CHAPTER 8

外部ディレクトリとの統合

概要

サービス カタログ（Service Catalog）ディレクトリ統合は、一元化されたユーザ認証およびエンタープライズ ディレクトリとの同期を実装することによって、セキュリティ管理を簡略化し、ユーザの利便性と生産性を向上します。

サービス カタログ（Service Catalog）では、カスタマーは（通常は LDAP プロトコルを使用して）外部ディレクトリと統合し、ユーザ情報を同期することができます。この同期は、ユーザが Order-on-behalf（OOB）で選択されたとき、または Person Lookup 時に起動されます。

シングル サインオン（SSO）の統合により、一元管理されたユーザ認証が可能になり、個別のログイン メカニズムが不要になります。SSO イベントを有効にすると、サービス カタログ（Service Catalog）が統合されているエンタープライズ ポータルにすでにログインしているユーザは、ログインし直す必要がありません。サービス カタログ（Service Catalog）は SSO ツールを利用してすべての Service Catalog URL を保護し、認証を実行します。サービス カタログ（Service Catalog）で認証を正常に行うには、SSO ツールにより個人の識別情報を HTTP ヘッダーまたは cgi ヘッダーを介して、毎回の認証でサービス カタログ（Service Catalog） URL へ提供する必要があります。認証されると、それらの情報をアプリケーション データベースと同期することができます。

SSO が有効でない場合は、サービス カタログ（Service Catalog）のログイン画面がすべてのユーザに表示され、ユーザは正しいユーザ名とパスワードの組み合わせを入力することができます。デフォルトでは、これらのクレデンシャルは内部データベースで認証されます。または、外部システム（通常は LDAP ディレクトリ）で認証するよう、Directory Integration を設定することもできます。サービス カタログ（Service Catalog）にアクセスするすべてのユーザは、認証が正常に行われるように、このソース内に存在している必要があります。

Directory Integration Framework は、頻繁に導入される SSO およびディレクトリ サーバ製品に対して、Administration モジュールで使用できる設定オプションを通じて上記の機能を提供します。フレームワークには、定義済みの設定機能を補完可能なアプリケーション プログラミング インターフェイス（API）も含まれています。プログラマは API を使用して追加の SSO ポータルやディレクトリ サーバにアクセスできるだけでなく、サービス カタログ（Service Catalog）と外部ディレクトリの間でユーザ情報が同期されるよう、デフォルトの動作を変更または補完できます。

この章では、Administration モジュールを使用してサービス カタログ（Service Catalog）に対してディレクトリ統合を設定する方法について説明します。また、有効な統合オプションのカスタマイズで使用できる公開 API とインターフェイスのセット、カスタム コードをコンパイルおよび導入するためのベスト プラクティス、Administration モジュールを使用してカスタム コードを設定するための手順についても説明します。

前提条件

ディレクトリ統合の設定では、次のことが必要です。

- 動作するサービス カタログ（Service Catalog）のインストール。
- ディレクトリ サーバがインストールされ、ディレクトリに会社データが入力されていること。潜在的なすべてのユーザに対するディレクトリ エントリでは、マッピングの定義で説明されているように、統合処理で必要なフィールドにマップされるすべての属性についてヌル以外の値が含まれている必要があります。
- 認証を行い、サービス カタログ（Service Catalog）へのアクセスを許可する SSO システム（シングル サインオン（SSO）が使用される場合）。
- ユーザは、「グローバル設定の管理」機能が含まれたロールでログインします。この機能は、「Site Administrator」ロールに自動的に含まれ、「admin」ユーザに割り当てられます。ただし、必要に応じて、Administration モジュールの Roles オプションを使用して他のロールまたはユーザに割り当てることもできます。

（注）　LDAP ブラウザにアクセスすることを強く推奨します。

ディレクトリ統合を設定するための前提条件

ディレクトリ統合を設定するには、SSO の現在の実装（使用している場合）および社内のディレクトリ サーバに関する有用な情報を入手し、これらのシステムをサービス カタログ（Service Catalog）に統合するための要件を文書化する必要があります。ここでは、この情報を収集するためのワークシートをいくつか示します。

