求されるファイル。

3. CA Identity Manager サーバによって要求される CA Identity Manager サーバ構成定義ファイルを展開します。

4. フィールド、画面、タブ、タスクおよびロール定義を CA Identity Manager にインポートします。

   フィールド、画面、タブ、タスクおよびロール定義をインポートすると、ユーザ コンソールでアカウント管理タスクを利用できるようになります。

ロール定義ジェネレータ

ロール定義ジェネレータとは、ユーザ コンソールから特定のエンドポイント タイプをアカウント管理するために CA Identity Manager サーバによって要求されるファイルを生成する、スタンドアロン ユーティリティです。

ロール定義ジェネレータは、CA Identity Manager サーバと一緒に以下のディレクトリにインストールされます。

- （Windows）%PROGRAMFILES%¥CA¥Identity Manager¥IAM Suite¥Identity Manager¥tools¥RoleDefinitionGenerator¥bin
- （UNIX）/opt/CA/Identity_Manager/IAM_Suite/Identity_Manager/tools/RoleDefinitionGenerator/bin

ロール定義ジェネレータのコマンド

Windows および UNIX システムで有効なコマンド

ロール定義ジェネレータのコマンドは、Connector Xpress から生成されたエンドポイント タイプ メタデータを解析し、*endpoint type*.jar を生成します。この JAR ファイルには JIAM マッピング ファイル、フレームワーク、管理対象オブジェクト定義ファイル、リソース バンドル ファイル、タスク ロールおよび画面定義ファイルが含まれます。

Windows でのこのコマンドの形式は、以下のようになります。

```
RoleDefGenerator.bat [-c jar_path] [d domain] -e fqn  -h hostname -l
-m filename -o directory -n -p port -u username -s -y
password_file.txt ] [endpoint_type ...]
```

UNIX でのこのコマンドの形式は、以下のようになります。

```
RoleDefGenerator.sh  [-c jar_path] [d domain] -e fqn  -h hostname -l
-m filename -o directory -n -p port -u username -s -y
password_file.txt ] [endpoint_type ...]
```

-c *jar_path*

JIAM 拡張 JAR ファイルを使用すると JAR がクラスパスに追加されることを指定します。

注: オプションですが、使用する場合は最初に指定してください。

-d *domain*

CA Identity Manager ドメインを指定します。 指定しない場合、ロール定義ジェネレータでは CA Identity Manager ドメインがデフォルトで使用されます。

-e *fqn*

使用されているメタデータに一致する JIAM オプション記述子クラスの完全修飾名を定義します。*-m* オプションと組み合わせて使用します。このエンドポイント タイプが含まれる JIAM 拡張 jar がクラスパスで利用可能である必要があります。

-h *hostname*

プロビジョニング サーバのホスト名を定義します。

-l

ロール定義ジェネレータによってロール定義が生成されずに、エンドポイント タイプがリスト表示されることを指定します。

-m filename

このファイルで指定されたメタデータがロール定義の生成に使用されることを指定します。

-o *directory*

出力ディレクトリを定義します。

デフォルト： '.'（現在の作業ディレクトリ）。

-n

指定すると、TLS は使用されません。 デフォルトでは、TLS 通信が有効に設定されています。

-p *port*

プロビジョニング サーバのポート番号を指定します。指定しない場合、20390 が使用されます。または、*-n* が指定される場合、20389 が使用されます。

-u *username*

プロビジョニング サーバの管理者ユーザ名を定義します。

-s

スタンドアロン CA IAM CS モードで実行されます。

-y *password_file.txt*

プロビジョニング サーバの管理者ユーザ パスワードが含まれるファイルを指定します。 指定しない場合、ユーティリティによってパスワードが要求されます。パスワード ファイルは UTF-8 形式です。このファイルの最初の行がパスワードとして使用されます。

endpoint_type

エンドポイント タイプの名前（長い形式）を定義します。

例: プロビジョニング サーバのエンドポイント タイプをすべてリスト表示する

この例では、プロビジョニング サーバのエンドポイント タイプがすべてリスト表示されます。

```
RoleDefGenerator.bat -d EXAMPLEDOMAIN -h im.example.com -u adminusername -l
```

例: 動的エンドポイント タイプのロール定義を生成する

この例では、*YourDynamicEndpointType* のロール定義を生成します。

```
RoleDefGenerator.bat -d EXAMPLEDOMAIN -h im.example.com -u adminusername
YourDynamicEndpointType
```

アカウント画面作成の例

この例では、シンプルな JNDI コネクタのユーザ コンソールのアカウント管理でタブとページ セクションを定義するプレゼンテーション メタデータを作成する方法を説明します。 この例では、MyJNDIEndpointType という名前の動的エンドポイント タイプのアカウント管理画面を作成します。

アカウント画面の生成方法

動的エンドポイント タイプ MyJNDIEndpointType のアカウント管理画面を生成するには、以下の手順に従います

1. Connector Xpress を使用して、MyJNDIEndpointType を記述するプロジェクトを作成します。
2. マップされた属性をアカウント管理画面内でタブおよびページ セクションとしてされる論理的なグループおよびサブグループにグループ化して、Connector Xpress でプレゼンテーション メタデータを作成します。（P. 85）
3. Connector Xpress を使用してプロビジョニング サーバにコネクタを展開します。（P. 44）
4. ロール定義ジェネレータを使用して、MyJNDIEndpointType.jar ファイルを生成します。

   MyJNDIEndpointType でアカウント管理するために、このファイルが CA Identity Manager サーバによって要求されます。
5. CA Identity Manager サーバに MyJNDIEndpointType.jar ファイルを展開します。
6. 特定のエンドポイント タイプのアカウントを管理する CA Identity Manager 環境にロールとタスクの定義をインポートします。

プレゼンテーション メタデータの例

以下の例では、MyJNDIEndpointType に対してマッピングした属性をユーザ コンソールのアカウント管理画面でタブおよびページ セクションとして表示する論理的なグループおよびサブグループにグループ化する方法を説明します。

この例では、ユーザが以下のタスクを実行したことを前提としています。

1. コネクタのプロジェクトを作成し、CA Directory などの JNDI データ ソースを指定した。（P. 34）
2. User Account クラスを作成し、プロビジョニング属性 *cn* および *sn* に *inetOrgPerson* の Account ID および Last Name 属性をそれぞれマッピングした。（P. 60）
3. Group クラスを作成し、プロビジョニング属性 *cn* および *member* に *groupOfNames* の Account ID および Member 属性をそれぞれマッピングした。（P. 59）
4. Group クラスと User Account クラスの間に直接関連付けおよび逆の関連付けを作成し、アカウントのクラス命名属性にグループの member 属性をマッピングした。（P. 62）

例: プレゼンテーション メタデータの作成

マッピングした属性をユーザ コンソールのアカウント管理画面でタブおよびページ セクションとして表示する論理的なグループおよびサブグループにグループ化するには、Connector Xpress を使用してプレゼンテーション メタデータを作成します。

次の手順に従ってください:

1. マッピング ツリーで［User Account］ノードの下の［Accounts screen］ノードをクリックします。

   ［Account Screens］ダイアログ ボックスが表示されます。

2. ［Account］ボタンをクリックします。

   ［Login］ページ セクションが表示されます。

3. ［Login］ページ セクションのドロップダウン リストから［Account Id］属性を選択します。

4. ［User］ボタンをクリックします。

   ［Name］ページ セクションが表示されます。

5. ［Name］ページ セクションのドロップダウン リストから［Last Name］属性を選択します。

6. ［Membership］ボタンをクリックします。

   ［Membership］ページ セクションが表示されます。

7. ドロップダウン リストから［Member］属性を選択します。

8. プロビジョニング サーバに MyJNDIEndpointType コネクタを展開し、プロジェクトを保存します。

9. 次に、ロール定義ジェネレータを使用して、プレゼンテーション メタデータを CA Identity Manager によって要求されるファイルに変換します。

ロール、タスク、および画面定義ファイルの例

Connector Xpress で作成したプレゼンテーション メタデータからロール定義ジェネレータを使用して、フィールド、画面、タブ、タスク、およびロール定義を生成し、ユーザ コンソールから特定のエンドポイント タイプのアカウントを管理するために、CA Identity Manager サーバによって要求されるファイルを生成します。

例: ロール、タスクおよび画面定義ファイルの生成

MyJNDIEndpointType でアカウント管理画面を提供するため、プレゼンテーション メタデータを CA Identity Manager によって要求されるファイルに変換するには、ロール定義ジェネレータを使用します。

Windows および UNIX システムで有効なコマンド

次の手順に従ってください:

1. オペレーティング システムに応じて以下のいずれかのディレクトリに移動します。
   - （Windows） %PROGRAMFILES%¥CA¥IAM Suite¥Identity Manager¥tools¥RoleDefinitionGenerator¥bin
   - （UNIX） /opt/CA/IAM_Suite/Identity_Manager/tools/RoleDefinitionGenerator/bin
2. お使いのオペレーティング システムによってコマンド プロンプト ウィンドウまたはターミナル ウィンドウを開き、以下のいずれかのコマンドを入力します。
   - （Windows） RoleDefGenerator.bat -d *exampledomain* –h *im.exmaple.com* –u *adminusername* MyJNDIEndpointType
   - （UNIX） RoleDefGenerator.sh -d *exampledomain* –h *im.exmaple.com* –u *adminusername* MyJNDIEndpointType

   コマンドによって MyJNDIEndpointType.jar ファイルが生成されます。

   メタデータから生成されたロール、タスク、および画面定義には、指定したエンドポイント タイプの基本的な管理者ロールおよびオーディター ロール（読み取り専用）が含まれます。
3. 次に、CA Identity Manager サーバに MYJNDI.EndpointType.jar ファイルを展開します (P. 88)。

CA Identity Manager サーバ構成定義ファイルの展開例

以下の例では、CA Identity Manager サーバに CA Identity Manager サーバ構成定義ファイルを展開する方法について説明します。

例: CA Identity Manager サーバ構成定義ファイルを展開する

MyJNDIEndpointType でアカウント管理するには、CA Identity Manager サーバに MyJNDIEndpointType.jar ファイルを展開します。

Windows および UNIX システムで有効なコマンド

次の手順に従ってください:

1. MyJNDIEndpointType.jar を以下のいずれかのディレクトリにコピーします。
   - （Windows） *app server home*/iam_im.ear/user_console.war/WEB-INF/lib
   - （UNIX） *app server home*¥iam_im.ear¥user_console.war¥WEB-INF¥lib

     注: WebSphere の場合、以下の場所に JAR ファイルをコピーします。

     WebSphere_home/AppServer/profiles/*Profile_Name*/config/cells/*Cell_name*/applications/iam_im.ear/user_console.war/WEB-INF
2. クラスタがある場合は、ノードごとに上記 2 つの手順を繰り返します。
3. CA Identity Manager サーバを再起動します。
4. CA Identity Manager 環境にロールとタスクの設定をインポートします。

例: CA Identity Manager にロールおよびタスクの設定をインポートする

以下の例では、ロール定義ジェネレータによって生成されたロールとタスクの設定を CA Identity Manager にインポートする方法について説明します。

MyJNDIEndpointType でアカウント管理するには、CA Identity Manager にロール定義ジェネレータによって生成されたロールとタスクの設定を CA Identity Manager にインポートします。

次の手順に従ってください:

1. CA Identity Manager 管理コンソールで、［Environments］をクリックします。
2. 指定されたエンドポイント タイプでアカウントを管理する環境を選択します。
3. ［Role and Task Settings］をクリックします。
4. ［Import］をクリックします。
5. 画面、ロール、タスク定義をインポートするエンドポイント タイプを選択して、［Finish］をクリックます。

   ［Role Configuration Output］ウィンドウにステータスが表示されます。
6. CA Identity Manager 環境を再起動します。

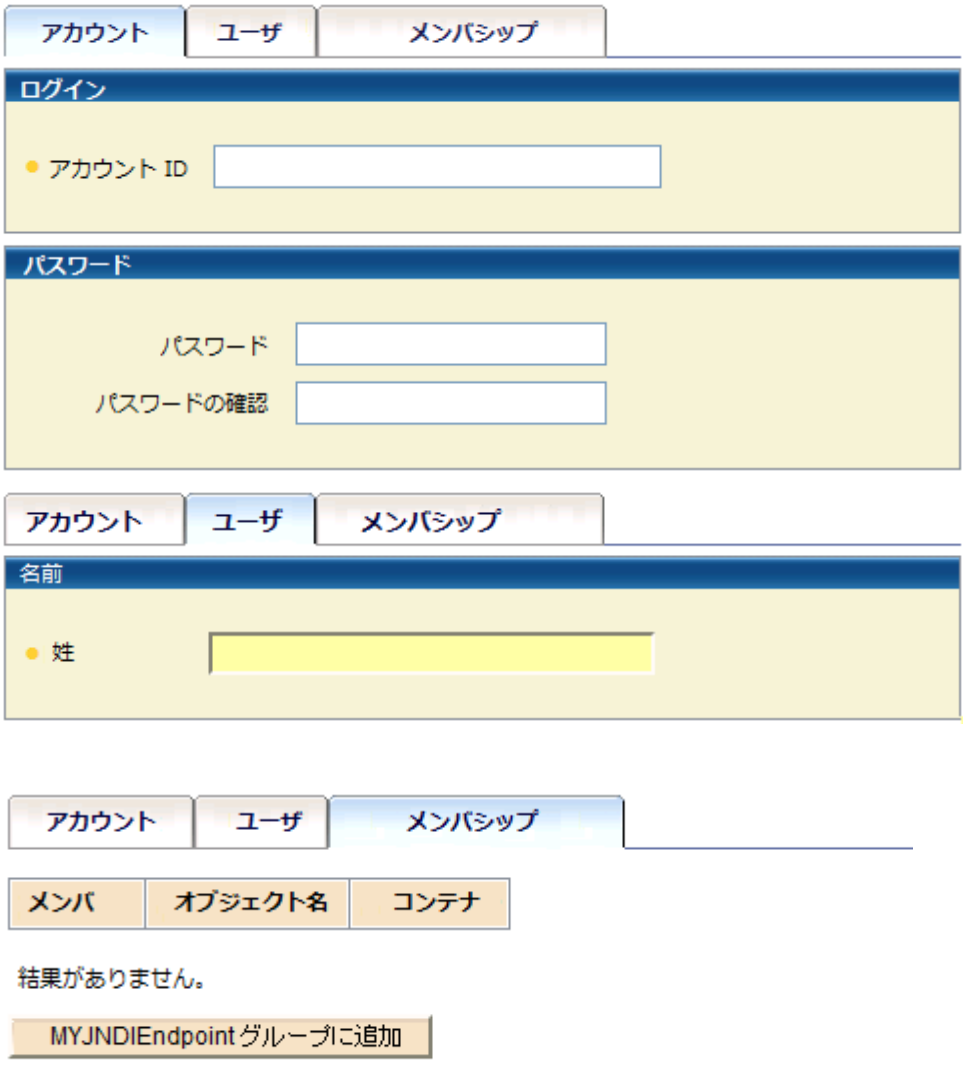
   MyJNDIEndpointType のアカウント管理画面は、エンドポイントのアカウントを作成、変更するなどのアカウント管理タスクを実行する場合にユーザ コンソールから利用できます。
7. システム マネージャ管理ロールのメンバでないユーザを特定します。
8. CA Identity Manager で、システム マネージャ管理ロール メンバシップのメンバではないユーザに、特定のエンドポイント タイプの新しく作成されたオーディター ロールまたはマネージャ管理ロールを付与します。

   これにより、ユーザが Account タスクおよび［Accounts］タブにアクセスできるようになります。

   システム マネージャ管理ロールのメンバには、［Modify User's Accounts］および［View User's Account］の管理タスクで新しい［Accounts］タブが自動的に表示されます。

例: 生成されたアカウント画面

この例では、CA Identity Manager にロールとタスクの定義をインポートした、アカウント管理タスクのアカウント管理画面を示します。



エンドポイント タイプのロール定義を展開解除する

以前にロール定義をインポートした CA Identity Manager 環境から、指定したエンドポイント タイプのロール定義を展開解除できます

次の手順に従ってください:

1. *iam_im.ear*¥user_console.war¥WEB-INF¥lib フォルダからエンドポイント タイプ固有の .jar ファイルを削除します。
2. CA Identity Manager サーバを再起動します。

エンドポイント タイプが CA Identity Manager サーバから登録解除され、CA Identity Manager ユーザ コンソールに表示されなくなります。 CA Identity Manager ユーザ コンソールでは今後、エンドポイント タイプのアカウントまたはアカウント テンプレートを管理できません。エンドポイント タイプに固有の .jar ファイルを削除しても、プロビジョニング サーバ側にあるこのエンドポイント タイプのアカウント テンプレート、エンドポイントなどのオブジェクトには影響しません。

テスト環境から実稼働環境にコネクタをプロモートする例

この例では、Forwardinc で Oracle データベース テーブルをプロビジョニング アカウントおよびグループにマッピングする新しい Oracle のコネクタの作成およびテストが完了し、 このコネクタをテストおよびステージング プロビジョニング サーバ（forwardinc_Test_PS）に展開しました。 現在、新しい実稼働プロビジョニング サーバ（forwardinc_Prod_PS）にコネクタを展開して、コネクタを展開した後にテスト環境からコネクタを削除しようとしています。

例: Forwardinc でテスト環境から実稼働環境にコネクタをプロモートする

以下の例では、Forwardinc の CA Identity Manager インストール管理者がテスト環境から新しい実稼働環境にコネクタをプロモートするプロセスについて説明します。 管理者：

1. Oracle のコネクタを ORA_Connector.con など、Connector Xpress プロジェクトとして保存します。
2. 新しい実稼働プロビジョニング サーバ、forwardinc_Prod_PS を設定します。

   Forwardinc がコネクタを展開する実稼働プロビジョニング サーバの接続詳細は指定されています。
3. Connector Xpress プロジェクト ORA_Connector.con を開きます。
4. forwardinc_Prod_PS プロビジョニング サーバにエンドポイント タイプを作成します。
5. 新しいエンドポイント タイプを管理する管理コネクタ サーバを指定します。

   管理コネクタ サーバを指定すると、エンドポイント タイプに対する要求が確実にルーティングされます。
6. 新しいエンドポイント タイプを検索および関連付けします。

   検索と関連付けにより、エンドポイントで検出されたデータが forwardinc_Prod_PS プロビジョニング サーバに入力されます。
7. テスト プロビジョニング サーバ、forwardinc_Test_PS から古いエンドポイント タイプを削除します。

   エンドポイント タイプ、関連するアカウント テンプレート、およびすべての子エンドポイントが削除されますが、エンドポイントのデータは変更されません。

新しいメインフレーム コネクタ用のカスタム属性のマッピング

このセクションは、以下のメインフレーム コネクタを対象としています。

- CA ACF2 v2
- CA Top Secret v2
- RACF v2

文字列で表されたメインフレーム エンドポイント上の LDAP 属性は、コネクタ内でカスタム属性として公開される場合があります。 これらのカスタム属性を、Connector Xpress でマップします。

次の手順に従ってください:

1. プロビジョニング サーバまたはコネクタ サーバを Connector Xpress に追加します。

- メインフレーム エンドポイント タイプ（CA ACF2 v2、CA Top Secretv2 または RACF v2）からプロジェクトを作成します。

1. 新しいプロジェクト内で、［クラス］-［ユーザ アカウント］-［属性］に移動し、新しい属性を追加します。

   新しい属性には以下のタイプがありえます。

   - String
   - Integer
   - Boolean
   - Date
   - Time

2. 各カスタム属性に以下の情報を入力します。

   名前

   属性を識別します。［ユーザ コンソール］ページに表示されます。

   マップ先

   エンドポイントの CA LDAP サーバに表示される、属性の LDAP 名が含まれます。

3. ブール属性については、以下のプロパティを編集します。

   Value Conversion

   ［Symbolic］を選択します。

   Boolean Values

   「Y」および「N」の文字列を「True」と「False」で置換します。

4. Date 属性については、値の形式が、エンドポイント上の一致する属性と同じ形式になるようにしてください。
5. カスタムの Date 属性については、String タイプを使用します。

第 4 章: Connector Xpress のユーティリティ

このセクションには、以下のトピックが含まれています。

Connector Xpress のユーティリティを使用して実行できる操作

Connector Xpress のユーティリティは以下に使用できます。

- プロビジョニング サーバ設定のセットアップ (P. 95)
- コネクタ サーバ設定のセットアップ (P. 98)
- 別の方法を使用して、コネクタ サーバとエンドポイントを関連付ける (P. 99)
- CA IAM CS のパスワードを更新する (P. 101)
- エンドポイントのマッピング概要の HTML ファイルをエクスポートする (P. 107)
- 動的エンドポイントを追加、入力、消去、削除する (P. 110)
- メタデータを編集および展開する (P. 117)
- エンドポイント タイプの管理 CS を設定する (P. 99)
- エンドポイント システムを取得、検索、関連付け、削除する (P. 112)