これらのワークシートは、ディレクトリ/SSO 統合を設定したり、統合を実装する前に解決すべき問題を特定したりするのに必要な情報を収集するうえで役に立ちます。また、ディレクトリ統合で必要な開発およびテストの時間を見積もる場合にも有用です。

データソースの定義

サービス カタログ（Service Catalog）は、アクセスされる個人データおよび組織データを格納する各ディレクトリに対して「データソース」を定義します。データソースの定義には、外部ディレクトリへの接続で必要なすべての情報、およびそのディレクトリからの抽出情報が含まれています。

各外部ディレクトリに対して 1 つのデータソースを定義する必要があります。たとえば、開発と実稼働で別のディレクトリを使用することができます。また、サービス カタログ（Service Catalog）は LDAP ディレクトリ照会をサポートしており、データソースは参照チェーン内の各ディレクトリに対して定義する必要があります。

表 8-1　　データソース定義テーブル

設定	値	説明
Datasource Name		データソースの名前。空白または特殊文字は使用しないでください。
Datasource Description		データソースの説明（オプション）。

表 8-1　データソース定義テーブル（続き）

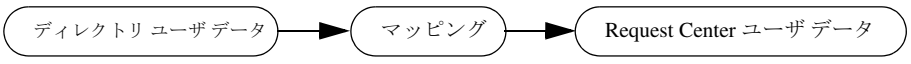
設定	値	説明
Protocol	• LDAP	現時点でサポートされているプロトコルは LDAP だけです。他のプロトコルを使用してディレクトリ情報を格納する場合は、この情報にアクセスするためのカスタム コードを作成する必要があります。
Server Product	• Sun™ ONE Directory • Microsoft® Active Directory® • IBM® Tivoli®	使用するディレクトリ サーバ製品を選択します。現時点でサーバがサポートされていない場合は、サーバにアクセスするためのカスタム コードを作成し、ディレクトリ情報を抽出する必要があります。
Authentication Method	• [シンプル（Simple）] • Anonymous • SASL	**Simple** はプレーン テキストのユーザとパスワードを表します。**SASL**（Simple Authentication and Security Layer）も使用できますが、SASL は Sun ONE Directory Server でのみ機能します。
Connection Mechanism	• SSL • 非 SSL	認証方法として **Simple** または **SASL** を選択した場合のみ必要です。 暗号化された情報を送信するには、**SSL** を選択します。
BindDN		バインドの識別名フィールド。BindDN は、サービス カタログ（Service Catalog）がディレクトリ操作を実行するときに LDAP サーバに接続するために使用されます。 そのためにサービス アカウントを作成します。 このデータソースを外部認証の手順で使用する場合は、[Options] 領域に EUA Bind DN を指定してこの値を変更します。詳細については、外部ユーザ認証（EUA）操作を参照してください。
[パスワード（Password）]		認証で **Simple** または **SASL** を選択した場合に必要です。Bind DN に指定されたユーザのパスワードです。アカウントでパスワード エージングを使用する場合は、このパスワードを定期的に更新する必要があります。
ホスト		LDAP ディレクトリ サーバの完全修飾ドメイン名または IP アドレス。
部品番号		ディレクトリ サーバに接続するためのポート番号。通常、SSL 以外のアクセスにはポート番号 389 を使用します。
ユーザ BaseDN		ディレクトリ内の個人の検索を開始するディレクトリ。社内ディレクトリには多くのブランチが含まれることがあるため、ユーザ データに対してベース DN を指定することによって、ディレクトリ検索が最適化されます。
AuthzID		**SASL** 認証を選択した場合に必要です。

表 8-1　データソース定義テーブル(続き)

設定	値	説明
Optional Filter		このフィルタは、使用する他の検索フィルタに追加されます。このフィルタを使用して、検索結果を効果的に変更できます。フィルタ式はカッコで囲む必要があります。たとえば、次のフィルタがあるとします。 (&(!(msExchHide=true)(ISC-GID=*))) この場合、msExchHide 属性が true で、ISC-GID 属性が定義されているエントリだけが返されます。
Security Certificate Name		接続メカニズムに SSL を選択した場合に必要です。 証明書のエイリアス名には、スペースや特殊文字を使用しないでください。 証明書のデータを入力する準備ができていることを確認してください。
Referral Datasource		1 つ以上のデータソースを照会用として追加できます。データソース検索で結果が返ってこない場合、システムは照会用のデータソースも検索します。照会は、検索でのみサポートされており、バインディングではサポートされていません。 *循環照会は設定できません。循環照会は、あるデータソースが照会用として別のデータソースを持っており、同時にそのデータソースが、照会用として元のデータソースを使用しているものです。たとえば、データソース A が照会用としてデータソース B を持っており、データソース B が照会用としてデータソース A を持っている場合です。*

マッピングの定義

「マッピング」は、外部ディレクトリからサービス カタログ(Service Catalog)へデータをどのように転送するか、という指示を与えるルール セットです。これにより、ディレクトリ内のソース属性と、サービス カタログ(Service Catalog)データベース内のターゲット フィールド間がマッピングされます。サービス カタログ(Service Catalog)データベースがディレクトリと同期されるとき、これらのルールを使用して、ディレクトリのデータを、指定されたターゲット フィールドへ転送します。



同じマッピングを複数のディレクトリ(データソース)に適用できます。

マッピングには、ユーザ/個人のプロファイルと、関連するすべてのエンティティ(住所、連絡先、場所、1 つ以上のグループ関係、1 つ以上の組織単位(OU)関係、1 つ以上の RBAC(ロールベース アクセス コントロール)ロール関係など)が含まれます。

個人のプロファイルには、7 個の必須フィールドが含まれており、これらのフィールドは以下のマッピング ワークシートの「必須」セクションにリストされています。これらのどのフィールドについても値を提供していないディレクトリ レコードはインポートできません。その他のフィールドで、個人プロファイルの一部になっているものもマップできます。詳細については、『Cisco Prime Service Catalog Administration and Operations Guide』の「Organization Designer」の章の「People」の項を参照してください。

個人プロファイルの大半のフィールドは、Service Catalog の機能を動作させるために使用されます。マッピングでは、マップされる属性が、フィールドに対して適切なソース値を提供していることを確認する必要があります。つまりこれらのフィールドに対して、フィールド名によって示されるよりも多くの情報、またはフィールド名に一致しない情報を指定しないようにします。

サービス カタログ（Service Catalog）には、標準の個人データを拡張するフィールドも含まれています。これらのフィールドは、以下の表では「拡張」として記載されており、Organization Designer の Person 情報の [Extensions] ページに表示されます。最も頻繁に必要になる拡張フィールドの一部には意味のある名前（たとえば、会社コードや部門）が割り当てられていますが、他のフィールドの名前は Custom 1 から Custom 10 であり、事前に考えられた意味を使用せずに、自由に使用できます。LDAP ディレクトリに追加の個人情報があり、Service Catalog で公開する必要がある場合は、この情報が含まれている属性を、個人の拡張フィールドの 1 つにマップします。

以下のワークシートの「Directory Attribute」カラムは、個人プロファイルのすべてのフィールドについて値を挿入する必要があります。このフィールドには、ディレクトリがデータを提供します。属性には次のいずれかを指定できます。

- ディレクトリの属性名。複数の属性を（オプションのリテラルを付けて）連結し、フィールドの値を構成する場合は複数の属性名。
- 「カスタム マッピング」、およびそれに続く番号または説明。すべてのカスタム マッピングは、カスタム マッピングに詳細を記述するか、「Comments」カラムに簡単に説明を記載する必要があります。カスタム マッピングは、正規表現の結果を属性に割り当てることも、カスタム Java コードのモジュールを介して実装することもできます。これらのマッピングの実装の詳細は、マッピングの設定に示してあります。

必須マッピング

表 8-2　必須マッピング フィールド説明テーブル

フィールド	説明
名	
姓	
ログイン ID	サービス カタログ（Service Catalog）で個人のログイン名として使用される固有識別子。
Person Identification	Person Identification は、各個人に対して一意の値を提供する属性にマップする必要があります。たとえば、従業員 ID や社会保障番号が格納されている属性を指定します。理想的には、Login ID と Person Identification の両方に同じ属性をマップする必要があります。最低限、これらの 2 つのフィールドは密接に関連付ける必要があります。
電子メール アドレス（Email Address）	

表 8-2　必須マッピング フィールド説明テーブル(続き)