プロビジョニング サーバの設定

Connector Xpress では、プロビジョニング サーバを追加、編集、削除してサーバのコネクタを管理します。

プロビジョニング サーバの追加および設定

サーバのコネクタを管理するには、プロビジョニング サーバを追加および設定します。

次の手順に従ってください:

1. プロビジョニング サーバ ツリーで、［Provisioning Servers］ノードを右クリックし、［Add Server］をクリックします。

   ［Provisioning Server Details］ダイアログ ボックスが表示されます。

2. ダイアログ ボックスのフィールドに入力して、［OK］をクリックします。

   Connector Xpress によって、プロビジョニング サーバ ツリーにプロビジョニング サーバを追加します。

   注: プロビジョニング サーバにすでにコネクタ サーバが設定 (P. 98)されている場合、プロビジョニング サーバ ノードの下に表示されます。

詳細情報:

［Provisioning Server Details］ダイアログ ボックス (P. 198)

プロビジョニング サーバ詳細の編集

ハードウェアをアップグレードする場合など、プロビジョニング サーバの詳細が変更された場合は、必要に応じてプロビジョニング サーバの詳細を編集します。

次の手順に従ってください:

1. プロビジョニング サーバ ツリーで詳細を編集するサーバを右クリックし、［Edit Server］をクリックします。

   ［Provisioning Server Details］ダイアログ ボックスが表示されます。

2. ダイアログ ボックスのフィールドに入力して、［OK］をクリックします。

   Connector Xpress によって、プロビジョニング サーバの詳細が変更されます。

詳細情報:

［Provisioning Server Details］ダイアログ ボックス (P. 198)

プロビジョニング サーバの削除

ハードウェアを取り外したり、交換したりする場合など、プロビジョニング サーバの詳細が変更された場合は、必要に応じてプロビジョニング サーバの詳細を削除します。

次の手順に従ってください:

1. プロビジョニング サーバ ツリーで削除するサーバを右クリックし、［Remove Server］をクリックします。
2. プロンプトが表示されたら、プロビジョニング サーバを削除することを確認します。

   Connector Xpress によってプロビジョニング サーバが削除されます。

プロビジョニング サーバへの接続

プロビジョニング サーバに接続するには、プロビジョニング サーバのパスワードを提供します（必要な場合）。

次の手順に従ってください:

1. プロビジョニング サーバ ツリーを展開し、接続するプロビジョニング サーバをダブルクリックします。

   ［Provisioning Server password］ダイアログ ボックスが表示されます。
2. パスワードを入力して、［OK］をクリックします。

   注: プロビジョニング サーバに接続する際、ユーザが無効なパスワードを入力すると、接続に失敗します。また、Connector Xpress で、プロビジョニング サーバ ツリーにエラー ノードが表示されます。別のパスワードで再接続するには、プロビジョニング サーバ ツリーをリフレッシュします。

詳細情報:

［Provisioning Server Password Required］ダイアログ ボックス (P. 199)

コネクタ サーバの設定

コネクタ サーバの設定により、プロビジョニング サーバがそれぞれのエンドポイントをどのようにコネクタ サーバにルーティングするかが決定されます。

複数の設定を作成し、指定されたエンドポイントに適切な設定を関連付けることができます。

注: コネクタを展開する前にエンドポイントのコネクタ サーバ設定を作成 (P. 98)します。

コネクタ サーバ設定の作成

プロビジョニング サーバがコネクタ サーバにエンドポイント タイプまたはそれぞれのエンドポイントをどのようにルーティングするかを指定するには、コネクタ サーバ設定を作成します。

次の手順に従ってください:

1. コネクタを展開したプロビジョニング サーバ ツリーのサーバを展開します。
2. ［CS Configs］ノードを右クリックし、［New CS Config］を選択します。

   ［Connector Server Configuration］ダイアログ ボックスが表示されます。
3. ダイアログ ボックスのフィールドに入力して、［OK］をクリックします。

   これで、プロビジョニング サーバがコネクタ サーバにエンドポイント タイプまたはそれぞれのエンドポイントをどのようにルーティングするかが指定されます。

詳細情報:

［Edit Connector Server Configuration］ダイアログ ボックス (P. 158)

管理コネクタ サーバを設定する方法

特定のエンドポイント タイプまたはエンドポイントを管理するサーバを選択できます。 代替プロビジョニング サーバを設定し、プロビジョニング サーバに個別の CS がある場合、複数のサーバにエンドポイント タイプを 1 つ関連付けることができます。

以下の方法のいずれかで、エンドポイントに CA IAM CS を関連付けることができます。

- エンドポイント タイプ ノードから（P. 99）
- ［CS Configs］ノードの CA IAM CS から（P. 100）

［Endpoint Type］ノードから管理コネクタ サーバを設定する

［Endpoint Type］ノードから特定のエンドポイント タイプまたはエンドポイントを管理するコネクタ サーバを選択できます。

次の手順に従ってください:

1. プロビジョニング サーバ ツリーでエンドポイント タイプを管理するサーバを右クリックし、［Set Managing CS］をクリックします。

   ［Select Connector Servers］ダイアログ ボックスが表示されます。

2. コネクタ サーバを選択して、［OK］をクリックします。

   これで、管理するコネクタ サーバが指定されました。Connector Xpress によって、選択されたコネクタ サーバの管理されたブランチにエンドポイント タイプまたはエンドポイントが追加されます。

CS 設定ノードから管理コネクタ サーバを設定する

［CS Configs］ノードから特定のエンドポイント タイプまたはエンドポイントを管理するコネクタ サーバを選択できます。

次の手順に従ってください:

1. ［CS Configs］ノードを展開します。
2. エンドポイントと関連付ける CS Config を右クリックし、［Edit CS Configuration］をクリックします。

   ［Edit Connector Server Configuration］ダイアログ ボックスが表示されます。
3. ダイアログ ボックスの以下のフィールドに入力して、［OK］をクリックします。

   Object Handle

   選択されたプロビジョニング サーバで管理されているエンドポイント タイプのオブジェクト ハンドルを定義します。

   例： Namespace=dyn_jndi_democorp,Domain=forward,Server=Server

コネクタ サーバ設定の編集

プロビジョニング サーバがコネクタ サーバにエンドポイント タイプをどのようにルーティングするかを変更するには、コネクタ サーバ設定を編集します。

次の手順に従ってください:

1. コネクタを展開したプロビジョニング サーバ ツリーのサーバを展開します。
2. ［CS Configs］ノードを展開します。
3. 編集するサーバ設定を右クリックし、［Edit CS Config］を選択します。

   ［Connector Server Configuration］ダイアログ ボックスが表示されます。
4. ダイアログ ボックスのフィールドに入力して、［OK］をクリックします。

   これで、プロビジョニング サーバがコネクタ サーバにエンドポイント タイプまたはそれぞれのエンドポイントをどのようにルーティングするかが変更されます。

詳細情報:

［Edit Connector Server Configuration］ダイアログ ボックス (P. 158)

コネクタ サーバ設定の削除

コネクタ サーバ設定を削除するには、プロビジョニング サーバ ツリーから削除します。

次の手順に従ってください:

1. コネクタを展開したプロビジョニング サーバ ツリーのサーバを展開します。
2. ［CS Configs］ノードを展開します。
3. 削除するサーバ設定を右クリックし、［Edit CS Config］を選択します。
4. プロンプトが表示されたら、コネクタ サーバ設定を削除することを確認します。

   コネクタ サーバ設定が削除されます。

CA IAM CS のパスワードを更新する

優れたセキュリティ プラクティスを保証するには、CA IAM CS パスワードを定期的に更新します。パスワードを更新することで、既知のプロビジョニング サーバ セットのみで CA IAM CS が使用されるようにできます。 たとえば、パスワードを更新した後、新しいパスワードを持った CA IAM CS を使用するべきプロビジョニング サーバのみを更新します。 CA IAM CS にアクセス権がないプロビジョニング サーバは、システムを再起動すると CA IAM CS を使用できなくなります。

次の手順に従ってください:

1. プロビジョニング サーバ ツリーで［CS Configs］ノードを展開します。
2. 編集する CS Config を右クリックし、［Edit CS Config］をクリックします。

   ［Edit Connector Server Configuration］ダイアログ ボックスが表示されます。
3. ダイアログ ボックスのフィールドに入力して、［OK］をクリックします。

詳細情報:

［Edit Connector Server Configuration］ダイアログ ボックス (P. 158)

Connector Xpress で CA IAM CS を管理する

CCS を持った CA IAM CS をインストールした場合、コネクタ サーバの管理に Connector Xpress を使用できます。 コネクタ サーバは、追加、編集、削除できます。

コネクタ サーバを設定すると、CA IAM CS を介してアクセスされるコネクタを管理できます。

次の手順に従ってください:

1. プロビジョニング サーバ ツリーで、［Connector Servers］ノードを右クリックし、［Add Connector Server］をクリックします。

   ［Connector Sever Details］ダイアログ ボックスが表示されます。

2. ［Connector Sever Details］ダイアログ ボックスの以下のフィールドに入力して、［OK］をクリックします。

   Host Name

   コネクタ サーバが存在するホストの名前を定義します。

   Port

   CA IAM CS への接続に使用されるポートを定義します

   Bind DN

   DN 形式でコネクタ サーバを接続するユーザ名を定義します。

   Use TLS

   コネクタ サーバに安全な TLS 接続が確立されることを指定します。つまり、パスワードはクリア テキスト形式で転送されません。 選択しないと、パスワードはクリア テキスト形式で転送されます。

   デフォルト： オン

   Key

   コネクタがコネクタ サーバとの接続に使用する設定オプションの名前を定義します。 例：
   com.ca.iam.server.class=com.ca.iam.model.impl.JCSServer および
   com.ca.iam.server.superagent=true

   Value

   設定オプションの値を定義します。

   ［Key］列

コネクタがコネクタ サーバとの接続に使用する設定オプションの完全なリストを定義します。

［Value］列

［Key］列で表示された設定オプションに対応する値を表示します。

Connector Xpress によって、既知のコネクタ サーバ リストがある、展開されたコネクタ サーバ ツリーが表示されます。

コネクタ サーバの設定

Connector Xpress を使用して、CA IAM CS を介してアクセスされるコネクタを管理するコネクタ サーバを設定できます。 CA IAM CS を設定するには、コネクタ サーバがインストールおよび実行されていることを確認します。

次の手順に従ってください:

1. プロビジョニング サーバ ツリーで、［Connector Server］ノードを右クリックし、［Add Connector Server］をクリックします。

   ［Connector Sever Details］ダイアログ ボックスが表示されます。

2. ［Connector Sever Details］ダイアログ ボックスの以下のフィールドに入力して、［OK］をクリックします。

   Host Name

   コネクタ サーバが存在するホストの名前を定義します。

   Port

   CA IAM CS への接続に使用されるポートを定義します

   Bind DN

   DN 形式でコネクタ サーバを接続するユーザ名を定義します。

   Use TLS

   コネクタ サーバに安全な TLS 接続が確立されることを指定します。つまり、パスワードはクリア テキスト形式で転送されません。 選択しないと、パスワードはクリア テキスト形式で転送されます。

   デフォルト： オン

   Key

   コネクタがコネクタ サーバとの接続に使用する設定オプションの名前を定義します。 例：
   com.ca.iam.server.class=com.ca.iam.model.impl.JCSServer および
   com.ca.iam.server.superagent=true

   Value

   設定オプションの値を定義します。

   ［Key］列

   コネクタがコネクタ サーバとの接続に使用する設定オプションの完全なリストを定義します。

   ［Value］列

［Key］列で表示された設定オプションに対応する値を表示します。

Connector Xpress によって、既知のコネクタ サーバ リストがある、展開されたコネクタ サーバ ツリーが表示されます。

コネクタ サーバへの接続

Connector Xpress を使用して、コネクタ サーバに接続し、コネクタ サーバによって管理されるエンドポイントを表示および管理できます。

次の手順に従ってください:

1. プロビジョニング ツリーで、接続するコネクタ サーバをクリックします。
2. プロンプトが表示されたら、接続するコネクタ サーバのパスワードを入力し、［OK］をクリックします。

   Connector Xpress がコネクタ サーバに接続され、サーバがスタンドアロン コネクタ サーバかどうかがクエリされます。

   コネクタ サーバから、スタンドアロン コネクタ サーバであると確認で応答します。

   Connector Xpress に、［Connector Servers］ノードの下の［End Points］ノードが表示されます。
3. プロビジョニング ツリーで、接続するコネクタ サーバの下にある［End Points Type］ノードを選択します。

   Connector Xpress によって、選択されたコネクタ サーバによって管理される CA IAM CS、CSS エンドポイント タイプ、エンドポイント、および動的エンドポイント タイプが表示されます。

コネクタ サーバ詳細の編集

ハードウェアをアップグレードする場合など、コネクタ サーバの詳細が変更された場合は、必要に応じてコネクタ サーバの詳細を編集します。

次の手順に従ってください:

1. プロビジョニング サーバ ツリーで［Connector Servers］ノードの下の詳細を編集するコネクタ サーバを右クリックし、［Edit Server］をクリックします。

   ［Connector Server Details］ダイアログ ボックスが表示されます。
2. ダイアログ ボックスのフィールドに入力して、［OK］をクリックします。

   Connector Xpress によって、コネクタ サーバの詳細が変更されます。

詳細情報:

［Connector Server Details］ダイアログ ボックス (P. 148)

コネクタ サーバによって管理されたエンドポイント タイプの作成

特定のコネクタ サーバで、エンドポイント タイプを作成および管理できます。

注: エンドポイント タイプは、Connector Xpress プロジェクトが開いている場合にのみ作成できます。

次の手順に従ってください:

1. 既存のプロジェクトを開きます (P. 39)。
2. プロビジョニング サーバ ツリーで［Connector Servers］ノードの下の［Endpoint Types］ノードを右クリックし、［Create New Endpoint Type］をクリックします。

   ［Create New Endpoint Type］ダイアログ ボックスが表示されます。
3. ダイアログ ボックスのフィールドに入力して、［OK］をクリックします。

   これで、新しいエンドポイント タイプの名前が定義され、指定したコネクタ サーバの［Endpoint Types］ノードにこのエンドポイント タイプが追加されました。

コネクタ サーバによって管理されたエンドポイント タイプの削除

コネクタ サーバによって管理されたコネクタの管理がプロビジョニング マネージャで不要になった場合、Connector Xpress を使用して、コネクタを削除できます。

重要: 動的コネクタに関連付けられたエンドポイントを削除する場合、正常に削除できるよう、Connector Xpress を使用することを強くお勧めします。

次の手順に従ってください:

1. プロビジョニング サーバ ツリーのコネクタ サーバ ノードの下にある［Endpoints］ノードを展開します。
2. 削除するエンドポイントを右クリックし、［Delete Endpoint］をクリックします。
3. プロンプトが表示されたら、エンドポイントを削除することを確認します。

Connector Xpress によってコネクタが削除され、エンドポイントと関連アカウント テンプレートが除去されますが、エンドポイントのデータは変更されません。

マッピング概要ファイル

デバッグおよびログの確認にマッピング概要ファイルを使用すると便利です。たとえば、汎用スキーマの属性名が eTDYN-str-i-01 のように記述的でない場合、マッピング概要ファイルを使用できます。

また、自動化するためのバッチ スクリプトをユーザが記述する場合に、マッピング概要ファイルを生成することが重要です。

マッピング概要ファイルは、選択したエンドポイント タイプまたは現在開いているプロジェクトに関する以下の情報を表示する HTML ファイルです。

- プロビジョニング LDAP 属性名
- 論理名
- ネイティブ名
- 表示名
- 動的エンドポイントで定義された各クラスの属性値

マッピング概要ファイルのエクスポート

デバッグおよびログの確認にマッピング概要ファイルをエクスポートすると便利です。たとえば、プロビジョニング LDAP 属性名とネイティブ属性名/列名の間のエンドポイント マッピングを調べるためにファイルを使用します。

ロードされているプロジェクトまたはプロビジョニング サーバ ツリーで選択されたエンドポイント タイプのマッピング ファイルをエクスポートできます。

次の手順に従ってください:

1. 以下の*いずれか*の操作を実行します。
   - プロビジョニング サーバ ツリーでマッピング概要ファイルをエクスポートするエンドポイント タイプを右クリックし、［Export Mapping Summary File］をクリックします。
   - マッピング概要ファイルをエクスポートするプロジェクトを開き (P. 39)、［Metadata］ -［Export Mapping Summary File］をクリックします。

     ［Export Mapping Summary File］ダイアログ ボックスが表示されます。
2. ファイルを保存するロケーションを指定して、［OK］をクリックします。

   Connector Xpress によって、マッピング概要ファイルが指定したロケーションに HTML 形式で保存されます。
3. Web ブラウザでファイルを開き、エンドポイント マッピングを確認します。

XML データ モデル ファイルのインポート

以前にエクスポートした XML データを編集するには、XML データ モデル ファイルをインポートします。 既存のプロジェクトに XML データ ファイルをインポートすると、エンドポイントの既存のメタデータが置換されます。

次の手順に従ってください:

1. ［Metadata］ - ［Import］をクリックします。

   ［Import Data Model from XML file］ダイアログ ボックスが表示されます。

2. インポートする XML ファイルのロケーションに移動し、［OK］をクリックします。

   Connector Xpress によって新しいデータ モデルが展開され、エンドポイントの既存のメタデータが置換されます。

例: 管理者が XML データ ファイルを使用してメタデータをインポートおよび展開する方法

この例で管理者は、エンドポイント *dyn_jdbc_endpoint* のメタデータを XML データ ファイル *export_dyn_jdbc_endpoint2.xml* のメタデータに置換しようとしています。管理者は、以下の手順に従います。

1. Connector Xpress プロジェクトで新しいメタデータを作成します。
2. メタデータを *dyn_jdbc_endpoint2.con* という名前のプロジェクトに保存します。
3. メタデータを *export_dyn_jdbc_endpoint2.xml* という名前の XML データ ファイルとしてエクスポートします。
4. *dyn_jdbc_endpoint.con* プロジェクト ファイルを開きます。

   このファイルには、管理者が置換したいエンドポイント *dyn_jdbc_endpoint* のメタデータが含まれています。

5. *dyn_jdbc_endpoint.con* プロジェクトに *export_dyn_jdbc_endpoint2.xml* データ ファイルをインポートします。
6. *dyn_jdbc_endpoint* という名前のエンドポイントにマージされたデータ モデルを展開します。

   Connector Xpress によって新しいデータ モデルが展開され、*dyn_jdbc_endpoint* の既存のメタデータが置換されます。

XML ファイルにデータ モデルをエクスポートする

現在開いているプロジェクトのメタデータを XML ファイルにエクスポートできます。

次の手順に従ってください:

1. メタデータをエクスポートするプロジェクトを開き (P. 39) ます。
2. ［Metadata］ - ［Export］をクリックします。

   ［Export Data Model to XML file］ダイアログ ボックスが表示されます。
3. ファイルを保存するロケーションを指定して、［OK］をクリックします。

   Connector Xpress によって、指定したファイルにメタデータがエクスポートされ、保存されます。

変更されたデータ モデルのマージ

ロードされているメタデータにロードおよびマージするデータ モデル XML ファイルを選択できます。XML データ ファイルをマージするとアカウント属性がエンドポイントのメタデータに追加されます。

次の手順に従ってください:

1. メタデータをマージする既存のプロジェクトを開き (P. 39) ます。
2. ［Metadata］ - ［Merge］をクリックします。

   ［Merge Data Model from］ダイアログ ボックスが表示されます。
3. メタデータをマージするファイルのロケーションに移動し、［OK］をクリックします。

   Connector Xpress によって、ロードされているデータ モデルに、開いたファイルのデータ モデル ファイルが上書きまたは追加されます。

例: 管理者が XML データ ファイルを使用してエンドポイントのメタデータをマージする方法

この例で管理者は、あるプロジェクトのアカウント属性をいくつか、別のプロジェクトのメタデータに追加しようとしています。 管理者は、*dyn_jdbc_endpoint.con* プロジェクトのメタデータを *dyn_jdbc_endpoint2.con* プロジェクトのメタデータとマージします。

dyn_jdbc_endpoint.con プロジェクトには、*dyn_jdbc_endpoint* という名前のエンドポイントのメタデータが含まれています。 管理者は、以下の手順に従います。

1. *export_dyn_jdbc_endpoint2.xml* という名前の XML データ ファイルに *dyn_jdbc_endpoint2.con* プロジェクトのメタデータをエクスポートします。
2. *dyn_jdbc_endpoint.con* プロジェクトを開きます。
3. *dyn_jdbc_endpoint.con* プロジェクトに *export_dyn_jdbc_ns2.xml* をインポートします。
4. Metadata Editor を使用して、Connector Xpress によって *dyn_jdbc_endpoint.con* プロジェクトのメタデータが *dyn_jdbc_endpoint2.con* プロジェクトで作成された追加のアカウント属性と正常にマージされたことを確認します。
5. *dyn_jdbc_endpoint* という名前のエンドポイントにマージされたデータ モデルを展開します。
6. プロビジョニング マネージャを使用して、新しいマッピングが正しいことを確認します。

エンドポイントの管理

お使いの環境でアカウント、ユーザ、グループおよびその他のリソース オブジェクトを管理するには、以下の手順に従います。

- エンドポイントを取得する
- 管理するオブジェクトのエンドポイントを検索する
- グローバル ユーザにオブジェクトを関連付けする

エンドポイントを検索および関連付ける方法を指定できます。 たとえば、コンテナで検索されるレベルの数を指定できます（例： 1 レベル、現在のサブツリーのみ）。

関連付け関数によって存在しないユーザを作成するか、一致しないユーザを（デフォルト ユーザ）グローバル ユーザに関連付けるかどうかを選択できます。

また、エンドポイント タイプを作成、消去、および削除できます。

エンドポイントを取得および管理する方法

エンドポイントを取得すると、取得したエンドポイントで検出されたアカウント、およびその他のオブジェクトがプロビジョニング サーバに自動的に入力されます。取得したエンドポイントは、プロビジョニング サーバによって管理された任意のアプリケーションまたはコンピュータを指します。

エンドポイントの取得は、エンドポイントの作成、検索、関連付けから構成されます。 エンドポイントの関連付けとは、エンドポイントのアカウントを検査し、アカウント属性から自動的にユーザを作成し、グローバル ユーザ属性の設定を可能にするプロセスです。

Connector Xpress を使用して、エンドポイントを管理するには、以下の手順に従います。

1. プロビジョニング サーバでエンドポイントを作成します。エンドポイントを登録すると、プロビジョニング サーバによって管理されたエンドポイントとして宣言しています。
2. コンテンツを検索します。エンドポイントを検索すると、Connector Xpress によってエンドポイントのオブジェクトが検出され、そのレコードがプロビジョニング サーバに保存されます。
3. 検索されたアカウントが関連付けられます。 エンドポイントのアカウントが関連付けられる場合、プロビジョニング サーバによってプロビジョニング ディレクトリのグローバル ユーザが関連付けられます。関連付け関数によって存在しないグローバル ユーザを作成するか、一致しないグローバル ユーザを（デフォルト ユーザ）グローバル ユーザに関連付けるかどうかを選択できます。
4. アカウントからグローバル ユーザ フィールドを更新します。

   関連付け関数を使用して存在しないユーザを作成する場合でも、オプションでアカウントの属性値を使用して、選択したユーザ属性を設定する（またはリフレッシュ）追加の関数を実行できます。

詳細情報:

［Create New Endpoint］ダイアログ ボックス (P. 149)
［Explore/Correlate Endpoint］ダイアログ ボックス (P. 167)

エンドポイントの取得、検索、および関連付け

エンドポイントを取得するには、まずプロビジョニング サーバにエンドポイントを登録します。登録では、コネクタと通信できるように、エンドポイントの接続の詳細および認証情報を指定します。次に、エンドポイントで管理するオブジェクトを検索して、オブジェクトをグローバル ユーザに関連付けます。

エンドポイントの取得、検索、および関連付けするには

1. プロビジョニング サーバ ツリーで検索または関連付けるエンドポイント ノードを右クリックし、［Acquire Endpoint］をクリックします。

   ［Create New Endpoint］ダイアログ ボックスが表示されます。

2. ［Create New Endpoint］ダイアログ ボックスのフィールドに入力して、［OK］をクリックします。

   これで、エンドポイントの接続詳細および認証情報が指定されました。

3. プロビジョニング サーバ ツリーで作成したエンドポイント ノードを右クリックし、［Explore/Correlate endpoint］をクリックします。

   ［Explore/Correlate Endpoint］ダイアログ ボックスが表示されます。

4. ［Explore/Correlate Endpoint ］ダイアログ ボックスのフィールドに入力して、［OK］をクリックします。

   これで、Connector Xpress でエンドポイントを検索および関連付けする方法が指定されました。

詳細情報:

［Create New Endpoint］ダイアログ ボックス (P. 149)
［Explore/Correlate Endpoint］ダイアログ ボックス (P. 167)

エンドポイント タイプの作成

ロードしたメタデータを使用して、エンドポイント タイプを作成できます。

注: エンドポイント タイプは、Connector Xpress プロジェクトが開いている場合にのみ作成できます。

次の手順に従ってください:

1. 既存のプロジェクトを開きます (P. 39)。
2. プロビジョニング サーバ ツリーで［Endpoint Types］ノードを右クリックし、［Create New Endpoint Type］をクリックします。

   ［Create New Endpoint Type］ダイアログ ボックスが表示されます。
3. ダイアログ ボックスのフィールドに入力して、［OK］をクリックします。

   これで、新しいエンドポイント タイプの名前が定義され、指定したプロビジョニング サーバの［Endpoint Types］ノードにこのエンドポイント タイプが追加されました。

詳細情報:

［UI from TM. Not exist in glossary］ダイアログ ボックス (P. 149)

エンドポイント タイプの削除

プロビジョニング サーバのエンドポイント タイプおよびそのエンドポイントの管理が不要になった場合、Connector Xpress を使用して、エンドポイント タイプを削除できます。

重要: 動的コネクタに関連付けられたエンドポイント タイプを削除する場合、正常に削除できるよう、Connector Xpress を使用することを強くお勧めします。

エンドポイント タイプを削除するには、プロビジョニング サーバ ツリーの［Endpoint Types］ノードを展開して、削除するエンドポイント タイプを右クリックし、［Delete Endpoint Type］をクリックします。

Connector Xpress によってエンドポイント タイプ、関連するアカウント テンプレートおよびすべての子エンドポイントが削除されますが、 エンドポイントのデータは変更されません。

エンドポイント タイプの消去

選択したエンドポイント タイプからアカウント テンプレートおよびエンドポイントをすべて削除すると、エンドポイント タイプを消去できます。エンドポイント タイプを消去すると、プロビジョニング サーバからエンドポイント オブジェクトがすべて削除され、デフォルトのアカウント テンプレート オブジェクトが再作成されます。

次の手順に従ってください:

1. プロビジョニング サーバ ツリーでエンドポイント タイプを消去するサーバを右クリックし、［Clean Endpoint Type］をクリックします。

   ［Clean Endpoint Type Objects］ダイアログ ボックスが表示されます。

2. エンドポイント タイプを消去することを確認します。

   Connector Xpress によって、選択されたエンドポイント タイプのアカウント テンプレートおよびエンドポイントがプロビジョニング ディレクトリからすべて削除されます。 エンドポイントのデータは変更されません。

エンドポイントの削除

プロビジョニング マネージャでコネクタの管理が不要になった場合、Connector Xpress を使用して、コネクタを削除できます。

重要: 動的コネクタに関連付けられたエンドポイントを削除する場合、正常に削除できるよう、Connector Xpress を使用することを強くお勧めします。

エンドポイントを削除するには、プロビジョニング サーバ ツリーの［Endpoints］ノードを展開して、削除するエンドポイントを右クリックし、［Delete Endpoint］をクリックします。

Connector Xpress によってコネクタが削除され、エンドポイントと関連アカウント テンプレートが除去されますが、エンドポイントは変更されません。

詳細情報:

エンドポイントの管理 (P. 110)

構成データのロケーション

Connector Xpress を使用して作成された構成データは、以下のセクションで説明される分散システム全体のさまざまなロケーションに保存されます。

CA IAM CS ルーティング ルールのロケーション

ルーティング ルールは、以下のロケーションにあります。

```
eTSAName=[CSNAME],eTSAContainerName=SAs,eTNamespaceName=CommonObje
cts,dc=[DOMAIN],dc=eta
```

エンドポイント タイプ メタデータのロケーション

エンドポイント タイプ メタデータは、以下のロケーションにあります。

```
eTNamespaceName=[NAMESPACE],dc=[DOMAIN],dc=eta
```

エンドポイント タイプ メタデータおよび CA IAM CS ルーティング ルール

エンドポイント タイプ メタデータおよび CA IAM CS ルーティング ルールは、プロビジョニング サーバに保存されます。

エンドポイント タイプ メタデータおよび CA IAM CS ルーティング ルールを表示するには、JXplorer などの LDAP ブラウザを使用してプロビジョニング サーバに接続します。

エンドポイント タイプ メタデータおよび CA IAM CS ルーティング ルールの表示

エンドポイント タイプ メタデータおよび CA IAM CS ルーティング ルールを表示するには、JXplorer などの LDAP ブラウザを使用してプロビジョニング サーバに接続します。

CA IAM CS ルーティング ルールの改装

CS にはそれぞれ、以下にエントリがあります。

```
eTSAName=CS
name,eTSAContainerName=SAs,eTNamespaceName=CommonObjects,dc=server
,dc=eta
```

CS によって管理されたエンドポイント タイプにはそれぞれ、以下のようなエントリがあります。

```
eTSABranchDN : eTNamespaceName=Oracle Server,dc=server
```

エンドポイント タイプの識別名

エンドポイント タイプの識別名は以下のとおりです。

eTNamespaceName=
MyEndpointType,dc=*Domain*,dc=*TopLevelProvisioningServer*

以下の属性を使用して、動的エンドポイント タイプ オブジェクトそれぞれにメタデータが格納されます。

eTMetaData

（必須）ストアド プロシージャおよび操作バインディングを除く、ウィザード マッピングがすべて含まれる XML ドキュメントを格納します。

eTOpBindingsMetaData

（オプション）ストアド プロシージャおよび操作バインディングが含まれる XML ドキュメントを格納します。

Connector Xpress のローカル ストレージ

Connector Xpress のプロジェクト ファイル（.con ファイル）はローカル ファイル システムに格納されます。

Connector Xpress のデータ ソース設定は Connector Xpress のユーザ設定に格納されます。

注: Connector Xpress データ ソース設定はエクスポートできません。

詳細情報:

環境設定の設定 (P. 25)
プロジェクト (P. 29)
ユーザ設定の格納場所 (P. 25)

メタデータの編集

プロジェクトのメタデータは編集できます。

重要: メタデータ エディタは、ユーザの責任で使用してください。メタデータの重要なオブジェクトを削除または変更すると、動作不能に陥る場合があります。

次の手順に従ってください:

1. 既存のプロジェクトを開きます (P. 39)。
2. プロビジョニング サーバ ツリーで、メタデータを編集するエンドポイント タイプを右クリックし、［Edit Metadata］をクリックします。

   ［Edit Metadata for Endpoint Type］ダイアログ ボックスが表示され、XML ツリーにメタデータが表示されます。
3. メタデータを編集し、［Save］をクリックします。

   Connector Xpress によってメタデータを再展開されます。

例: 管理者が手動でメタデータを変更する方法

この例で管理者は、マッピング ツリーを使用せずに、手動でメタデータを変更することにより、表示名マッピングを last name から first name に変更しようとしています。 管理者は、以下の手順に従います。

1. dyn_jdbc_namespace.con プロジェクト ファイルを開きます。
2. dyn_jdbc_namespace.con エンドポイントを右クリックし、［Edit Metadata］を選択します。
3. dyn_jdbc_namespace ノードを展開し、Metadata Editor で［Classes］ - ［eTDYNAccount］ - ［Properties］ - ［eTDYN-str-01］に移動します。
4. Last name から First name に displayName 属性を変更します。
5. beanPropertyName 属性を lastName から firstName に変更します。
6. 変更されたメタデータを再展開します。
7. メタデータをプロビジョニング マネージャで検証して、正常に変更されたことを確認します。

詳細情報:

プロジェクト (P. 29)
［Edit Metadata for Endpoint Type］ダイアログ ボックス (P. 159)

列挙値

静的列挙の場合、以下に従い新規タイプを定義します。

- 一意のタイプ名を新しく作成する
- それぞれ内部値から構成されている、既知の値のリストおよびオプションの表示値を作成する

注: 新しい enum タイプの名前に int または integer を指定しても問題になりませんが、メタデータをメンテナンスしにくくなります。

クエリ ベースの列挙については、以下を定義します。

- 非管理対象クラスへのマッピング
- そのクラスのオブジェクトがクエリのため利用可能にしておく属性
- 以前のクラスで適切に選択された属性による管理クラスおよび非管理対象クラス間の関連付け

メタデータと操作バインディング エディタの考慮事項

重要: メタデータ エディタを使用して、作成したマッピングを確認できますが、メタデータ エディタは、ユーザの責任で使用することをお勧めします。メタデータの重要なオブジェクトを削除または変更すると、動作不能に陥る場合があります。

メタデータの再展開

コネクタ プロジェクトが開いていて、メタデータが利用可能な場合、メタデータをプロビジョニング サーバのエンドポイント タイプにプッシュできます。 メタデータは、動的エンドポイント タイプにのみ展開できます。

注: エンドポイント タイプにすでにメタデータがある場合、新しいメタデータによって属性名や属性タイプを追加するなどのデータ モデルの基本的な要素は変更できません。

次の手順に従ってください:

1. 既存のプロジェクトを開きます (P. 39)。
2. プロビジョニング サーバ ツリーでデータを展開するエンドポイント タイプを右クリックし、［Deploy Metadata］をクリックします。

   ［Deploy Metadata］ダイアログ ボックスが表示されます。
3. メタデータを展開することを確認します。

   Connector Xpress によって、メタデータが選択されたエンドポイント タイプに展開され、既存のメタデータがすべて上書きされます。

詳細情報:

プロジェクト (P. 29)

DYN Schema 拡張

DYN Schema 以下の属性に対して拡張されました。

- コンテナ オブジェクト数（eTDYNContainerXXX）が 10 に拡張されました。
- 汎用オブジェクト数（eTDYNObjectXXX）が 35 に拡張されました。
- 機能属性数（eTDYN-str-multi-ic-、eTDYN-str-ic-、eTDYN-str-c-、eTDYN-bool-c-、eTDYN-int-multi-c-、eTDYN-int-c- など）がすべて 99 に拡張されました。ただし、複数値の大文字と小文字を区別する属性数（eTDYN-str-multi-c-）は例外として、500 に拡張されました。
- 機能属性ではない複数値の大文字と小文字を区別するまたは大文字と小文字を区別しない属性数（eTDYN-str-multi-、eTDYN-str-multi-i- など）は、500 に拡張されました。
- キャッシュされた（データ ロケーションが BOTH と等しい）複数値の大文字と小文字を区別する汎用属性数（eTDYN-str-multi-ca-）は、99 に拡張されました。 これらの属性はすべての DYN オブジェクトに含まれています。

  キャッシュされた属性の値は、プロビジョニング サーバに格納されるため、エンドポイント システムに問い合わせずに、アクセスできます。 この値がエンドポイント システムによって提供されると、値をエンドポイント システムの属性と一致するために、検索操作を実行して、格納された値を更新する必要があります。
- キャッシュされた接続専用の複数値の大文字と小文字を区別する 99 の属性（eTDYN-str-multi-ca-sec-）は、DYN Directory にのみ追加されました。 これらの属性は、DYN パスワード属性の 「encryptwith」行に追加され、機密設定を難読化するために使用できます。

メタデータの説明

Connector Xpress、JIAM および CA IAM CS を含むアーキテクチャのレイヤすべてで使用されるメタデータの詳細については、CA Identity Manager Bookshelf の JavaDocs を参照してください。

Connector Xpress を汎用メタデータ エディタとして使用する

Connector Xpress を JNDI または JDBC 以外のエンドポイント タイプの汎用エディタとして使用できます。 Connector Xpress を汎用エディタとして使用すると、特定の実装バンドル用ではない、汎用メタデータを編集できます。 たとえば、スクリプト記述コネクタのメタデータを編集するには、特定のスキーマにマップされないクラスおよび属性を最初から作成します。

以下の方法で汎用メタデータを作成および編集できます。

- 属性マッピング テーブルで検出された属性を選択するのではなく、新しい属性を作成します。
- 事前定義済み辞書ファイルから、事前定義済みカスタム動作属性のセットをロードします。（P. 123）
- 事前定義済み辞書ファイルから選択した動作属性をロードします。
- データ ファイルからメタデータ プロパティの拡張セットをロードします。（P. 122）

新規属性の作成

Connector Xpress を汎用メタデータのエディタとして使用して、属性マッピング テーブル内で検出された属性を選択するのではなく、新規属性を作成できます。この手順に従い、複合型属性やエンドポイント属性にマップされない他のコネクタに固有の属性を定義できます。

次の手順に従ってください:

1. プロジェクト（P. 34）を作成し、データ ソースを選択するプロンプトが表示されたら、［No source］を選択します。

   ［Endpoint Type Details］ダイアログ ボックスが表示されます。

2. ［Endpoint Type Details（P. 166）］ダイアログ ボックスのフィールドに入力します。
3. ［Classes］ノードを展開し、マッピングするクラスの［Attributes］ノードをクリックします。

   ［Map Attributes］ダイアログ ボックスが表示されます。

4. ［Maps To］フィールドに、追加する属性の名前を入力します。

   Connector Xpress によって、新しい属性がマッピング ツリーに追加されます。

5. マッピング ツリーの［Attributes］ノードを展開し、属性の新規ノードをクリックします。

   ［Attribute Details］ダイアログ ボックスが表示されます。

6. ［Attribute Details］ダイアログ ボックスのフィールドに入力します。

詳細なメタデータ プロパティを表示する

さまざまなダイアログ ボックスにメタデータ プロパティの拡張セットを表示できます。

詳細なメタデータ プロパティを表示する

1. プロジェクトを作成 (P. 34)するか、既存のプロジェクトを開き (P. 39)ます。
2. ［Tools］ - ［Preferences］をクリックします。

   ［Connector Xpress Preferences］ダイアログ ボックスが表示されます。
3. ［Show Extended Metadata］チェック ボックスをオンにします。

   Connector Xpress には拡張メタデータのプロパティが表示され、このデータ モデル アイテムに関連する、追加のメタデータがすべて含まれます。

   注: 表示されるフィールドの詳細については、「拡張メタデータのプロパティ」を参照してください。

詳細情報:

［Endpoint Type Details］ダイアログ ボックス (P. 166)

動作属性のロード方法

Connector Xpress では、動作属性が自動的にロードされません。ただし、JNDI エンドポイント タイプの場合、以下のいずれかの方法で動作属性をロードできます。

- 辞書から、カスタム動作属性の辞書をロードする (P. 123)

  辞書はそれぞれのベンダーまたは規格に基づきます。 以下の辞書を選択できます。

  - CA Directory
  - RFC 2252
  - X.500
- 選択された動作属性を、選択された構造ネイティブ クラスのすべての属性リストにロードします。

辞書から動作属性をロードする

JNDI の場合、辞書から、事前定義済みカスタム動作属性のセットをロードします。

に動作属性は、他の属性と同じようにマッピングできます。

次の手順に従ってください:

1. プロジェクトを作成 (P. 34)するか、既存のプロジェクトを開き (P. 39)ます。
2. ［Project］ - ［Settings］をクリックします。

   ［Edit Project Settings］ダイアログ ボックスが表示されます。
3. 動作属性をロードする辞書を選択し、［OK］をクリックします。
4. マッピングするクラスの［Attributes］ノードをクリックします。

   ［Map Attributes］ダイアログ ボックスが表示されます。
5. ［Maps to］フィールドの右にある［Edit］アイコンをクリックします。

   Connector Xpress で、動作属性が［Maps to］リストに表示されます。
6. 動作属性をマッピングします。

   注: 動作属性の一部のみをロードするには、リストからロードする辞書属性を選択し、作成した属性をマッピング ツリーでクリックします。 ［Attribute Details］ダイアログ ボックスのフィールドに入力します。