Home Organizational Unit	ホーム OU は必ず事業部門とします(サービス チームにはしません)。
[パスワード(Password)]	ディレクトリ サーバは通常、パスワードを返しません。ただしこのフィールドを使用して、新しいユーザのデフォルト パスワードなどを作成できます。

オプションのマッピング

表 8-3　オプションのマッピング フィールド説明テーブル

フィールド	説明
役職(Title)	
SSN(社会保障番号)	
Birthdate	マップされる LDAP 属性の戻り値の型は、long にする必要があります。サービス カタログ(Service Catalog)では他の形式はサポートされていません。
雇用日(Hire Date)	マップされる LDAP 属性の戻り値の型は、long にする必要があります。サービス カタログ(Service Catalog)では他の形式はサポートされていません。
タイムゾーン ID(TimeZone ID)	マップされる属性は、次の形式のいずれかの値を返す必要があります。 • GMT+- オフセット • Country/Language 2008 年 3 月以降、一般的に使用されていた 3 文字のタイム ゾーン指定(東部標準時は「EST」など)は使用されなくなりました。上記の形式についてサポートされている値のリストは、サポートされるタイム ゾーンを参照してください。戻り値が、正しい形式のどれにも一致しない場合、サービス カタログ(Service Catalog)はデフォルトのタイム ゾーンとして PST を使用します。
ロケール ID(Locale ID)	マップされた属性は、次の形式で値を返す必要があります。 language_COUNTRY language は 2 文字の言語コードで、country は 2 文字の国コードです。 Directory integration は次のロケールをサポートしています。 • en_US(米国英語) • de_DE(ドイツ語) • es_ES(スペイン語) • fr_FR(フランス語) • ja_JP(日本語) • zh_CN(簡体字中国語) • zh_TW(繁体字中国語) • Korean
従業員コード(Employee Code)	

表 8-3　オプションのマッピング フィールド説明テーブル（続き）

フィールド	説明
スーパーバイザ（Supervisor）	このフィールドはマネージャの ID を表します。詳細については、[マネージャのインポート（Import Manager）] 操作を参照してください。
注記（Notes）	
Company Street 1	
Company Street 2	
会社の所在都市	
Company State	
郵便番号	
Company Country	
建物	
水準器	
オフィス（Office）	
パーティション	
Personal Street 1	
Personal Street 2	
Personal City	
Personal State	
Personal Postal Code	
Personal Country	
職場の電話	
自宅の電話	
ファクス（Fax）	
携帯電話（Mobile Phone）	
ポケットベル（Pager）	
その他	
Main Phone	
プライマリ Phone（Primary Phone）	
Primary Fax	
Sales Phone	
Support Phone	
Billing Phone	

表 8-3　オプションのマッピング フィールド説明テーブル（続き）

フィールド	説明
その他連絡先情報	
会社コード	拡張
部門（Division）	拡張
部門	拡張
部署番号	拡張
Cost Center	拡張
Management Level	数値を返す必要があります。このフィールドが Import Manager イベントで使用された場合、階層に従って Management Level が大きくなるようにしてください。たとえば、Junior Engineer と Senior Engineer の 2 つの指定があった場合、Junior Engineer に対して返される Management Level は、Senior Engineer の Management Level よりも小さくなる必要があります。
地域	拡張
Employee Type	拡張
Location Code	拡張
カスタム 1	拡張
カスタム 2	拡張
カスタム 3	拡張
カスタム 4	拡張
カスタム 5	拡張
カスタム 6	拡張
カスタム 7	拡張
カスタム 8	拡張
カスタム 9	拡張
カスタム 10	拡張
Organizational Unit List	このマッピングを使用して、個人を 1 つ以上の組織単位に関連付けます。マッピングでは、複数の値が返されることがあります。このフィールドの場合、サービス カタログ（Service Catalog）は複数値の LDAP 属性によって返されたすべての値を使用します。このフィールドの入力は、次の形式のいずれかにする必要があります。 • Directory Integration API ドキュメントの定義に従って複数値を返す Java クラスの名前。 • 「::」で区切った 1 つ以上の簡単なマッピング。 例：ou::departmentNumber • 次のように、「::」で区切った 1 つ以上の式のマッピング expr:#memberOf#=(cn=(.*),cn=Users,dc=celosis,dc=com)?($1):Default:: expr:#memberOf#=(cn=(.*),ou=Users,dc=celosis,dc=com)?($1):Default

表 8-3　オプションのマッピング フィールド説明テーブル（続き）

フィールド	説明
Group List	Organizational Unit List と同様です。
Role List	Organizational Unit List と同様です。返されるロールは、システム定義またはユーザ定義のいずれかです。 システム定義ロールの場合、名前は言語が米国英語のときにブラウザに表示されるものと同一にする必要があります。他の言語はサポートされていません。たとえば、ユーザに「My Services Executive」ロールを関連付けるには、このロールが返される必要があります。 ユーザ定義ロールの場合、この名前は、ロールの作成時にユーザ言語で入力した名前と完全に一致している必要があります。

カスタム マッピング

以下のワークシートを使用して、カスタム マッピングの要件を文書化できます。

表 8-4　カスタムのマッピング フィールド説明テーブル

フィールド	タイプ（Type）	要件
	式 Java	
	式 Java	

統合イベント、操作、および手順の定義

統合イベントはサービス カタログ（Service Catalog）と外部ディレクトリや SSO プログラムとの間のインターフェイスです。外部プログラムまたはディレクトリにアクセスするサービス カタログ（Service Catalog）を使用しているときだけのものです。これらのイベントは、順次実行される複数の操作によって構成されています。

イベント

サービス カタログ（Service Catalog）は、4 つのディレクトリ統合イベントをサポートしています。

- 「ログイン（Login）」イベントは、ユーザのクレデンシャルが検証され、ユーザがサービス カタログ（Service Catalog）に接続したときに発生します。このイベントは、ユーザが最初にサービス カタログ（Service Catalog）セッションを開始したときに発生します。また、セッションがタイムアウトし（管理者が指定したタイムアウト期間が経過）、再接続が必要な場合にも発生します。
- 「**Person Lookup**」イベントは、ユーザ情報が取得されるたびに発生します。実際には、次の 3 つのタイプの Person Lookup イベントがあります。

  - 代理オーダーの個人検索（Person Lookup for Order on Behalf）：あるユーザが他の個人の代わりにサービスを要求し、そのサービスのカスタマーとなっている個人を選択する必要があります。
  - **Person Lookup for Service Form**：サービス フォームに [Person] フィールドが含まれ、これによってユーザが他の個人をサービス データの一部として指定できます。
  - **Person Lookup for Authorization Delegate**：サービスの確認または承認を行うユーザが、他の個人を一時的な代理承認者として指定するために、自身のプロファイルの変更を要求します。

操作

さまざまな種類の操作を実行するよう、イベントを設定できます。操作はそれぞれのイベントごとに一連の手順で指定され、これにより、呼び出される各操作の順序が決まります。