第 5 章: Connector Xpress のトラブルシューティング

このセクションには、以下のトピックが含まれています。

Flexi-DN データが一重引用符を含むように変換される

MS SQL Server コネクタにのみ適用されます。

症状:

ユーザのデータは、多数の DN 文字列を含む複数値属性に関して、Flexi-DN データ型を使用できます。

Connector Xpress では、デフォルト コンバータによって、すべての値に一重引用符が追加されます。これにより、SQL Server に関する問題が発生する場合があります。

解決方法:

SQL クエリに渡す前に属性をフォーマットする方法を選択できるようになりました。

引用符のない文字列を Connector Xpress に生成させるには、平坦化スタイル「SQLQUOTELESS」を使用します。

例: SQLQUOTELESS による引用符のない文字列の生成

この例では、以下のような DN 文字列を処理します。

```
eTDYNObject001Name=1,eTDYNContainer001Name=Role Container
```

Flexi-DN データ型用のデフォルト コンバータは、この完全 DN 文字列からロール ID 「1」を抽出します。

この属性にストアド プロシージャ バインディングを追加すると、CA IAM CS によって、各値に一重引用符の付いたロール ID 文字列が生成されます。

```
"'1', '2', '3', '4'"
```

平坦化スタイル「SQLQUOTELESS」を使用すると、このロール ID 文字列が、SQL Server に適した形式に変換されます。

```
"1, 2, 3, 4"
```

動的エンドポイントの管理コネクタ サーバを設定できない

Windows および UNIX システムで有効なコマンド

症状:

DYN エンドポイントを C++ コネクタ サーバで管理するように選択すると、失敗します。また、エラー メッセージが表示されます。

この失敗は、一般的にエンドポイント タイプ（DYN など）と管理コネクタ サーバ（C++ CS など）のタイプが一致していないことが原因で発生します。

解決方法:

動的エンドポイント タイプの管理に CA IAM CS を選択します。

ID/シーケンス列のサポート

Connector Xpress では、現在 ID/シーケンス列（自動増分オブジェクト ID 列）のマッピングをサポートしません。

IDENTITY_INSERT が Off に設定されているため、挿入に失敗します

MSSQL のみで有効

症状:

アカウントを作成すると、「Failed to Insert because IDENTITY_INSERT is Set to Off」というメッセージが表示されます。このメッセージは、データベース テーブルの IDENTITY_INSERT がオフになっているため発生します。

解決方法:

アカウントを作成するストアド プロシージャを記述し、挿入操作の前に IDENTITY_INSERT をオンに設定し、IDENTITY_INSERT をオフにするスクリプトを記述します。 以下の手順を実行します。

1. timing=PRE (Before) を使用して IDENTITY_INSERT [database. [owner.]] { table } をオンに設定する 1 行のストアド プロシージャを記述します。
2. timing=POST (After) を使用して IDENTITY_INSERT [database. [owner.]] { table } をオンに設定する 1 行のストアド プロシージャを記述します。

複数の外来キー制約

アカウントにマップされたデータベース テーブルが複数の外来キー制約に関連付けられている場合は、以下のいずれかの手順に従います。

- アカウント、およびアカウントの他のテーブルとの関連付けを削除するストアド プロシージャを記述します。
- データベース スキーマにカスケード削除を設定します。

たとえば、社内インベントリ データベースでは、カスタマ グループ テーブルおよびカスタマ注文テーブルの外来キーは、カスタマ アカウント テーブルを参照します。 Connector Xpress で、カスタマ アカウントはアカウントにマップされ、カスタマ グループはアカウント グループにマップされます。 ユーザがアカウントを削除すると、Connector Xpress によって、このアカウントに関係するアカウント グループのエントリがすべて自動的に削除されます。 ただし、カスタマ注文テーブルは Connector Xpress によって削除されません。 削除操作は外来キー制約の違反のために失敗します。

その問題を解決するには、以下のいずれかの手順に従います。

- 実際のアカウントが削除される *前* に削除するアカウントに関連付けられたエントリをすべて削除するストアド プロシージャを記述します。
- データベース ベンダーまたはスキーマによって、カスケード削除がサポートされている場合は、カスタマ アカウントに設定します。

ロール定義ジェネレータが実行されない

Windows および UNIX システムで有効なコマンド

症状:

ロール定義ジェネレータを実行しようとすると、以下のエラー メッセージが表示されます。

```
java.lang.UnsupportedClassVersionErrorException
```

パスで新バージョンの前に古いバージョンの Java JRE が表示されるとこのエラーが発生します。

解決方法:

パスで、Java 5 JRE が古いバージョンの前に表示されることを確認します。

バイナリタイプ属性のサポート

CA Identity Manager を使用してアカウントを管理するため、Connector Xpress で DYN マッピングを実行している場合、CA Identity Manager では、DYN エンドポイント タイプに対してバイナリタイプ属性をサポートしないので注意してください。

その結果、Connector Xpress でバイナリ データ型として指定した属性はすべて、CA Identity Manager アカウント管理画面に表示されません。

DYN エンドポイント タイプでバイナリ属性を管理する場合は、プロビジョニング マネージャを使用することをお勧めします。

JDBC エンドポイントの必須属性のサポート

JDBC エンドポイントで必須（つまり、NOT NULL）として定義されていて、プロビジョニングの視点では必須ではない属性をサポートするため、JDBC コネクタで、プロビジョニング属性に空の値を使用できます。

アカウントを作成し、説明フィールドなどの非必須プロビジョニング属性にエンドポイントの必須属性をマップする場合、説明フィールドに値を入力しないと、JDBC コネクタによって、エンドポイント データベースの NOT NULL 列にスペースが入力され、空のプロビジョニング属性がエンドポイント属性にマッピングされます。

プロビジョニング マネージャで非必須フィールドのマッピングをサポートするため、CA IAM CS には、非必須プロビジョニング属性の空の値をエンドポイントの必須属性のスペースに変換する NullValueClassConverter が含まれます。

たとえば、レガシー データベース システムでユーザ アカウントにマップされるテーブルに NOT NULL の説明フィールドがあると、空の値が発生する場合があります。ただし説明フィールドは、プロビジョニング マネージャで必須に設定されていません。つまり、アカウントの作成にユーザの説明を入力する必要がありません。 空の値を格納するために NullValueClassConverter が使用されます。

NullValueClassConverter の *pluginConfig* クラスにあるプロパティ *nullValue* は、デフォルトでスペースに設定されます。 *nullvalue* は他の値に変更できますが、*nullvalue* を変更すると、エンドポイント データベースで追加の設定が必要になります。 (P. 133)

JDBC エンドポイントの場合、CA IAM CS によってコンバータが自動的にロードされます。 他のエンドポイント タイプ（JNDI など）では、コンバータをロードするには、手動で CA IAM CS を設定します。 (P. 132)

プロビジョニング マネージャで非必須フィールドのマッピングをサポートする方法

JDBC コネクタを使用して、プロビジョニング マネージャの非必須フィールドを JDBC データベースの not-null 列にマッピングできるようにするには、*NullValueClassConverter* を使用して変換する各属性の Connector Xpress Dyn マッピングにメタデータ属性 *useSpecialNullValue* を追加します。 CA IAM CS によって、コンバータを有効にする前にこのメタデータ属性が存在していることが確認されます。

注: この手順を使用して、他のタイプのエンドポイントのサポートを有効にできます。

サポートを有効にするには、以下の手順に従います。

1. Connector Xpress または別のエディタを使用して、エンドポイント タイプのメタデータを編集します。
2. *NullValueClassConverter* で変換する属性に、新しいブール メタデータ属性 *useSpecialNullValue* を追加し、ブール値を true に設定します。

   注: 変換する各属性の Connector Xpress Dyn マッピングにメタデータ属性 *useSpecialNullValue* を追加します。 CA IAM CS によって、コンバータを有効にする前にこのメタデータ属性が存在していることを確認されます。
3. *isRequired* メタデータ属性を false に設定します。
4. 変換するすべての属性に対して手順を繰り返します。

例: プロビジョニング マネージャで非必須フィールドのマッピングをサポートする

この例では、メタデータの編集に Connector Xpress を使用します。 アカウントの Description 属性は eTDYN-str-01 にマップされます。

次の手順に従ってください:

1. Connector Xpress で、メタデータを編集するエンドポイント タイプを右クリックし、［Edit Metadata］をクリックします。

   ［Edit Metadata for Endpoint Type］ダイアログ ボックスが表示されます。

2. ［eTDYN-str-01］ノードを展開し、メタデータ サブノードを選択します。
3. ［Add］ボタンをクリックします。

   Connector Xpress によって、新しいメタデータ属性ノードがツリーに追加されます。

4. 新しいメタデータ ノードの名前を *useSpecialNullValue* に指定します。
5. タイプ ドロップダウン リストから［Boolean］を選択します。
6. ドロップダウン リストの横にあるチェック ボックスをオンにします。

   useSpecialNullValue 属性の値が *true* に設定されます。

7. *eTDYN*-str-01 で isRequired メタデータ属性を選択します。
8. ドロップダウン リストの隣のチェック ボックスをオフにします。

   isRequired 属性の値が *false* に設定されます。

JDBC 以外のエンドポイントで必須属性のサポートを有効にする方法

属性を必須（すなわち NOT NULL）として定義するが、プロビジョニングの視点では必須である必要がない JDBC 以外のエンドポイントをサポートするには、以下の手順に従います。

1. CA IAM CS を設定して、NullValueClassConverter をロードします。（P. 132）
2. プロビジョニング マネージャで非必須フィールドのマッピングをサポートします。（P. 130）

CA IAM CS を設定して、NullValueClassConverter をロードします。

NullValueClassConverter プラグインは JDBC コネクタと一緒に提供されます。 JDBC 以外のエンドポイント タイプで NullValueClassConverter を使用するには、CA IAM CS を設定して、コンバータをロードします。

注: JDBC エンドポイントでは、コンバータは自動的にロードされます。

注: プロパティ *metadataPropNames* の値は、*useSpecialNullValue* です。プラグインによって処理される各属性の Connector Xpress Dyn マッピングに *metadataPropNames* メタデータ属性を追加します。 CA IAM CS によって、プラグインを有効にする前にこのメタデータ属性が存在していることが確認されます。

プラグインを有効にするには、エンドポイントの connector.xml を使用してプラグインを設定します。

次の手順に従ってください:

1. C:¥Program Files¥CA¥Identity Manager¥Connector Server¥conf¥override¥jdbc to connector.xml にある SAMPLE.connector.xml のファイル名を変更します。
2. connector.xml ファイルに以下のエントリを追加します。

```
<property name="classPluginConfigs">
<list>
    <bean class="com.ca.jcs.cfg.MetaPluginConfig">
    <property name="pluginClass">
    <value>com.ca.jcs.converter.meta.NullValueClassConverter</value>
</property>
<property name="pluginConfig">
    <bean
    class="com.ca.jcs.converter.meta.NullValueClassConverter$NullValueConverter
    Config">
    <property name="nullValue">
    <value> </value>
    </property>
    </bean>
    </property>
<property name="metadataPropNames">
    <list>
    <value>useSpecialNullValue</value>
```

```
</list>
</property>
</bean>
</list>
</property>
```

注: NullValueClassConverter の *pluginConfig* クラスにあるプロパティ **nullValue** は、デフォルトでスペースに設定されます。 プラグインによって使用されるヌル値を変更するには、設定 bean で nullValue を変更します。

空の値を格納するために使用されるデフォルト値を変更する

NullValueClassConverter の pluginConfig クラスにあるプロパティ *nullValue* は、デフォルトでスペースに設定されます。空の値を格納するために使用されるデフォルト値を変更するには、connector.xml ファイルでクラス プラグイン設定にある設定 bean の nullValue を変更します。

あるエンドポイント対してマッピングし、別のエンドポイントから取得する

あるエンドポイントに対してマッピングし、別のエンドポイントから参照するディレクトリを取得する場合は、いずれのテーブルとストアド プロシージャ（JDBC）または LDAP スキーマ（JNDI）も同じことを確認します。

テーブル属性のマッピング

テーブルと属性のマッピングを定義する場合、［Map Class and Attributes］ダイアログ ボックスの［Type］フィールドでさまざまな選択を行うことができます。選択を行うと、使用する事前定義済み属性、および属性動作が定義されます。

［Attribute Details］ダイアログ ボックスの［Synchronized］チェック ボックスをオンにすると、Connector Xpress によって属性のデータ型から自動的に同期タイプが決定されます。

たとえば、ユーザの選択により、そのタイプおよび動作に対して事前定義済み属性数の制限が設定されます。 以下のテーブルには、制限を示します。

タイプと動作	上限
単一値の非機能整数	10
単一値の機能整数	99
複数値の非機能整数	10
複数値の機能整数	99
単一値の大小文字を識別する非機能文字列	30
単一値の大小文字を識別しない非機能文字列	30
単一値の大小文字を識別する機能文字列	99
単一値の大小文字を識別しない機能文字列	99
複数値の大小文字を識別する非機能文字列	500
複数値の大小文字を識別しない非機能文字列	500
複数値の大小文字を識別する機能文字列	500
複数値の大小文字を識別しない機能文字列	99
単一値の非機能ブール値	10
単一値の機能ブール値	99
単一値のバイナリ	10

タイプのマッピング

Connector Xpress では、明示的に整数同期を要求しないと、*edittype=int* によって LDAP テキスト形式属性にマッピングされます。

MySQL および Informix のストアド プロシージャ サポート

Connector Xpress では、以下のベンダーが提供する以下のストアド プロシージャがサポートされません。

- **MySQL** - MySQL ストアド プロシージャ パラメータは必要な JDBC メソッドによってアクセスできないため、サポートされていません。
- **Informix** - Informix でサポートされる、ストアド プロシージャの名前は同じですが、パラメータが異なります。 このことから、ストアド プロシージャとそのパラメータをリスト表示することが難しいため、サポートされていません。

JDBC 命名属性

アカウントおよびグループ命名属性は、マッピングされたネイティブ テーブルで必要な列のみをマッピングできます。 つまり、ヌル値を許可しない属性が該当します。この列が含まれるテーブルでプライマリ キーとして機能していることを確認するか、少なくとも一意のインデックスが適用されることを確認してください。

Sybase ストアド プロシージャのエラー

Windows および UNIX システムで有効なコマンド

症状:

Sybase のストアド プロシージャを呼び出すと、以下の例外がレポートされます。

Nested exception is com.sybase.jdbc3.jdbc.SybSQLException: Stored procedure 'AddObject_Instead_Parameter' may be run only in unchained transaction mode. The 'SET CHAINED OFF' command will cause the current session to use unchained transaction mode.

解決方法:

CA IAM CS では、Sybase ストアド プロシージャがセットを、オプションを上につなぐことを必要とします。

Sybase ストアド プロシージャすべてに対してチェーン オプションをオンに設定します。

Connector Xpress のログ記録

Connector Xpress のログ メッセージは以下のファイルに書き込まれます。

im_home¥Connector Xpress¥logs¥conxp-log.txt

注: 生成されたメタデータは通常、診断されている問題に関する主要リソースです。 Connector Xpress ログ ファイルを使用して、生成されたメタデータを分析することをお勧めします。

第 6 章: 画面とダイアログ ボックス

このセクションには、以下のトピックが含まれています。

［Account Screens］ダイアログ ボックス

［Account Screens］ダイアログ ボックスは、マッピングされた属性をグループとサブグループへグループに論理的にグループ化します。 これらの属性は、CA Identity Manager ユーザ コンソールのアカウント画面でタブおよびページとして表示されます。 このダイアログ ボックスを使用して、エンドポイントから CA Identity Manager でアカウントを編集するのに使用するアカウント画面を設計できます。Connector Xpress によって、この画面上で作成されたグループがメタデータに保存されます。 このメタデータをロール定義ジェネレータで使用して、適切な画面定義を生成して、CA Identity Manager にインポートできます。

注: 詳細については、「Connector (80P.)Xpress ガイド」の「ユーザ コンソールのアカウント画面の生成方法」を参照してください。

以下のフィールドは、CA Identity Manager ユーザ コンソールのアカウント画面のタブおよびページのセクションで属性が表示される方法を定義します。

Connector Xpress によって、以下のデフォルト アカウント画面レイアウトに共通の属性がグループ化されます。

Account

エンドポイント アカウントのログイン、パスワードおよびステータス情報を定義します。

Login page section

エンドポイント アカウントのログイン情報を定義します。

Password page section

エンドポイント アカウントのパスワード情報を定義します。

Status page section

エンドポイント アカウントのステータス情報を定義します。

User

ユーザの ID 情報を定義します。

Name page section

ユーザの名前を定義します。

Organization page section

ユーザが属する組織を定義します。

Contact

ユーザの連絡先および住所の情報をすべて定義します。

Internet page section

ユーザのインターネット連絡先情報を定義します。

Address page section

ユーザの住所を定義します。

Phone page section

ユーザの連絡先電話番号情報を定義します。

Membership

ユーザのメンバシップ情報をすべて定義します。

Add Tab

新しいタブを追加します。このタブは CA Identity Manager ユーザ コンソールのアカウント画面に表示されます。

Add Page Section

選択されたタブに新しいページ セクションを追加します。このページ セクションは、CA Identity Manager ユーザ コンソールのアカウント画面の指定されたタブに表示されます。

[Add Attribute]ドロップダウン

選択されたページ セクションに追加する属性を指定します。この属性は、CA Identity Manager ユーザ コンソールのアカウント画面の指定されたページおよびタブに表示されます。

[Attribute Details]ダイアログ ボックス

［Attribute Details］ダイアログ ボックスでは、特定のプロビジョニング属性の処理方法を制御する論理パラメータを定義できます。

このダイアログには以下のフィールドが含まれます。

Name

属性の名前を指定します。このラベルはユーザ インターフェースで属性を示すために表示されます。

Description

（オプション）属性の説明です。

Connector Map To

イン コネクタ対話にマッピングするオブジェクト クラス（コネクタ自体を含めて）または属性をマッピングする名前を指定します。 動的コネクタの場合、この属性によって、属性をマッピングするネイティブ システム アイテムの名前が指定されます。 たとえば、JDBC ベースのコネクタの場合、データベース テーブルの名前にオブジェクト クラスをマッピングし、その内の各プロパティを列のいずれかの名前にマッピングします。

Default Value

（オプション）属性のデフォルト値を指定します。

注: この値はクライアント（ユーザ インターフェースなど）のみが使用します。 クライアントによって値が提供されない場合、デフォルトではプロビジョニング サーバは値を使用しません。

Required

オンにすると、このタイプのオブジェクトすべてにこの属性があることをエンドポイントが要求するように指定します。

注: エンドポイント スキーマによってこの属性があることを要求される場合、Connector Xpress によってチェック ボックスが自動的にオンにされます。また手動で、チェック ボックスをオフにできません。

Minimum Length

この属性値に対する値の最小のバイト長を指定します。

この値は CA Identity Manager アカウント画面で入力妥当性検査に使用されます。

Maximum Length

この属性値に対する値の最大のバイト長を指定します。

この値は入力妥当性検査に使用されます。

Allowed Operations

この属性に対して実行できる操作を指定します。以下のいずれかのオプションを選択します。

Create

親クラスのオブジェクト インスタンスを作成するとき属性の値が設定されるかどうかを指定します。

Read

親クラスのオブジェクト インスタンスを表示するとき属性の値が読み取られるかどうかを指定します。

Modify

親クラスのオブジェクト インスタンスを変更するとき属性の値が設定されるかどうかを指定します。

Minimum Value

（数値の属性タイプ）この属性に対して設定できる最小値を指定します。

Maximum Value

（数値の属性タイプ）この属性に対して設定できる最大値を指定します。

Data type

ネイティブ属性にマッピングしたプロビジョニング属性のデータ型を指定します。

Binary Data

値が任意のバイナリ データである属性を定義します。

Boolean

XML で true または false を論理的に指定しますが、プロビジョニング サーバおよび JIAM API では LDAP 属性値で 1 または 0 として表されます。

Date

日付を指定します。

例： 1999-05-31

注: プロビジョニング マネージャの動的名前空間プラグインでは、1800 ～ 9999 年までサポートします。ソリューションの他のコンポーネントではこのような制限はないので、実質的にどの年も表すことができます。

Date & Time

特定の曜日の特定の時刻を指定します。

例： 1999-05-31T13:20:00

注: プロビジョニング マネージャの動的名前空間プラグインでは、1970 ～ 2036 年までサポートします。したがって、この範囲外の日を表すには、［日付］を使用してください。