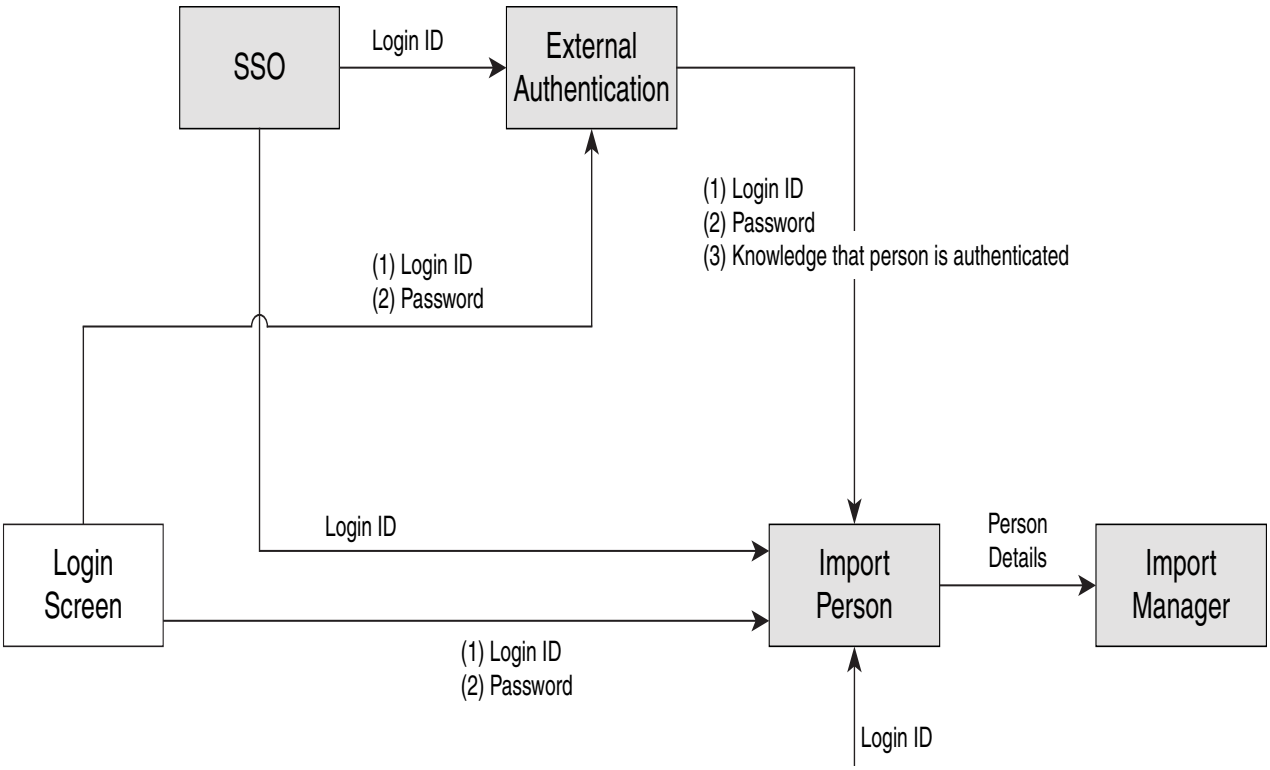
ディレクトリ フレームワークには次の操作が含まれます。

- **シングル サインオン**（SSO）は常に、「ログイン（Login）」イベントの最初の手順です。SSO 操作は、ユーザのログイン名を識別します。
- **External Authentication** は SSO 操作の後に発生するか、または SSO を使用しない場合は、デフォルトの [ログイン（Login）] 画面の後に発生します。外部認証はユーザのログイン名とパスワードを使用して、外部データ ソースに対してユーザを認証します。
- **Person Search** は、ユーザがデータ ソース上で検索を呼び出したときにトリガーされます。Person Search はユーザの First Name と Last Name を使用して、一致した項目のリストを出力します。
- **Import Person** は、外部認証の後、または SSO の後に発生するか、[個人検索（Person Search）] ダイアログ ボックスで個人が選択された後で発生します。Import Person は、検索されている個人、またはログインしている個人のログイン名を使用し、データ ソースに問い合わせて、個人をデータベースにインポートします。
- **Import Manager** は、個人のインポート後のみで発生します。[マネージャのインポート（Import Manager）] 操作は、インポートされた個人情報を使用して、このユーザのマネージャをインポートします。

各操作は、カスタム コード インターフェイスの実装によってカスタマイズできます。

次のトリガー順序は、操作がトリガーされる順序を示しています。

図 8-1　　トリガー順序



ログイン イベント

ディレクトリ統合のログイン イベントが設定されていない場合は、デフォルトでログイン画面が表示され、アプリケーション データベースに対して入力されたクレデンシャル(ユーザ名とパスワード)が検証されます。

ディレクトリ統合イベントが有効になっている場合は、最初の手順として、次のいずれかの操作を使用して Login イベントを設定できます。

- シングル サインオン:すべてのユーザが SSO ベンダーを使用して事前に認証されている社内環境では、要求ヘッダーまたは CGI ヘッダーからユーザのログイン ID を自動的に抽出し、アプリケーションのログイン画面を表示せずに透過なログインを可能にします。
- 外部ユーザ認証:アプリケーションのログイン画面を表示し、指定した外部ディレクトリに対して入力されたクレデンシャルを検証します。外部ユーザ認証は、SSO 操作の後にも発生することがあります。
- 混在モード認証:アプリケーション サーバ ポート経由の NTLM 認証を回避します。DB クレデンシャル(ユーザ名とパスワードの両方)がアプリケーション サーバ ポート 経由で渡され、Web サーバで NTLM 認証が有効になっている場合は、データベース認証がトリガーされ、データベース ユーザがセッションを所有します。