注: ベンダーが混在すると、Connector Xpress によって時間関連の列が処理される方法が複雑になります。 たとえば、MSSQL では「DATETIME」は DateTime 値を示します。しかし他のベンダーでは、標準的な「TIMESTAMP」が使用されています。また、MSSQL の TIMESTAMP は自動的に生成されるバイナリ値を表します。 さらに、Oracle では「TIME」タイプがサポートされず、「DATE」タイプは事実上 TIMESTAMP でもあります。そのため、ベンダー非依存にするため、Connector Xpress では、常に必要に応じて、Date/DateTime/Time のいずれにもマッピングできるようになっています。

Double-precision floating-point

倍精度 64 ビット浮動小数点値を指定します。

Enumeration - enumeration type name

列挙された値の固定リストがある属性を指定します。

Flexi-DN

識別名の文字列形式を

「cn=Bob,ou=Sales,o=ExampleCorp」のように指定します。 これは、コネクタによって適用されます。

Flexi-Email

電子メール アドレスの文字列形式を指定します。

Flexi-Quoteless

引用符を属性値から削除するように指定します。

Floating Point

単精度 32 ビット浮動小数点数を指定します。

Integer

-2147483648 ～ 2147483647 の 32 ビット値を指定します。

Long Integer

9223372036854775808 ～ 9223372036854775807 の 64 ビット値を指定します。

String

無制限のフィールドを指定します。

Time

0 秒 ～ 23:59:59 のオフセットを指定します。

例： 13:20:00

Force Case

文字列のコネクタ対話値の大文字/小文字を指定します。

Upper

文字列が大文字に設定されます。

例： 大文字に設定すると、*String* は *STRING* になります

Lower

文字列が小文字に設定されます。

例： 小文字に設定すると、*STRING* は *string* になります

Preserve

文字列の大文字/小文字が保持されます。

注: エンドポイントの検索では大文字と小文字を区別しない場合があります。したがって、「EXAMPLE」を検索すると、今までどおりエンドポイント システムの「eXaMpLe」が一致します。

Multi-valued

オンにすると、この属性が複数値であることを示します。

JNDI： ネイティブ属性が複数値の場合にのみ選択されます。 ネイティブ属性が単一値の場合は、編集できません。

JDBC： デフォルトはオフです。

Flattening Style

（JDBC のみ）使用する平坦化スタイルを指定します。

CSV (comma-separated value)

属性の複数値が区切り値であることを指定します。

SQL

この属性の複数値が SQL 形式であることを指定します。

XML

この属性の複数値が XML 形式であることを指定します。

Trim Whitespace

Connector Xpress によって属性の値の先頭または末尾の空白を切り捨てるかどうかを指定します。

Sensitive

属性の値のマスクにユーザ インターフェースでアスタリスクを使用するかどうかを指定します。

Boolean Values

デフォルトの 0 と 1 のシンボリック値をブール属性タイプの false と true の値で上書きします。

True

true に使用されるシンボリック値を指定します。

False

false に使用されるシンボリック値を指定します。

Value Conversion

isStoreSymbolic または isStoreNumeric のメタデータ値を設定します。

（デフォルト）Pass-through

Boolean（および enum）の値が数値相当物として書き込まれることを指定します。

Symbolic

LDAP で数値のテキストが要求されるが、同等のシンボル値がコネクタ対話で格納されることを指定します。 したがって、ブール値および enum がシンボル値として書き込まれることを指定します。

Numeric

Boolean（および enum）の値が数値相当物として書き込まれることを指定します。つまり、boolean と enum 属性は、数値として書き込まれます。

注: このオプションが使用できるのは、ブール値と（静的）列挙データ型の場合のみです。両方のブール値フィールドを設定すると、この値は Symbolic に変更されます。boolValues の設定は影響しません。

Account Template Value

この属性のデフォルト値がデフォルト アカウント テンプレートで使用および生成される方法を指定します。

None

デフォルト アカウント テンプレートの中にこの属性の値がないことを指定します。

By Value

デフォルト アカウント テンプレートで使用される属性の値を指定します。

By Rule String

デフォルト アカウント テンプレートで使用される属性の値がルール文字列によって生成されることを指定します。

注: ルール文字列の詳細については、「CA Identity Manager 管理ガイド」を参照してください。

Synchronized

このアカウントがテンプレートと同期された場合に、この属性が更新されるかどうかを指定します。

Cached

属性がプロビジョニング サーバによって格納されエンドポイント システムを参照せずに取得できることを指定します。

たとえば、エンドポイント システムに接続するための認証情報はキャッシュされる必要があります。キャッシュされていないと、コネクタが取得されるたびに、ユーザは認証情報を入力する必要がある場合があります。 また、エンドポイント システムに値を問い合わせると処理に長時間要する場合があるので、値をキャッシュする必要がある場合があります。

Well Known Name

この属性が *well-known* 属性かどうかを指定します。

JavaBean Property Name

JIAM API でこの属性に使用される JavaBean プロパティ名を指定します。［Well-Known Name］リスト内の値を選択した場合、この値は手動で編集できません。

Expensive Retrieval

この属性の値は負荷が大きすぎるため、標準検索では返すことができない場合があることを指定します。 オンにすると、明示的に要求された場合にのみ、属性値が返されます。

Converter Regex

この機能は動作しません。 正規表現を使用してデータを変換する場合は、変換する属性用の flexi 文字列タイプを新たに作成し、上書きコネクタ XML ファイルのコネクタ タイプに追加します。

UI Field Length (String types only)

この属性のユーザ インターフェース入力フィールドの推奨文字幅を指定します。 使用しない場合は、Connector Xpress では、属性データ型および最大長を使用して適切な入力フィールド長を定義します。

Connector Generator

CA IAM CS によって使用されるジェネレータ式を指定します。

値を文字としてエンドポイント システムに渡すには、式を二重引用符で囲みます。

例： ¥"NEXT VALUE FOR my_sequence¥"

式を二重引用符で囲まない場合、値を解釈する方法をコネクタが知っている必要があります。 たとえば、以下の場合、JDBC は、渡された値がシーケンス名であることがわかります。

例： ¥'my_sequence¥'

Is Connector Generated

エンドポイントによってプロパティの値が暗示的に生成されることを指定します。 たとえば JDBC 内の IDENTITY 列の *true* が該当します。

Extended Properties

詳細なメタデータ プロパティを表示します。 これらのフィールドは、［Connector Xpress Preferences (P. 197)］ダイアログ ボックスで［Show Extended Metadata (P. 122)］を選択すると表示されます。

注: 詳細については、「拡張メタデータのプロパティ」を参照してください。

[Class Associations]ダイアログ ボックス

［Class Associations］ダイアログ ボックスでは、直接的および間接的に関連付けを作成するクラスを指定できます。

このダイアログには以下のフィールドが含まれます。

Create direct association with

直接的な関連付けを作成するクラスを指定します。

Create indirect association with

（JDBC のみ）関節的な関連付けを作成するクラスを指定します。

associated with

関連付けられたクラス名を示します。

Association attribute

関連付けの概要を表示します。

Remove

マッピング ツリーから関連付けを削除します。

[Connector Server Details]ダイアログ ボックス

ダイアログ ボックスを使用して、スタンドアロンの CA IAM CS を使用して、コネクタを管理する、コネクタ サーバを設定できます。

このダイアログには以下のフィールドが含まれます。

Host Name

コネクタ サーバが存在するホストの名前を定義します。

Port

CA IAM CS への接続に使用されるポートを定義します

Bind DN

DN 形式でコネクタ サーバを接続するユーザ名を定義します。

Use TLS

コネクタ サーバに安全な TLS 接続が確立されることを指定します。つまり、パスワードはクリア テキスト形式で転送されません。選択しないと、パスワードはクリア テキスト形式で転送されます。

デフォルト： オン

Key

コネクタがコネクタ サーバとの接続に使用する設定オプションの名前を定義します。

Value

設定オプションの値を定義します。

[Key]列

コネクタがコネクタ サーバとの接続に使用する設定オプションの完全なリストを定義します。

[Value]列

［Key］列で表示された設定オプションに対応する値を表示します。

[UI from TM. Not exist in glossary]ダイアログ ボックス

［Create New Endpoint Type］ダイアログ ボックスでは、エンドポイント タイプの名前を定義し、またオプションでエンドポイント タイプを管理するために使用するコネクタ サーバの名前を定義できます。

このダイアログには以下のフィールドが含まれます。

New Endpoint Type

作成するエンドポイント タイプの名前を定義します。

Connector Xpress では、［Define Endpoint］ダイアログ ボックスに入力された値に基づいたエンドポイント タイプの名前が自動的に提供されます。デフォルトを受け入れるか、または別の名前を入力できます。

注: 名前に含まれるスペースは、アンダースコアに変換されます。

Connector Servers

新しいエンドポイント タイプを管理する、コネクタ サーバを指定します。

詳細情報:

エンドポイント タイプの作成 (P. 113)

[Create New Endpoint]ダイアログ ボックス

［Create New Endpoint］ダイアログ ボックスでは、JDBC 用および JNDI 用のコネクタのエンドポイントを作成できます。

このダイアログには以下のフィールドが含まれます。

LDAP URL（**必須**）

（JNDI のみ）LDAP の URL を指定します。

Base DN（**必須**）

（JNDI のみ）ディレクトリ エンドポイントで管理する DIT（ディレクトリ情報ツリー）の最上位を指定します。

Use TLS（**必須**）

（JNDI のみ）CS およびディレクトリ エンドポイントの間で TLS 接続を使用することを指定します。

LDAP Version（**必須**）

LDAP バージョンを定義します。

注: このフィールドは、この属性が *isRequired =true* に設定されている既存のエンドポイントにのみ表示されます。 JNDI メタデータでは、この属性はデフォルトで Connector Xpress によって *isRequired* が false に設定されるので、Connector Xpress を使用して作成された新しいプロジェクトでは表示されません。

Endpoint Name（**必須**）

（JDBC と JNDI）エンドポイント名を定義します。エンドポイントを説明した名前を使用することをお勧めします。

Password（**必須**）

（JDBC と JNDI）ターゲット エンドポイントにログインするために使用されるパスワードを指定します。

Bind User DN（**必須**）

（JNDI のみ）ディレクトリ エンドポイントをバインドする、ユーザの識別名を指定します。

Connection URI（**必須**）

（JDBC のみ）セキュア接続 URI を指定します。

Username（**必須**）

（JDBC と JNDI）ターゲット データベースにログインするために使用されるユーザ名を指定します。

詳細情報:

コネクタの展開 (P. 44)
エンドポイントを取得および管理する方法 (P. 111)
エンドポイントの取得、検索、および関連付け (P. 112)

[Create Operation Binding]ダイアログ ボックス

このダイアログ ボックスでは、操作バインディング情報を指定できます。

このダイアログには以下のフィールドが含まれます。

Available object classes

操作バインディングを作成できるオブジェクト バインディングを表示します。

Added object classes

操作バインディングを適用するオブジェクト クラスを指定できます。

Stored Procedure

（JBDC のみ）操作バインディングが JDBC ストアド プロシージャ タイプに設定されることを指定します。

Script

操作バインディングがスクリプト スタイル タイプに設定されることを指定します。

Available Operations

この操作バインディングをバインドできる利用可能な操作を指定します。

Add

管理されたシステム上でオブジェクトを作成します。

Modify

管理されたシステム上のオブジェクトを変更します。

Delete

管理されたシステム上のオブジェクトを削除します。

Modify-Rn

管理されたシステム上のオブジェクトの名前を変更します。

Move

管理されたシステム上でオブジェクトをあるロケーションから別のロケーションに移動します。

Activate

コネクタを有効化して、外部からも見えるようにします。 コネクタの初期化が完了し、リクエストを受信する準備ができると、このメソッドが呼び出されます。

Deactivate

コネクタを無効化して、外部から見えなくします。 コネクタが不要になると、このメソッドが呼び出され、コネクタが保持する接続またはリソースがすべて整理されます。

Lookup

管理されたシステム上のオブジェクトをルックアップします。

Search

管理されたシステム上でオブジェクトを検索します。

Move-Assocs

管理されたシステム上でターゲット オブジェクトが移動された後、関連付け属性を更新します。

Modify-Rn-Assocs

管理されたシステム上でターゲット オブジェクトの名前が変更された後、関連付け属性を更新します。

Delete- Assocs

管理されたシステム上のターゲット オブジェクトを参照する関連付けをすべて削除します。

Add-Attr-Assocs

管理されたシステム上のターゲット オブジェクトに関連付けを追加します。

Remove-Attr-Assocs

管理されたシステム上のターゲット オブジェクトから関連付けを削除します。

Lookup-Assocs

管理されたシステム上のターゲット オブジェクトの関連付けをすべてルックアップします。

Search-Assocs

管理されたシステム上のターゲット オブジェクトの関連付けをすべて検索します。

Modify-Update-Attrs-Assocs

管理されたシステム上のターゲット オブジェクトに加えられた変更に基づいて、逆の関連付けメンバシップ リストを更新します。

Assoc-Update-References-To

管理されたシステム上のターゲット オブジェクトを参照する、逆の関連付けポインタをすべて更新します。

Assoc-Search-For-References-To

ターゲット オブジェクトを参照する関連付けクラスの各インスタンスの検索結果をコネクタ対話で返します（ターゲット オブジェクトおよび関連付けを指定）。

Post-Query-Attributes

LOOKUP によって返された属性および SEARCH によって返された各結果に対して実行されます。この場合、検索結果を引き続きストリームすることができます。

Post-Query-Search-Result

各検索結果が返される前に実行されます。

Timing

操作バインディングが実行されるタイミングを指定します。

Before

指定された LDAP 操作の前にコネクタによって操作バインディングが実行されるように指定します。

After

指定された LDAP 操作の後にコネクタによって操作バインディングが実行されるように指定します。

Instead Of

指定された LDAP 操作の代わりにコネクタによって操作バインディングが実行されるように指定します。

注: オブジェクト クラスを選択しないと、操作バインディングはマッピングされたオブジェクト クラスすべてに適用されます。

詳細情報:

Operation Bindings (P. 69)

[Custom Types]ダイアログ ボックス

［Custom Types］ダイアログ ボックスでは、属性マッピングでメタデータ タイプとして使用する任意の文字列（フォーマットされた文字列タイプ）を定義できます。 ［Map Class and Attributes］ダイアログ ボックスの［Type］リストでユーザがマッピングできる属性タイプとして定義した文字列が表示されます。このダイアログ ボックスでは、属性の静的列挙および動的列挙も定義できます。

一部の事前定義されたフォーマット済み文字列タイプ（電子メール、DN、および引用符なし）は CA IAM CS フレームワークにすでに知られているので、デフォルトで使用できます。これらのタイプのハンドラは、コネクタ サーバ上にあります。

注: 新しいフォーマット済み文字列タイプ コードの作成およびカスタム タイプを追加でサポートするのに必要な手順については、「Connector Programming Guide」を参照してください。

このダイアログには以下のフィールドが含まれます。

Formatted String Types

属性マッピングでメタデータ タイプとして使用できるカスタム文字列をリスト表示します。

Name

定義されたフォーマット済み文字列タイプの名前を表示します。

DN

識別名の文字列形式を

Quoteless

引用符を属性値から削除するように指定します。

Email

電子メール アドレスの文字列形式を指定します。

Add

［Formatted String Types］リストに新しいカスタム文字列を追加できます。

Remove

［Formatted String Types］テーブルからカスタム文字列タイプを削除します。

Enumerated Types

属性として使用する静的な列挙型を定義します。

Name

定義した静的な列挙型の名前を表示します。

Suspended

アカウントの一時停止の状態を定義します。

Endpoint Groups

［Accounts］画面上の［Endpoint］画面タブおよびページ セクションの列挙 ID を定義します。

Groups

［Account］画面のタブおよびページ セクションの列挙 ID 定義します。

Add

新しい列挙型を追加します。

Remove

選択された列挙型を削除します。

Values

選択された列挙型の値を表示します。

Value

エンドポイント システムで使用される、列挙型の値を定義します。

［Display Name］

（オプション）CA Identity Manager ユーザ コンソールおよびプロビジョニング マネージャで表示される、列挙型の名前を定義します。

Ordinal

（オプション）列挙値の順序を定義します。

Add

新しい列挙型の値を定義できます。

Remove

以前に定義された列挙型の値を削除します。

Compound Type Classes

別のクラスのコンテンツをプロビジョニング属性にマッピングします。

Name

追加した複合クラスの名前を表示します。

Add

マッピング ツリーに複合クラスを追加します。

Remove

マッピング ツリーから複合クラスを削除します。

[Direct Association]ダイアログ ボックス

［Direct Association］ダイアログ ボックスでは、任意の 2 つのクラスのオブジェクト間で直接関連付けを指定できます。 このタイプの関連付けでは常に、*from Class1 to Class2* のような形式で方向があります。

また、このダイアログ ボックスを使用して、同じクラス間で反対方向に関連付けをマッピングする逆の関連付けを確立できます。

ほとんどの双方向関連付けは 1 つのクラスに物理属性があり、もう一方のクラスに仮想属性があります。最初に物理関連付け属性を定義することをお勧めします。

このダイアログには以下のフィールドが含まれます。

Physical Attribute

関連付けされる物理属性を指定します。属性がネイティブ システムに存在するため、物理的です。

Virtual Attribute

関連付けの一部を構成するが、ネイティブ システムに存在しない属性を指定します。 属性は CA IAM CS によって計算されます。 ネイティブ システムでは永続化されません。

memberof は JNDI の一般的な仮想属性の例です。ネイティブ システムはグループからアカウントへの一方向のみの関連付けを格納します。仮想属性により、アカウントからグループへの関連付けを作成できます。

By Attribute

ソース クラスの関連付け属性が参照するターゲット クラスの関連付け属性を指定します。

Value Template

比較するフォーマットされた属性値を指定します。 たとえば、場合によっては、ターゲット クラスの関連付け属性は、ソース クラスの関連付け属性の値より複雑になります。 デフォルトでは、非 DN 関連付け属性は「to」クラスの名前または代替キーの値に完全に対応する値を持つと仮定されています。 関連付け属性にいくつかの内部構造がある場合、関連付けの作成時に、構造を扱う補助としてテンプレート フィールドを使用できます。

テンプレート値には、${名前} または ${dn} 文字列を格納できます。 関連付けの作成時に、${名前} は関連するオブジェクトの単純名と置換され、${dn} はその完全な DN と置換されます。

フィルタ別

ターゲット クラスの属性に一致するために使用する特定の LDAP フィルタを指定します。 これは、実行時に ${attributeName} 形式の文字列が *from* クラスから対応する属性の値に置換される、LDAP 検索フィルタとして指定されます。

Use as a Base Association

既存の関連付けを選択し、既存の関連付けに基づいた結果を加工する新規の関連付けの結果を取得できます（ネストにより表示される関連付け結果を追加など）。

Objects Must Exist

関連付けにオブジェクトを追加する前に参照されたオブジェクトが存在することをコネクタ サーバが確認するように指定します。 このチェック ボックスは、逆の関連付けにも適用されます（ある場合）。つまり、このチェック ボックスは、ソースとターゲットの両方のクラスに対して双方向に適用されます。

Use DNs in Attributes

ネイティブ システムで永続化された値を指定します。 オンにすると、完全なネイティブ DN 値が格納されることをフラグします。 たとえば myaccount など DN からの命名属性値ではなく ou=myorgunit、cn=myaccount のようになります。

Association is Nested

グループ クラス間の関連付けがネストされるかどうかを指定します。たとえば、単一の関連付け定義で「グループのグループのグループ」関係を満たすことができます。

Include a Reverse Association

ダイアログ ボックスの逆の関連付け部分を表示します。

[Edit Connector Server Configuration]ダイアログ ボックス

［Edit Connector Server Configuration］ダイアログ ボックスでは、サーバのホスティング情報、ルーティング情報、管理コネクタ情報を設定できます。

このダイアログには以下のフィールドが含まれます。

Connector Server Host Name

コネクタ サーバを実行するコンピュータのホスト名を定義します。

Port

非 TLS 接続に使用される、コネクタ サーバの TCP ポートを定義します。

TLS Port

TLS 接続に使用される、コネクタ サーバの TCP ポートを定義します。

Use TLS

プロビジョニング サーバとコネクタ サーバの間の接続に暗号化された TLS 接続を使用することを指定します。指定しない場合、プロビジョニング サーバとコネクタ サーバの間の接続にクリア テキストが使用されます。

プロビジョニング サーバ（オプション）

操作プロビジョニング サーバをプロビジョニング サーバ ツリーで選択されているプロビジョニング サーバではなく、ユーザが指定できます。

Password

コネクタ サーバへのアクセスに使用するパスワードを定義します。通常このパスワードは、サーバのインストール時に設定します。

注：［Connector Server Host Name］、［Ports］または［Use TLS］フィールドを編集する場合、パスワードを再入力します。

Make this the default CS

デフォルトでプロビジョニング サーバがこのコネクタ サーバを使用するように指定します。

注: 明示的に CS Config と関連付けられていない、ブランチ（エンドポイントタイプまたはエンドポイント）は、デフォルト CS を介してルーティングされます。 その結果、管理コネクタ サーバを明示的に指定せずに、エンドポイント タイプまたはエンドポイントを作成すると、不一致が発生する場合があります。たとえば、デフォルト CS が C++ コネクタ サーバで、管理コネクタ サーバとして明示的に CA IAM CS を指定せずに、動的エンドポイント タイプを作成したときです。

重要: 単一のプロビジョニング サーバにある複数のコネクタ サーバに対してこのチェック ボックスをオンにできます。このチェック ボックスをオンにすると、リクエストがルーティングされるとき、CS がランダムに選択されます。

Object Handle

選択されたプロビジョニング サーバが管理しているエンドポイント タイプのオブジェクト ハンドルを指定します。

注: Set Managing コネクタ サーバ手順を使用して、このリストを管理することをお勧めします。

詳細情報:

コネクタ サーバ設定の編集 (P. 100)
コネクタ サーバ設定の作成 (P. 98)
管理コネクタ サーバを設定する方法 (P. 99)

[Edit Metadata for Endpoint Type]ダイアログ ボックス

［Edit Metadata for Endpoint Type］ダイアログ ボックスでは、メタデータを手動で編集できます。

このダイアログには以下のフィールドが含まれます。

Node Type

［Add］ボタンで追加できるノード タイプのリストを表示します。 リストの内容は、XML ツリーで選択されているノードのタイプによって異なります。

Add

［Node Type］ドロップダウン リストで指定されたタイプの新規ノードを XML ツリーに追加します。

Edit

選択されているノードを XML ツリーで編集します。

Remove

選択されているノードを XML ツリーから削除します。

XML Tree

編集可能なメタデータの XML ツリー ビューを表示します。

Export

XML ファイルにデータ モデルをエクスポートします。

詳細情報:

メタデータの編集 (P. 117)

[Edit Project Settings]ダイアログ ボックス

［Edit Project Settings］ダイアログ ボックスでは、プロジェクトの設定を編集できます。

このダイアログには以下のフィールドが含まれます。

Project File Comment

プロジェクトの自由形式のコメントを定義します。

Metadata Dictionaries

事前定義済みカスタム動作属性を 1 セット ロードするのに使用できる辞書を指定します。

[Edit Script]ダイアログ ボックス

このダイアログ ボックスでは、スクリプトのパラメータを指定できます。

Script Language

CA IAM CS フレームワークで現在サポートされている利用可能なスクリプト言語を指定します。

詳細情報:

スクリプトへのバインド操作 (P. 73)
スクリプト (P. 70)

[Edit Source]ダイアログ ボックス

［Edit Source］ダイアログ ボックスでは、データ ソースの名前および接続詳細を指定できます。 JDBC データ ソースの場合、このダイアログ ボックスで、選択したデータベースに合った JDBC URL を生成します。 JNDI データ ソースの場合、このダイアログ ボックスで、LDAP 接続の詳細を指定できます。

注: このダイアログ ボックスのフィールドは、選択されたデータ ソースのタイプによって異なります。

このダイアログには以下のフィールドが含まれます。

Name

データ ソースの名前を指定します。

注: データ ソースの使用目的を明確に説明した名前を選択することをお勧めします。

Server Name

アクセスするサーバのホスト名を指定します。

Host Name

（Informix Dynamic Server）Informix サーバの名前を指定します。

Database Type

（JDBC のみ）データベースの接続タイプを指定します。

- DB2
- DB2 for IBMi
- Informix Dynamic Server
- Microsoft SQL Server
- MySQL
- Oracle
- Sybase Adaptive Server Enterprise
- その他

Username

（JDBC のみ）ターゲット データベース エンドポイントのログオンにコネクタが使用するユーザ名を指定します。

デフォルト設定：

Oracle： System

Microsoft SQL Server： sa

DB2： db2admin

Native

（JDBC および MS SQL Server のみ）Windows の MSSQL ネイティブ認証をアクティブ化することを指定します。接続に使用される JDBC URL に以下が追加されます。

```
integratedSecurity=true
```

Server name

（JDBC のみ）データベース サーバのホスト名を指定します。

SID

（JDBC および Oracle のみ）Oracle の SID を指定します。

Port

コネクタがエンドポイントとの通信に使用するポート番号を指定します。

デフォルト

JNDI： 389

Oracle： 1521

Microsoft SQL Server： 1433

DB2： 6789

Ingres： II7

Informix Dynamic Server： 9088

MySQL： 3306

データベース

（JDBC のみ）データベース名（MS SQL を含めてほとんどのベンダー）またはその SID（Oracle）を定義します。

デフォルト

Microsoft SQL Server： master

Oracle SID： ORCL

JDBC URL

（JDBC のみ） JDBC の URL を定義します。 以前のフィールドに入力すると、URL が Connector Xpress によって自動的に生成されます。

注: このフィールドを編集するには、［Edit］チェック ボックスをオンにします。

Bind DN

（JNDI のみ）ディレクトリ エンドポイントのバインドに使用したユーザの識別名を定義します。

Anonymous

（JNDI のみ）バインド DN が使用されず、代わりに、ディレクトリ エンドポイントに匿名バインドが使用されることを指定します。

Use TLS

Connector Xpress がエンドポイントとの接続に TLS を使用することを指定します。

Base DN

（JNDI のみ）ディレクトリ エンドポイントで管理する DIT （ディレクトリ情報ツリー）のトップ レベルを定義します。

Edit

（JDBC のみ）［JDBC URL］フィールドを編集します。

Test

指定した情報を使用して、指定のディレクトリまたはデータベースへの接続を試行します。

詳細情報:

プロジェクトの作成 (P. 34)
データ ソースの設定 (P. 27)

[Endpoint Class]ダイアログ ボックス

［Endpoint Class］ダイアログ ボックスでは、属性マッピング テーブル内で検出された属性を選択するのではなく、マッピングする新規属性を作成できます。また、属性マッピング テーブルに辞書から事前定義済み属性のセットをロードできます。

このダイアログには以下のフィールドが含まれます。

Name

マッピングするクラスの名前を定義します。

制限： 先頭は英字のみ使用できます。

Description

マッピングするクラスを説明します。

Connector Map To

インコネクタ対話にマッピングするオブジェクト クラス（コネクタ自体を含めて）または属性をマッピングする名前を指定します。 動的コネクタの場合、この属性によって、属性をマッピングするネイティブ システム アイテムの名前が指定されます。 たとえば、JDBC ベースのコネクタの場合、データベース テーブルの名前にオブジェクト クラスをマッピングし、その内の各プロパティを列のいずれかの名前にマッピングします。

注: Endpoint クラスをマッピングする場合、Endpoint クラスはネイティブ オブジェクトにマップされていないため、このフィールドを *connector* に設定します。

Managed

オフにすると、関連付けを確立する目的でのみこのクラスがマップ済みとしてマークされます。 その結果、Connector Xpress では名前とタイプのみがマップされます。非管理対象クラスのインスタンスはリスト表示して他のオブジェクトと関連付けることができますが、作成、編集、削除できません。

複合クラスの場合は、Connector Xpress によってこのフィールドはデフォルトでオンにされ、オフにできません。

Add Dictionary Attribute

事前定義済みカスタム動作属性のセットを辞書からロードして、マッピング ツリーで属性を表示します。

注: このフィールドは、辞書から動作属性をロードすることを指定すると、利用可能になります。

Extended Properties

詳細なメタデータ プロパティを表示します。 これらのフィールドは、［Connector Xpress Preferences (P. 197)］ダイアログ ボックスで［Show Extended Metadata (P. 122)］を選択すると表示されます。

注: 詳細については、「拡張メタデータのプロパティ」を参照してください。

[Enter Password for Data Source]ダイアログ ボックス

［Enter Password for Data Source］ダイアログ ボックスで、接続の基礎になるデータ ソースのパスワードを指定します。

このダイアログには以下のフィールドが含まれます。

Source name

Connector Xpress がデータ ソースを作成するときに指定された、データ ソース名を表示します。

Source URL

データ ソースの接続 URL を表示します。

Source type

データ ソースのタイプ（JNDI、JDBC など）を表示します。

User name

データ ソースの接続に使用されるユーザ名を表示します。

Password

データ ソースの接続に Connector Xpress が使用するユーザのパスワードを指定します。

詳細情報:

プロジェクトの作成 (P. 34)
プロジェクトの編集 (P. 39)

[Endpoint Type Details]ダイアログ ボックス

［Endpoint Type Details］ダイアログ ボックスでは、定義するエンドポイント タイプの名前、説明およびバージョン番号を指定できます。

このダイアログには以下のフィールドが含まれます。

Name

（必須）エンドポイント タイプ名を指定します。

注: 名前には、#、/、¥、[、]、;、:、|、=、,、+、*、?、<、>、@、（、） を使用できません。また、スペースはすべてアンダースコアに変換されます。

いずれかの文字を入力すると、Connector Xpress によって警告が表示されます。既存のメタデータからこの文字を削除することをお勧めします。

Description

エンドポイント タイプの詳細な説明を指定します。

Version

エンドポイント タイプのバージョン番号を指定します。バージョンを指定すると、増分変更を記録するように Connector Xpress プロジェクト ファイルを変更するときに、役立つ場合があります。バージョンを指定しても、動的コネクタに論理的な影響はありません。

Extended Properties

詳細なメタデータ プロパティを表示します。 これらのフィールドは、［Connector Xpress Preferences (P. 197)］ダイアログ ボックスで［Show Extended Metadata (P. 122)］を選択すると表示されます。

注: 詳細については、「拡張メタデータのプロパティ」を参照してください。

［Explore/Correlate Endpoint］ダイアログ ボックス

［Explore/Correlate Endpoint］ダイアログ ボックスでは、エンドポイントを検索および関連付ける方法を指定できます。

このダイアログには以下のフィールドが含まれます。

Explore Endpoint first endpoint

未検出のアカウントおよびグループを検索するのにこのエンドポイントが検索されるかどうかを指定します。

Sub Tree

サブツリーのコンテナすべてがエンドポイントの検索に含まれるように指定します。

One Level

エンドポイントの検索にコンテナの 1 レベルのみが含まれるように指定します。

Correlate Accounts with Users

グローバル ユーザ アカウントをエンドポイント システム上のアカウントおよびグループ内の情報に関連付けるかどうかを指定します。

Use existing Users

関連付けによって、各アカウントが同じ名前のユーザにリンクされるように指定します。 関連付けでは既存ユーザのみリンクされ、ユーザは作成されません。

Create Users as needed

関連付けによって、各アカウントが同じ名前を共有するユーザにリンクされるように指定します。 ユーザが存在しない場合、関連付けプロセスによってユーザが作成されます。

詳細情報:

コネクタの展開 (P. 44)
エンドポイントを取得および管理する方法 (P. 111)
エンドポイントの取得、検索、および関連付け (P. 112)

Extended Metadata Properties

以下のフィールドは、［Connector Xpress Preferences (P. 197)］ダイアログ ボックスで［Show extended set of metadata properties (P. 122)］を選択すると Connector Xpress のダイアログ ボックスに表示されます。

Connector Xpress には、以下のダイアログ ボックス上にフィールドが表示されます。

- ［Endpoint Type Details］ダイアログ ボックス（P. 166）
- Map Class and Attributes （クラスと属性のマッピング） （JDBC）
- Map Class and Attributes （クラスと属性のマッピング） （JNDI）
- ［Map Compound Class and Attributes］ダイアログ ボックス（P. 180）
- ［Attribute Details］ダイアログ ボックス
- ［Endpoint Class］ダイアログ ボックス（P. 164）
- ［Map Container Class］ダイアログ ボックス（P. 185）

注: 一部のフィールドは、ダイアログ ボックスに表示されない場合があります。

Assoc Obj Key Attr

間接関連付けプロパティをターゲットにします。 ターゲット プロパティの親クラスを命名するのに使用される代替キーです（参照クラスではない）。

Assoc Ref Object Class

あるオブジェクト タイプから参照先オブジェクト タイプの関係を定義します。属性を使用した直接的な関連付けと両方のオブジェクト タイプの外にあるテーブルを使用した間接的な関連付けがサポートされています。

Assoc Reverse Attr

直接的な関連付けプロパティすべてがターゲットになり、逆方向に関連付け情報が含まれる、参照されたクラス（つまりターゲット プロパティの親ではない）内の属性に対して命名されます。

Assoc Table

間接的な関連付けプロパティすべてがターゲットになり、関連付けリンクを格納するテーブルを命名します。

Assoc Table Obj Column

間接的な関連付けプロパティすべてがターゲットになり、ターゲット プロパティの親であるクラスへの参照が含まれる ${MD_ASSOC_TABLE} で指定されたテーブルで列を指定します。

Assoc Table Ref Column

間接的な関連付けプロパティすべてがターゲットになり、参照クラス（すなわち親クラスではないターゲット プロパティ）への参照が含まれるス ${MD_ASSOC_TABLE} で指定されたテーブルで列を指定します。

Assoc Type

true の場合、ターゲット属性によって、親オブジェクトとそのコンパウンド値の子オブジェクトの間の関連付けが指定されます。

Connector Generated Override

LDAP ADD リクエストで生成された属性値を上書きする試行を処理する方法を指定します。

Connector Generator

プロパティに割り当てられる値を指定するジェネレータ名（または、¥" で囲まれている場合、エンドポイントに渡されるリテラル式）

例： JDBC の ¥'my_sequence¥' または ¥'¥"NEXT VALUE FOR my_sequence¥"¥'

Connector Map To

イン コネクタ対話にマッピングするオブジェクト クラス（コネクタ自体を含めて）または属性をマッピングする名前を指定します。 動的コネクタの場合、この属性によって、属性をマッピングするネイティブ システム アイテムの名前が指定されます。 たとえば、JDBC ベースのコネクタの場合、データベース テーブルの名前にオブジェクト クラスをマッピングし、その内の各プロパティを列のいずれかの名前にマッピングします。

Connector Map To Alias

インコネクタ対話にマッピングするオブジェクトクラス（コネクタ自体を含めて）または属性をマッピングする名前を指定します。 たとえば MD_CONN_MAP_TO が複雑な表現にマッピングされ、そのため、コネクタのコード内で役立つ参照名として機能しない場合などです。

Connector Map To Ambiguous

単一の LDAP 属性が 1 つより多いコネクタ オブジェクト クラスまたは属性にあいまいなマッピングするまれな場合を指定します。 明示的な選択が行われず、新規オブジェクトが作成される場合、最初の選択肢がデフォルトになるので、選択の順序は重要です。

Connector Map To Ambiguous Choice Attr

あいまいなマッピングの属性がオプションで別の LDAP 属性を指定できます。この属性は、オブジェクト クラスの新規インスタンスが追加される際、明示的な選択を受け取るために使用されます。

Connector Map To Auxiliary

マッピングが存在する名前によって補助クラスをすべて識別します。

Connector Map To Derived

名前に基づいてマッピングがある構造クラスを継承するすべてのクラスと一緒にすべて識別します。

Connector Map To Lax

MD_CONN_MAP_TO マッピングが *lax* であることを指定します。 MD_CONN_MAP_TO が指定されたすべてのオブジェクト クラスで Boolean を使用できます。

Connector Map to Multiple

複数のコネクタ対話クラスまたは属性クラスを指定します。 単一の LDAP クラスまたは属性を複数のコネクタ対話属性に分割するか、複数のコネクタ対話属性の値から生成する場合に使用されます。

Connector Map To Same

この属性に対して定義されたプロパティがすべてそれらのコネクタ対話名としてそれらの LDAP 名をリテラルに使用します。 この属性は、MD_CONN_MAP_TO を指定しないオブジェクト クラスに表示されます。 MD_CONN_MAP_TO またはこの属性のいずれもないオブジェクト クラスはスキップされます。

Connector Map To Strict

存在用の明示的な確認を含めるフレームワークのオブジェクト クラス フィルタを増大して、特定のコネクタ対話命名属性が存在するか明示的に確認します。 これは、命名属性が潜在的にあいまいで、1 つの選択肢のみをマッピングするときに使用されます。

Connector XML

この名前空間にあるすべてのコネクタに使用される connector.xml 設定の動的なバージョンが含まれます。

Connector Map to Derived

名前に基づいてマッピングがある構造クラスを継承するすべてのクラスと一緒にすべて識別します。

Default Value

（オプション）属性のデフォルト値を指定します。

注: この値はクライアント（ユーザ インターフェースなど）のみが使用します。 クライアントによって値が提供されない場合、デフォルトではプロビジョニング サーバは値を使用しません。

Implementation Bundle

コネクタ タイプによってターゲットにするコネクタ実装の名前を定義します。

Is Compound Value

オンにすると、複合式または構造型属性によって受け入れられる値の形式をターゲット クラスが定義します。

Is Connection

コネクタの接続関連の属性を区別します。

Is Connector Generated

エンドポイント（たとえば JDBC 内の IDENTITY 列の true）によってプロパティの値が暗示的に生成されることを指定します。

Is Interesting to Compliance

属性が CA Role and Compliance Manager に関連しているものとしてマークされるように指定します。 たとえば、コンプライアンス関連としてマークされたアカウント オブジェクトの属性は、分析のリソースとして CA Role and Compliance Manager で利用できます。 通常、これらの属性は、エンドポイント システムの許可または権限付与の割り当てを示します。

Is Hidden

プロパティの値が GUI で表示されないことを指定します。

Is Naming

属性が LDAP 命名属性かどうかを指定します。

Is Operational Attribute

ターゲット プロパティが操作可能かどうかを（つまり、保持されたものではなくネイティブ エンドポイントによって計算）指定します。

Is Read Only Native

プロパティがエンドポイント システム上で読み取り専用かどうかを指定します。

isTransactionsEnabled

（DB2 IBM i のみ）JDBC エンドポイントで操作（Add、Delete、Modify など）を実行するとき、トランザクションが使用されるかどうか制御します。 記録されない DB2 IBM i データベース テーブルに対してトランザクションを使用できない場合に適しています。

Is Well Known

このプロパティが共通のセットのメンバとして認識されるかどうかを指定します。

Metadata Item Name

（読み取り専用）メタデータ XML 内のオブジェクトの名前の値を定義します。 クラスと属性には通常、LDAP プロビジョニング名が使用されます。

Virtual

ターゲット プロパティを *persisted* ではなく、*computed* としてフラグします。

[Indirect Association]ダイアログ ボックス（JDBC のみ）

［Indirect Association］ダイアログ ボックスを使用して、任意の 2 つのオブジェクト クラスを双方向に関連付けを指定できます。 たとえば、2 つのオブジェクト間の関連が双方向であり、オブジェクト間の関連リンクを保持する、テーブルなどの 3 番目のエンティティに含まれている場合です。

エントリはそれぞれ、各オブジェクトのインスタンス間の関連付けを示します。関連付けを表すノードは、マッピング ツリーの［Associations］ノードの下に表示されます。

3 番目のエンティティで関連付けを格納した場合の対称的な性質により、どちらの方向のルックアップ速度も同じで、逆の関連付けは自動的に継承および作成されます。

このダイアログには以下のフィールドが含まれます。

スキーマ

［Membership Table］フィールドで表示されるテーブルのスキーマ名（たとえばテーブルスペースまたはデータベース）を指定します。

Membership Table

各グループのメンバを定義するテーブルを指定します。

注: 選択するデータベース テーブルは少なくとも 2 列必要で、必須列の数は 2 を超えることはできません。

Source Class Attribute

メンバシップ テーブルで値が保持されているソース クラスの属性を指定します。

Membership Table Columns

ソースおよびターゲット クラス属性を参照するメンバシップ テーブル列を指定します。

Target Class Attribute

メンバシップ テーブルで値が保持されているターゲット クラスの属性を指定します。

Source Class Attribute

関連するオブジェクトの名前のリストが格納されたソース クラスの仮想属性の名前を指定します。

Target Class Attribute

関連するオブジェクトの名前のリストが格納されたターゲット クラスの仮想属性の名前を指定します。

Objects Must Exist

関連するエントリが存在する必要があるかどうかを指定します。

デフォルト： オン

Use DNs in Attributes

仮想メンバシップ属性に DN のリストが含まれるか、ターゲット オブジェクトの命名属性のリストのみが含まれるかを指定します。

デフォルト： オフ

詳細情報:

間接関連付け (P. 57)

[Mapped Classes]ダイアログ ボックス

［Mapped Classes］ダイアログ ボックスには、マッピングしたクラスすべての読み取り専用リスト、およびそれらにマッピングしたネイティブ クラスが表示されます。

このダイアログには以下のフィールドが含まれます。

Name

マッピングしたクラスを表示します。

Maps to

クラスをマッピングしたネイティブ クラスを表示します。

Add

マッピング ツリーに新しいクラス ノードを追加できます。

Remove

マッピング ツリーからマッピングされたクラスを削除します。

[Map Attributes]ダイアログ ボックス

JNDI の場合、［Map Attributes］ダイアログ ボックスで、プロビジョニング属性をマッピングするネイティブ オブジェクト クラス属性（JNDI）を指定できます。

JDBC の場合、［Map Attributes］ダイアログ ボックスで、プロビジョニング属性をマッピングするテーブル列名を指定できます。

クラス マッピングで定義されていれば、選択した構造クラスおよび補助クラスの属性をすべてマッピングできます。

複数のエンドポイント属性にマッピングされたプロビジョニング属性にはすべて、選択属性が生成されます。 使いやすくするため、これらを対象の属性の近くのユーザ コンソール アカウント画面に配置してください。

Account クラスの場合、属性マッピング テーブルには推奨された一連の共通アカウント属性が事前に入力されています。 これらの属性すべてをマッピングする必要はありません。

このダイアログには以下のフィールドが含まれます。

Name

属性のプロビジョニング表示名を表示します。

問題のクラスの名前属性は太字で表示されます。

Data Type

ネイティブ属性にマッピングしたプロビジョニング属性のデータ型を指定します。

デフォルト： 文字列 デフォルトは、必要に応じて変更できます。

重要: クライアントおよび CA IAM CS でデータ妥当性検証と変換を実行するので、データ モデル タイプは重要です。以下のタイプは、XML スキーマ（XSD）の一部として定義された構文に一致します。 XML スキーマ定義に関する情報については、以下の Web サイトを参照してください。

http://www.w3.org

Binary Data

値が任意のバイナリ データである属性を定義します。

Boolean

XML で true または false を論理的に指定しますが、プロビジョニング サーバおよび JIAM API では LDAP 属性値で 1 または 0 として表されます。

Date

日付を指定します。

例： 1999-05-31

注: プロビジョニング マネージャの動的名前空間プラグインでは、1800 ～ 9999 年までサポートします。ソリューションの他のコンポーネントではこのような制限はないので、実質的にどの年も表すことができます。

Date & Time

特定の曜日の特定の時刻を指定します。

例： 1999-05-31T13:20:00

注: プロビジョニング マネージャの動的名前空間プラグインでは、1970 ～ 2036 年までサポートします。したがって、この範囲外の日を表すには、［日付］を使用してください。

注: ベンダーが混在すると、Connector Xpress によって時間関連の列が処理される方法が複雑になります。 たとえば、MSSQL では「DATETIME」は DateTime 値を示します。しかし他のベンダーでは、標準的な「TIMESTAMP」が使用されています。また、MSSQL の TIMESTAMP は自動的に生成されるバイナリ値を表します。 さらに、Oracle では「TIME」タイプがサポートされず、「DATE」タイプは事実上 TIMESTAMP でもあります。そのため、ベンダー非依存にするため、Connector Xpress では、常に必要に応じて、Date/DateTime/Time のいずれにもマッピングできるようになっています。

Double-precision floating-point

倍精度 64 ビット浮動小数点値を指定します。

Enumeration - enumeration type name

列挙された値の固定リストがある属性を指定します。

Flexi-DN

識別名の文字列形式を

「cn=Bob,ou=Sales,o=ExampleCorp」のように指定します。 これは、コネクタによって適用されます。

Flexi-Email

電子メール アドレスの文字列形式を指定します。

Flexi-Quoteless

引用符を属性値から削除するように指定します。

Floating Point

単精度 32 ビット浮動小数点数を指定します。

Integer

-2147483648 ～ 2147483647 の 32 ビット値を指定します。

Long Integer

9223372036854775808 ～ 9223372036854775807 の 64 ビット値を指定します。

String

無制限のフィールドを指定します。

Time

0 秒 ～ 23:59:59 のオフセットを指定します。

例： 13:20:00

Multi

オンにすると、この属性が複数値であることを示します。

注: ネイティブ属性が複数値の場合、このチェック ボックスは Connector Xpress によって自動的にオンにされます。 ネイティブ属性が単一値の場合、このオプションはオフになり、編集できません。

Maps to

プロビジョニング属性がマッピングされるネイティブ属性名を表示します。

Extended Properties

詳細なメタデータ プロパティを表示します。 これらのフィールドは、［Connector Xpress Preferences (P. 197)］ダイアログ ボックスで［Show Extended Metadata (P. 122)］を選択すると表示されます。

注: 詳細については、「拡張メタデータのプロパティ」を参照してください。

[Map Class]ダイアログ ボックス(JNDI)

［Map Class］ダイアログ ボックスでは、クラスをマッピングするネイティブ オブジェクト クラス（JNDI）を指定できます。 クラス マッピングと補助 LDAP クラスを 0 以上関連付けます。

JNDI ベースのエンドポイント タイプの場合、複数のエンドポイント オブジェクト クラスにアカウント クラスをマッピングできます。CA Identity Manager ユーザ コンソールを使用して、このエンドポイント タイプのアカウントを作成すると、アカウントに使用するオブジェクト クラスがリストから自動的に選択されます。このリストは、アカウント クラスが複数の構造クラスにマッピングされる場合に必ず、Connector Xpress によって自動的に生成される選択属性によって実現されます。 この属性は他の属性と同様にユーザ コンソール アカウント画面で追加します。

このダイアログには以下のフィールドが含まれます。

Name

マッピングするクラスの名前を定義します。

制限： 先頭は英字のみ使用できます。

Description

マッピングするクラスを説明します。

Managed

オフにすると、関連付けを確立する目的でのみこのクラスがマップ済みとしてマークされます。 その結果、Connector Xpress では名前とタイプのみがマップされます。非管理対象クラスのインスタンスはリスト表示して他のオブジェクトと関連付けることができますが、作成、編集、削除できません。

複合クラスの場合は、Connector Xpress によってこのフィールドはデフォルトでオンにされ、オフにできません。

Search Container

このクラスのオブジェクトがすべてある DIT の特定ロケーションを定義します。これは、コネクタ サーバによって実行される一部の検索処理のパフォーマンスを向上します。

Add structural class

ネイティブ LDAP オブジェクト クラスをすべて指定し、属性テーブル内の属性を表示します。

作成するすべてのクラス マッピングには、構造クラスを 1 つ関連付ける必要があります。

Add auxiliary class

選択した既存の補助クラスの構造クラスを指定し、属性テーブル内の他の補助 LDAP クラスから属性を表示します。

Class Name

このプロビジョニング クラス マッピングで選択されたオブジェクト クラスを表示します。

Type

ネイティブ オブジェクト クラスが構造クラスまたは補助クラスのどちらかを表示します。

Derived From

このネイティブ オブジェクト クラスの継承階層を表示します。

Remove

選択されたオブジェクト クラスを削除します。

Extended Properties

詳細なメタデータ プロパティを表示します。 これらのフィールドは、［Connector Xpress Preferences (P. 197)］ダイアログ ボックスで［Show Extended Metadata (P. 122)］を選択すると表示されます。

注: 詳細については、「拡張メタデータのプロパティ」を参照してください。

［Map Class］ダイアログ ボックス（JDBC）

［Map Class］ダイアログ ボックス（JDBC）では、このクラスをマッピングするデータベース テーブルを指定できます。 作成するすべてのクラス マッピングには、データベース テーブルを 1 つ関連付ける必要があります。

このダイアログには以下のフィールドが含まれます。

Name

マッピングするクラスの名前を定義します。

制限： 先頭は英字のみ使用できます。

Description

マッピングするクラスを説明します。

Managed

オフにすると、関連付けを確立する目的でのみこのクラスがマップ済みとしてマークされます。 その結果、Connector Xpress では名前とタイプのみがマップされます。非管理対象クラスのインスタンスはリスト表示して他のオブジェクトと関連付けることができますが、作成、編集、削除できません。

複合クラスの場合は、Connector Xpress によってこのフィールドはデフォルトでオンにされ、オフにできません。

Schema（必須）

［Table］リストで表示されるテーブルのスキーマ名（たとえばテーブルスペースまたはデータベース）を指定します。

Table（必須）

選択されたスキーマでアカウント/グループ情報を表すテーブルを指定します。

作成するすべてのクラス マッピングには、データベース テーブルを 1 つ関連付ける必要があります。

Extended Properties

詳細なメタデータ プロパティを表示します。 これらのフィールドは、［Connector Xpress Preferences (P. 197)］ダイアログ ボックスで［Show Extended Metadata (P. 122)］を選択すると表示されます。

注: 詳細については、「拡張メタデータのプロパティ」を参照してください。

[Map Compound Class and Attributes]ダイアログ ボックス（JDBC）

［Map Compound Class and Attributes］ダイアログ ボックス（JDBC）では、JDBC コネクタにマルチテーブル サポートを提供するのに使用する複合型クラスを作成できます。

作成するすべてのクラス マッピングには、データベース テーブルを 1 つ関連付ける必要があります。

このダイアログには以下のフィールドが含まれます。

Name

マッピングするクラスの名前を定義します。

制限： 先頭は英字のみ使用できます。

Description

マッピングするクラスを説明します。

Managed

オフにすると、関連付けを確立する目的でのみこのクラスがマップ済みとしてマークされます。 その結果、Connector Xpress では名前とタイプのみがマップされます。非管理対象クラスのインスタンスはリスト表示して他のオブジェクトと関連付けることができますが、作成、編集、削除できません。

複合クラスの場合は、Connector Xpress によってこのフィールドはデフォルトでオンにされ、オフにできません。

Schema（**必須**）

［Table］リストで表示されるテーブルのスキーマ名（たとえばテーブルスペースまたはデータベース）を指定します。

Table（**必須**）

選択されたスキーマでアカウント/グループ情報を表すテーブルを指定します。

作成するすべてのクラス マッピングには、データベース テーブルを 1 つ関連付ける必要があります。

Map Columns

選択した列のマッピングの概要を表示します。

注: このテーブルでは、一部のマッピング オプションが表示されません。 頻繁に使用しないオプションを使用するには、クラス ノードを展開して個別の属性詳細ノードを表示します。

Native Name

ネイティブ属性名を表示します。

太字のエントリは、1 つのクラス当たり少なくとも 1 回マッピングする必要がある必須エントリを示します。

問題のクラスの名前属性は太字で表示されます。

Native Type

この用法における、SQL タイプに適したデータモデル タイプを指定します。

Native Size

［Account/Group Table］列のサイズを文字数で表示します。

注: NVARCHARS またはベンダー固有の形式で表された時間関連の値の格納など、場合によっては、この値は「概算」を示します。

Name

ネイティブ属性にマッピングできるプロビジョニング属性のリストを表示します。

Italic entries

属性がすでにマップされたことを示します。JNDI の場合、再度マッピングできます。

Bold entries

1 つのクラス当たり少なくとも 1 回マッピングする必要がある必須エントリを示します。

Custom attributes

属性がそのネイティブ属性名に基づいてデフォルトの名前を割り当てられたことを示します。 ［Provisioning Attribute Details］ダイアログ ボックスでのこの属性を変更できます。

Blank entry

マッピングを削除できます。

注: アカウント クラスの場合、このリストには、一般的な属性のリストも表示されます。

Type

この用法における、SQL タイプに適したデータモデル タイプを指定します。

Binary Data

値が任意のバイナリ データである属性を定義します。

Boolean

XML で true または false を論理的に指定しますが、プロビジョニング サーバおよび JIAM API では LDAP 属性値で 1 または 0 として表されます。

Date

日付を指定します。

例： 1999-05-31

注: プロビジョニング マネージャの動的名前空間プラグインでは、1800 ～ 9999 年までサポートします。ソリューションの他のコンポーネントではこのような制限はないので、実質的にどの年も表すことができます。

Date & Time

特定の曜日の特定の時刻を指定します。

例： 1999-05-31T13:20:00

注: プロビジョニング マネージャの動的名前空間プラグインでは、1970 ～ 2036 年までサポートします。したがって、この範囲外の日を表すには、［日付］を使用してください。

注: ベンダーが混在すると、Connector Xpress によって時間関連の列が処理される方法が複雑になります。 たとえば、MSSQL では「DATETIME」は DateTime 値を示します。しかし他のベンダーでは、標準的な「TIMESTAMP」が使用されています。また、MSSQL の TIMESTAMP は自動的に生成されるバイナリ値を表します。 さらに、Oracle では「TIME」タイプがサポートされず、「DATE」タイプは事実上 TIMESTAMP でもあります。そのため、ベンダー非依存にするため、Connector Xpress では、常に必要に応じて、Date/DateTime/Time のいずれにもマッピングできるようになっています。

Double-precision floating-point

倍精度 64 ビット浮動小数点値を指定します。

Enumeration - enumeration type name

列挙された値の固定リストがある属性を指定します。

Flexi-DN

識別名の文字列形式を

「cn=Bob,ou=Sales,o=ExampleCorp」のように指定します。 これは、コネクタによって適用されます。

Flexi-Email

電子メール アドレスの文字列形式を指定します。

Flexi-Quoteless

引用符を属性値から削除するように指定します。

Floating Point

単精度 32 ビット浮動小数点数を指定します。

Integer

-2147483648 ～ 2147483647 の 32 ビット値を指定します。

Long Integer

9223372036854775808 ～ 9223372036854775807 の 64 ビット値を指定します。

String

無制限のフィールドを指定します。

Time

0 秒 ～ 23:59:59 のオフセットを指定します。

例： 13:20:00

Multi-valued

オンにすると、この属性が複数値であることを示します。

注: ネイティブ属性が複数値の場合、このチェック ボックスは Connector Xpress によって自動的にオンにされます。 ネイティブ属性が単一値の場合、このオプションはオフになり、編集できません。

Extended Properties

詳細なメタデータ プロパティを表示します。 これらのフィールドは、［Connector Xpress Preferences (P. 197)］ダイアログ ボックスで［Show Extended Metadata (P. 122)］を選択すると表示されます。

注: 詳細については、「拡張メタデータのプロパティ」を参照してください。

[Mapped Containers]ダイアログ ボックス

このダイアログ ボックスでは、コネクタでコンテナとして使用するオブジェクト クラスを指定できます。

このダイアログ ボックスには、一目では意味がわからない以下のフィールドがあります。

Name

コンテナの名前を指定します。

Contains

定義されたコンテナをすべてリスト表示します。

Add

マッピング ツリーに新しいコンテナ クラスを追加します。

削除

コンテナ クラスの定義を削除します。

[Map Container Class]ダイアログ ボックス(JNDI)

［Map Class and Attributes Containers］ダイアログ ボックスでは、アカウントとグループのコンテナとして機能できるオブジェクト クラスおよびそれらの命名属性を指定できます。

デフォルトでは、Connector Xpress によって、［Object Class］リストは inetOrg スキーマの inetOrgPerson および groupOfNames のコンテナである以下のエントリが入力されます（アカウントとグループの共通選択肢）。

- Organization – o
- Organizational Unit – ou

注: inetOrgPerson アカウント オブジェクトに対して表示される属性には対応するコンテナと直接の関係がありません。ただし、理想的にはそれらは一致しますが、すべてのアカウント オブジェクトで一致する保証はありません。

このダイアログには以下のフィールドが含まれます。

Name

マッピングするクラスの名前を定義します。

制限： 先頭は英字のみ使用できます。

Description

マッピングするクラスを説明します。

Managed

オフにすると、関連付けを確立する目的でのみこのクラスがマップ済みとしてマークされます。 その結果、Connector Xpress では名前とタイプのみがマップされます。非管理対象クラスのインスタンスはリスト表示して他のオブジェクトと関連付けることができますが、作成、編集、削除できません。

複合クラスの場合は、Connector Xpress によってこのフィールドはデフォルトでオンにされ、オフにできません。

Contained Classes

このコンテナの子にできるクラスを指定します。 たとえば、コンテナ Employee Groups が個別の Account Class ではなく Staff Group および Executive Group クラスのみを許可できることを指定できます。

Add structural class

ネイティブ LDAP オブジェクト クラスをすべて指定し、属性テーブル内の属性を表示します。

作成するすべてのクラス マッピングには、構造クラスを 1 つ関連付ける必要があります。

Add auxiliary class

選択した既存の補助クラスの構造クラスを指定し、属性テーブル内の他の補助 LDAP クラスから属性を表示します。

Class Name

このプロビジョニング クラス マッピングで選択されたオブジェクト クラスを表示します。

Type

ネイティブ オブジェクト クラスが構造クラスまたは補助クラスのどちらかを表示します。

Derived From

このネイティブ オブジェクト クラスの継承階層を表示します。

削除

選択されたオブジェクト クラスを削除します。

Map Object Class Attributes

選択した属性マッピングの概要を表示します。

注: このテーブルでは、一部のマッピング オプションが表示されません。 頻繁に使用しないオプションを使用するには、クラス ノードを展開して個別の属性詳細ノードを表示します。

Native Name

ネイティブ属性名を表示します。

太字のエントリは、1 つのクラス当たり少なくとも 1 回マッピングする必要がある必須エントリを示します。

問題のクラスの名前属性は太字で表示されます。

Native Type

デフォルト： JNDI 用の文字列。 デフォルトは、必要に応じて変更できます。

Name

ネイティブ属性にマッピングできるプロビジョニング属性のリストを表示します。

Italic entries

属性がすでにマップされたことを示します。JNDI の場合、再度マッピングできます。

Bold entries

1 つのクラス当たり少なくとも 1 回マッピングする必要がある必須エントリを示します。

Custom attributes

属性がそのネイティブ属性名に基づいてデフォルトの名前を割り当てられたことを示します。 [Provisioning Attribute Details] ダイアログ ボックスでのこの属性を変更できます。

Blank entry

マッピングを削除できます。

注: アカウント クラスの場合、このリストには、一般的な属性のリストも表示されます。

Type

デフォルト： JNDI 用の文字列。 デフォルトは、必要に応じて変更できます。

重要: クライアントおよび CA IAM CS でデータ妥当性検証と変換を実行するので、データ モデル タイプは重要です。以下のタイプは、XML スキーマ (XSD) の一部として定義された構文に一致します。 XML スキーマ定義に関する情報については、以下の Web サイトを参照してください。

http://www.w3.org

Binary Data

値が任意のバイナリ データである属性を定義します。

Boolean

XML で true または false を論理的に指定しますが、プロビジョニング サーバおよび JIAM API では LDAP 属性値で 1 または 0 として表されます。

Date

日付を指定します。

例： 1999-05-31

注: プロビジョニング マネージャの動的名前空間プラグインでは、1800 ～ 9999 年までサポートします。ソリューションの他のコンポーネントではこのような制限はないので、実質的にどの年も表すことができます。

Date & Time

特定の曜日の特定の時刻を指定します。

例： 1999-05-31T13:20:00

注: プロビジョニング マネージャの動的名前空間プラグインでは、1970 ～ 2036 年までサポートします。したがって、この範囲外の日を表すには、［日付］を使用してください。

注: ベンダーが混在すると、Connector Xpress によって時間関連の列が処理される方法が複雑になります。 たとえば、MSSQL では「DATETIME」は DateTime 値を示します。しかし他のベンダーでは、標準的な「TIMESTAMP」が使用されています。また、MSSQL の TIMESTAMP は自動的に生成されるバイナリ値を表します。 さらに、Oracle では「TIME」タイプがサポートされず、「DATE」タイプは事実上 TIMESTAMP でもあります。そのため、ベンダー非依存にするため、Connector Xpress では、常に必要に応じて、Date/DateTime/Time のいずれにもマッピングできるようになっています。

Double-precision floating-point

倍精度 64 ビット浮動小数点値を指定します。

Enumeration - enumeration type name

列挙された値の固定リストがある属性を指定します。

Flexi-DN

識別名の文字列形式を

「cn=Bob,ou=Sales,o=ExampleCorp」のように指定します。 これは、コネクタによって適用されます。

Flexi-Email

電子メール アドレスの文字列形式を指定します。

Flexi-Quoteless

引用符を属性値から削除するように指定します。

Floating Point

単精度 32 ビット浮動小数点数を指定します。

Integer

-2147483648 ～ 2147483647 の 32 ビット値を指定します。

Long Integer

9223372036854775808 ～ 9223372036854775807 の 64 ビット値を指定します。

String

無制限のフィールドを指定します。

Time

0 秒 ～ 23:59:59 のオフセットを指定します。

例： 13:20:00

Multi-valued

オンにすると、この属性が複数値であることを示します。

注: ネイティブ属性が複数値の場合、このチェック ボックスは Connector Xpress によって自動的にオンにされます。 ネイティブ属性が単一値の場合、このオプションはオフになり、編集できません。

Extended Properties

詳細なメタデータ プロパティを表示します。 これらのフィールドは、［Connector Xpress Preferences (P. 197)］ダイアログ ボックスで［Show Extended Metadata (P. 122)］を選択すると表示されます。

注: 詳細については、「拡張メタデータのプロパティ」を参照してください。

［Merge XML］ダイアログ ボックス

［Merge XML］ダイアログ ボックスでは、メタデータが格納された XML ファイルをプロジェクトにインポートできます。プロジェクトにはそれぞれ自動的にメタデータが格納されますが、別のプロジェクトから再利用したいメタデータがある場合など、外部メタデータ ファイルをインポートする場合があります。XML メタデータ ファイルをインポートすると、システムは新規ファイルをプロジェクトにある既存のファイルとマージします。ただし、競合が発生した場合にどのように処理するか指定する必要があります。このダイアログ ボックスでは、競合を処理する方法を指定できます。

このダイアログには以下のフィールドが含まれます。

Retain existing project values

プロジェクトの既存のメタデータ値を保持します。

Overwrite existing project values with those in the XML file being selected

プロジェクトの既存のメタデータをインポートしたメタデータで置換します。

詳細情報:

変更されたデータ モデルのマージ (P. 109)

Operations Bindings Editor

［Operations Bindings］ダイアログ ボックスには、既存の操作バインディングの概要が表示されます。

このダイアログには以下のフィールドが含まれます。

Available object classes

操作バインディングを作成できる opbindings を表示します。

Filtered object classes

Opbindings 概要リストで表示された操作バインディングをフィルタします。 このリストで表示されたクラスの操作バインディングは Opbindings 概要リストに表示されます。

このリストで選択した内容は、すべての Opbinding Operation エディタ内のすべての Filtered オブジェクト クラス リストで表示されます。また、このリストに対して他の Operation バインディング エディタで行った変更も、このリストで表示されます。

Opbindings summary

既存の操作バインディングの概要を表示します。

Create

操作バインディング情報を指定できる［Create Operation Binding］ダイアログ ボックスを表示します。

Delete

Opbindings 概要リストから操作バインディングを削除します。

詳細情報:

Operation Bindings (P. 69)
ストアド プロシージャ (P. 70)
ストアド プロシージャへのバインド操作 (P. 71)
ストアド プロシージャと列の考慮事項 (P. 76)

Operation Bindings – Stored Procedure Editor

このダイアログ ボックスでは、ストアド プロシージャのスタイル操作バインディング パラメータを指定できます。

このダイアログには以下のフィールドが含まれます。

利用可能なオブジェクト クラス

以下の形式での利用可能なマップ済みオブジェクト クラスを表示します。

- **オブジェクト クラス形式**

  形式： DisplayName [connectorMapToObjectClassName]

  例： User Account [inetOrgPerson]

- **あいまいなオブジェクト クラス形式 -** 以下の形式での単一のアイテムとして表示されます。

  形式： DisplayName [connectorMapToObjectClassName1 | connectorMapToObjectClassName2 | …]

  例： Default Container [organizationalUnit | organization]

注: あいまいなオブジェクト クラスにコンカレント Opbinding を選択すると、Opbinding XML コンテンツには必ず LDAP オブジェクト クラス名が使用されます。その結果、Connector Xpress によって［Use Native Name］チェック ボックスがオフにされ、無効になります。

追加されたオブジェクト クラス

このスクリプト操作バインディングに追加するオブジェクト クラスを指定します。

Execute this procedure

コネクタによってストアド プロシージャが実行されるタイミングを指定します。

Description

操作バインディングの説明です。

Abort operation if procedure fails

エラーの処理方法を指定します。 たとえば、タイミングが［Before］または［After］に設定され、このチェック ボックスがオンになっている場合、例外がスローされ、エラーが発生する場合、操作が中止されます。 このチェック ボックスがオフになっている場合、コネクタによってエラーがログ記録されますが、エラーが発生しても、操作は続行されます。

タイミングが［Instead Of］に設定されている場合にエラーが発生すると、コネクタによって必ず例外がスローされ、操作バインディングを実行している場合は操作が中止されます。

タイミングが［Instead Of］に設定されている場合、このフィールドはデフォルトでは［true］に設定され、タイミングが［After］に設定されている場合、デフォルトではクリアされます。

Use Native Name

バインディングにコネクタによって DYN LDAP 属性名ではなくコネクタ マップ属性名を使用するように指定します。

Lookup Level

opbinding が機能するため、属性または削除されたオブジェクトをキャッシュするかどうかを指定し、オブジェクトの値すべてのルックアップを制御します。 キャッシングは通常、［Post］削除の操作に対する予防措置として実行されます。

スクリプトのデフォルトは［Full］です。メソッドでは、フレームワークによって［Post］削除ストアドプロシージャに対してマップされたパラメータがすべて検証しコンテキスト属性以外が存在しているかどうかによって、［Full］または［Exists］を選択します。

注: ターゲット オブジェクトが削除される前に、属性値がキャッシュされる必要がある場合があるため、LookUpLevel 属性は Post 削除ストアドプロシージャ操作のバインディングに重要です。

Full

属性値をすべてキャッシュします。 これはスクリプトのデフォルトです。

Exists

CA IAM CS によって削除されるオブジェクトが存在することが確認されます。 オブジェクトが存在しない場合、CA IAM CS によって LdapNameNotFoundException がスローされます。

None

メソッドまたはスクリプト ペイロードによってターゲット オブジェクトが存在するかどうかが判別されることを指定します。［Full］または［Exits］を選択すると、フレームワークによってターゲット オブジェクトが存在するかどうか自動的に判別します。 つまり、CA IAM CS がコールを渡し、コールするスクリプトまたはストアドプロシージャによって、オブジェクトが存在するかどうかが判別されます。

Procedure

選択された操作の以前、条件に一致しない場合、または以降に実行するストアドプロシージャを指定します。

In / Out

パラメータのタイプを指定します。

In

パラメータまたはタイプがストアドプロシージャへ値を渡すことを指定します。

InOut

ストアドプロシージャによって値をアウトおよびインの両方に渡すことを指定します。

Out

ストアド プロシージャよって値をアウトに渡すことを指定します。

Parameter

ストアドプロシージャの定義に従い、パラメータの名前を指定します。

Data Type

指定されたパラメータの SQL タイプを表示します。

Attribute

ストアドプロシージャのパラメータにマッピングされる属性を指定します。

注: このリストで先頭と末尾にアスタリスクがある値は、指定された操作に対するランタイム コンテキスト値です。 たとえば、操作対象アカウントの識別名または MODIFY RN 名前変更操作の「new name」などの操作固有値が該当します。 その他の値は指定されたクラスに対して、以前にマッピングされた属性値です。

Name

目標オブジェクトで最もネストされる RDN 値を定義します。

DN

目標オブジェクトの完全な識別名を定義します。

ErrorStatus

ストアドプロシージャから記述エラー文字列を渡す属性をコネクタが使用するように指定します。 バインディングの *strict completion* が true に設定されている場合、この値によってエラーが発生する場合があります。

AddModify_AttrsAsXML

Add または Modify 操作全体が単一の XML 文字列として渡されるように指定します。

注: これは、エンドポイントの組み込み XML が十分にサポートされている場合、ADD または MODIFY 操作にバインドされたストアド プロシージャで有用です。 リクエスト全体は、単一の XML 文字列で渡され、その後、必要に応じてストアドプロシージャ内で分類できます。

ModifyRn_NewRdn

新しい RDN を指定します。

Move_NewParentName

新しい親名を定義します。

MoveRename_NewParentName

新しい親名を定義します。

MoveRename_NewRdn

新しい RDN を指定します。

CLASS

オブジェクト タイプの名前を定義します。

OPERATION

操作名（Add、Delete など）を定義します。

ALIAS

スクリプトのエイリアスを定義します。

注: 複数のオブジェクト クラスを選択した場合、選択したオブジェクト クラスの属性の和集合およびランタイム属性（*NAME*、*DN* など）のみが、このフィールドで利用可能です。

Multivalued

パラメータに複数の値を指定でき、平坦化スタイルを必要とすることを指定します。

Flattening Style

単一の文字列リテラルで複数の値を表すために使用される平坦化メソッドを指定します。 属性が複数値の場合は、［Flattening Style］列を使用します。

Flattening Mode

追加（ADD）されたまたは削除（REMOVE）された値を使用するか、完全に新しい値のリストを渡す（REPLACE）ことで変更を表現する、複数値パラメータの平坦化モードを表示します。

詳細情報:

Operation Bindings (P. 69)
ストアド プロシージャ (P. 70)
ストアド プロシージャへのバインド操作 (P. 71)
ストアド プロシージャと列の考慮事項 (P. 76)

Operation Bindings – Operations Editor

このダイアログ ボックスでは、現在選択された操作の操作バインディングの概要が表示されます。操作はリストで表示された順に実行されます。このダイアログ ボックスを使用して、スクリプトまたはストアド プロシージャを操作にバインドし、操作バインディングが実行される順番を変更できます。

このダイアログ ボックスには、一目では意味がわからない以下のフィールドがあります。

Available object classes

利用可能なマップ済みオブジェクト クラスを表示します。

Filtered object classes

［Before］、［Instead of］、［After］リストで表示される opbindings をフィルタできます。このリストで表示されたクラスの操作バインディングはこれらのリストに表示されます。

このリストで選択した内容は、すべての Opbinding Operation エディタ内のすべての Filtered オブジェクト クラス リストで表示されます。また、このリストに対して他の Operation バインディング エディタで行った変更も、このリストで表示されます。

Before

指定された操作の前にコネクタによって実行される操作バインディングを表示します。 コネクタによって、リストに表示された順に操作バインディングが実行されます。

Instead Of

指定された操作の代わりにコネクタによって実行される操作バインディングを表示します。 コネクタによって、リストに表示された順に操作バインディングが実行されます。

After

指定された操作の後にコネクタによって実行される操作バインディングを表示します。 コネクタによって、リストに表示された順に操作バインディングが実行されます。

Create

操作バインディング情報を指定できる［Create Operation Binding］ダイアログ ボックスを表示します。

Delete

選択されている操作バインディングを削除します。

Move Up

選択された操作バインディングが実行される順序を変更します。

Move Down

選択された操作バインディングが実行される順序を変更します。

[Preferences]ダイアログ ボックス

［Connector Xpress Preferences］ダイアログ ボックスを使用して、Connector Xpress のルック アンド フィール、検索制限、コマンド パラメータなどの、外見および動作のさまざまな要素を変更できます。

このダイアログには以下のフィールドが含まれます。

Search Size Limit

プロビジョニング サーバ ツリーが特定のノードから取得する検索結果の最大数を指定します。多くのポリシーが存在する場合、Connector Xpress によってこの値が使用され、システム リソースが大量にドレインするのが防止されます。

Show Extended Metadata

このデータ モデル アイテムに関連する、追加のメタデータがすべて含まれた、複数のダイアログ ボックスにある拡張セット メタデータを表示します。

Command to launch web browser

コマンド Web ブラウザを起動するコマンドを指定します。

デフォルト： （Windows）"rundll32" "url.dll,FileProtocolHandler"

デフォルト： （Solaris）"/usr/sfw/bin/mozilla"

詳細情報:

環境設定の設定 (P. 25)

[Provisioning Server Details]ダイアログ ボックス

［Provisioning Server Details］ダイアログ ボックスでは、コネクタを展開できるプロビジョニング サーバへの接続を設定できます。

このダイアログには以下のフィールドが含まれます。

Host Name

プロビジョニング サーバが実行される、ホストの名前を定義します。

ユーザ ドメイン

プロビジョニング サーバにログオンする際にコネクタが使用する認証情報のユーザが含まれるドメインを定義します。

User Name

Connector Xpress によってサーバへのログオンに使用されるユーザ名を定義します。

Use TLS

サーバに安全な TLS 接続が確立されることを指定します。 つまり、パスワードはクリア テキスト形式で転送されません。選択しないと、パスワードはクリア テキスト形式で転送されます。

Key

コネクタがプロビジョニング サーバとの接続に使用する設定オプションの名前を定義します。

Value

設定オプションの値を定義します。

[Key]列

コネクタがプロビジョニング サーバとの接続に使用する設定オプションの完全なリストを定義します。

[Value]列

［Key］列で表示された設定オプションに対応する値を表示します。

詳細情報:

プロビジョニング サーバの追加および設定 (P. 96)
プロビジョニング サーバ詳細の編集 (P. 96)

[Provisioning Server Password Required]ダイアログ ボックス

［Provisioning Server Password Required］ダイアログ ボックスでは、選択されたプロビジョニング サーバのパスワードを指定できます。

このダイアログには、以下のフィールドがあります。

Password

選択したプロビジョニング サーバにアクセスするのに必要なパスワードを指定します。

詳細情報:

コネクタの展開 (P. 44)

[Script Editor]ダイアログ ボックス

このダイアログ ボックスでは、スクリプトのスタイル操作バインディング パラメータを指定できます。

このダイアログには以下のフィールドが含まれます。

Available object classes

利用可能なマップ済みオブジェクト クラスを表示します。

Added object classes

このスクリプト操作バインディングに追加するオブジェクト クラスを指定します。

Execute this script

選択された操作の以前、条件に一致しない場合、または以降にコネクタが実行するスクリプトを指定します。

Description

操作バインディングの説明です。

Abort operation if script fails

エラーの処理方法を指定します。 たとえば、タイミングが［Before］または［After］に設定され、このチェック ボックスがオンになっている場合、例外がスローされ、エラーが発生する場合、操作が中止されます。 このチェック ボックスがオフになっている場合、コネクタによってエラーがログ記録されますが、エラーが発生しても、操作は続行されます。

タイミングが［Instead Of］に設定されている場合にエラーが発生すると、コネクタによって必ず例外がスローされ、操作バインディングを実行している場合は操作が中止されます。

Lookup Level

opbinding が機能するため、属性または削除されたオブジェクトをキャッシュするかどうかを指定し、オブジェクトの値すべてのルックアップを制御します。 キャッシングは通常、［Post］削除の操作に対する予防措置として実行されます。

スクリプトのデフォルトは［Full］です。メソッドでは、フレームワークによって［Post］削除ストアドプロシージャに対してマップされたパラメータがすべて検証しコンテキスト属性以外が存在しているかどうかによって、［Full］または［Exists］を選択します。

注: ターゲット オブジェクトが削除される前に、属性値がキャッシュされる必要がある場合があるため、LookUpLevel 属性は Post 削除ストアドプロシージャ操作のバインディングに重要です。

Full

属性値をすべてキャッシュします。 これはスクリプトのデフォルトです。

Exists

CA IAM CS によって削除されるオブジェクトが存在することが確認されます。 オブジェクトが存在しない場合、CA IAM CS によって LdapNameNotFoundException がスローされます。

None

メソッドまたはスクリプト ペイロードによってターゲット オブジェクトが存在するかどうかが判別されることを指定します。［Full］または［Exits］を選択すると、フレームワークによってターゲット オブジェクトが存在するかどうか自動的に判別します。 つまり、CA IAM CS がコールを渡し、コールするスクリプトまたはストアドプロシージャによって、オブジェクトが存在するかどうかが判別されます。

Execute Directly

ScriptStyleOpProcessor が後ほど実行するテキストを生成するのではなく、コネクタが直接バインドする操作を実行することを指定します。

Search Asynchronously

オンにすると、ペイロードの実行が非同期で発生し、1 レベルおよびサブツリーの検索で NamingEnumeration を使用して結果をストリーミング形式で返されるように指定されます。

注: ［Create Operation］ダイアログ ボックスの［Available Operations］ダイアログ ボックスから［Search］操作を選択すると、Connector Xpress にこのフィールドが表示されます。

Execute a function in a global script

グローバル スクリプトの関数がこの操作バインディングのスクリプトとして実行されることを指定します。

Global Script

関数が選択されるグローバル スクリプトを指定します。

New

新しいグローバル スクリプトを作成する［Edit Script］ダイアログ ボックスを表示します。

Function Name

選択されたグローバル スクリプトで実行される関数を定義します。

Execute an individual script

この操作バインディングに対して実行される個別のスクリプトを指定します。

Edit Script

個別のスクリプトを追加または変更する［Edit Script］ダイアログ ボックスを表示します。

[Script Name]ダイアログ ボックス

このダイアログ ボックスでは、グローバル スクリプトのパラメータを変更できます。

このダイアログには以下のフィールドが含まれます。

Script Language

CA IAM CS フレームワークで現在サポートされている利用可能なスクリプト言語を指定します。

Execute Directly

ScriptStyleOpProcessor が後ほど実行するテキストを生成するのではなく、コネクタが直接バインドする操作を実行することを指定します。

詳細情報:

Operation Bindings (P. 69)
スクリプトへのバインド操作 (P. 73)
スクリプト (P. 70)

[Script]ダイアログ ボックス

このダイアログ ボックスでは、グローバル スクリプトのパラメータを指定できます。

このダイアログには以下のフィールドが含まれます。

Add New Global Script

スクリプトを作成し、［Scripts］ノードの下に新しいグローバル スクリプト ノードを追加します。

Delete Global Script

選択されたグローバル スクリプトを削除し、［Scripts］ノードの下からスクリプトを削除します。

[Select Data Source for new project]ダイアログ ボックス

［Select Data Source for new project］ダイアログ ボックスでは、新しいプロジェクトで以前に設定されたデータ ソースを選択する、既存のデータ ソースを追加または編集する、またはデータ ソースを削除することができます。

このダイアログには以下のフィールドが含まれます。

Name

プロジェクトの作成時に使用できる設定されたデータ ソースの名前を表示します。

Type

プロジェクトの作成時に使用できる設定されたソースのタイプを表示します。

Add

利用可能なソース タイプから選択し、新しいデータ ソースを設定できる［Source Types］ダイアログ ボックスを表示します。

編集

指定されたデータ ソースの接続詳細を編集できる［Edit Source］ダイアログ ボックスを表示します。

Remove

リスト ボックスからデータ ソースを削除します。

詳細情報:

プロジェクトの作成 (P. 34)
データ ソースの設定 (P. 27)

[Select Template]ダイアログ ボックス

このダイアログ ボックスでは、共通のエンドポイント スキーマのマップの基礎として使用できるテンプレートを選択できます。 RFC などの規格が広く使用されるスキーマを定義して、完全なカスタム スキーマが一般的ではない LDAP のプロジェクトでテンプレートから開始すると便利です。テンプレートには、アカウント管理画面を描画するのに CA Identity Manager が使用する専用の JavaScript マークアップが含まれています。

各テンプレートをクリックすると、簡単な説明が表示されます。

Open in Wizard Mode

ウィザードを使用してプロジェクトを開始して、選択したテンプレートに応じてアカウント クラスのみ、またはアカウント クラスおよびグループ クラスをマッピングする基本的な手順をステップ バイ ステップでサポートします。

詳細情報:

テンプレート (P. 31)

[Source Types]ダイアログ ボックス

［Source Types］ダイアログ ボックスでは、データ ソースで利用可能なソース タイプを選択できます。

このダイアログには以下のフィールドが含まれます。

None

属性を作成し、データ ソースを指定せずに、展開を目的としない汎用メタデータを編集します。

Available Source Types

コネクタ定義の基礎として使用できるソース タイプを表示します。

JDBC

データベース テーブルとストアド プロシージャにマッピングします。

JNDI

ディレクトリ サーバに格納されているオブジェクト クラスと属性にマッピングします。

詳細情報:

プロジェクトの作成 (P. 34)
データ ソースの設定 (P. 27)

[Wizard Summary]ダイアログ ボックス

このダイアログ ボックスでは、作成したマッピングの概要が表示されます。

詳細情報:

ウィザードを使用したプロジェクトの作成 (P. 37)