  EUA が有効で、ユーザ名とパスワードの両方が渡されると、LDAP 認証がトリガーされます。提供されるパスワードは LDAP パスワードである必要があり、LDAP ユーザはセッションを所有します。

ユーザのクレデンシャルが検証されると、「ログイン(Login)」イベントには、外部データベースとサービス カタログ(Service Catalog)間でユーザ データを同期するために、次の操作が含まれることがあります。

- 「Import Person」操作がその次の手順です。この操作は、選択された認証済みの個人プロファイルをサービス カタログ（Service Catalog）にインポートし、データと同期します。
- 「Import Manager」の操作は、「Import Person」手順の次の操作です。この操作は選択した個人のマネージャに関する情報を外部ディレクトリから検出して、その情報を持つサービス カタログ（Service Catalog）データベースを更新します。

Single Sign-On 操作

シングル サインオン（SSO）ソリューションとの統合では、次の 2 つのメカニズムやプロトコルのいずれかを使用できます。

1. Active Directory Services（ADS）/NT LAN Manager（NTLM）ベースの認証済みユーザ
   - サービス カタログ（Service Catalog）へのログインに、サードパーティの IM/AM/SSO 製品は必要ありません。
   - ログインしたユーザの、POSIX 準拠 OS のクレデンシャルが、ブラウザによって サービス カタログ（Service Catalog）に返されます。
   - これは、SSO の CGI ヘッダーによる統合とも呼ばれます。
2. HTTP 要求ヘッダー
   - これは非 ADS/NTLM 統合です。
   - http プロトコルで要求ヘッダーを使用してサービス カタログ（Service Catalog）にログインするために、サードパーティの IM/AM/SSO 製品が必要です。

Portlet と Directory Integration の両方で SSO の使用を計画しているカスタマーについては、HTTP ヘッダー SSO のみがサポートされます。Directory Integration フレームワークのカスタム SSO プラグインはサポートされていません。

設定	値	説明
Single Sign-On Type	HTTP Header リモート ユーザ（Remote User）	使用する SSO ソリューションに応じてタイプを指定します。ログイン ID 情報にアクセスできることを確認してください。 http 要求ヘッダー プロトコルを使用する場合は HTTP Header をチェックします。 ADS/NTLM プロトコルを使用する場合は Remote User をチェックします。
Login ID Mapping		HTTP サインオンに対する Login ID マッピングは、サインインしているユーザのログイン名が含まれている Http 要求ヘッダー内の名前と同じ名前にする必要があります。 ADS/NTLM サインオンに対する Login ID マッピングは、次の形式にする必要があります。 #AnyDomain#\#LoginId# たとえば celosis\#LoginId# は、ユーザが「celosis」ドメインに制限されますが、#AnyDomain#\#LoginId# は複数のドメインでログインができます。 複数のドメインを使用している場合、LoginId はドメイン間で一意にする必要があります。
リダイレクト URL		通常、ユーザがサービス カタログ（Service Catalog）製品にアクセスする社内ポータルの URL。認証に失敗した場合、またはアプリケーション ユーザ セッションがタイム アウトになった場合に、ユーザはこの URL にリダイレクトされます。

SSO の管理バイパス

一部のユーザにシングル サインオンのバイパスを許可して、サービス カタログ（Service Catalog）に直接ログインできるようにすることが必要な場合があります。この機能は通常、次のユーザにとって必要です。

- シングル サインオンの問題を調査する必要があるシステム管理者
- サービス設計およびタスク計画を検証するために、複数ユーザのパフォーマンスをエミュレートする必要があるテスト担当者

サービス カタログ（Service Catalog）には、ユーザがログイン画面にアクセスし、ユーザ名とパスワードを入力できるようなメカニズムが用意されています。newscale.properties ファイル（RequestCenter.war にあります）では「BackDoorURLParam」の値が指定されます。たとえば、次のようになります。

```
BackDoorURLParam=AdminAccess
```

ログイン画面からサービス カタログ（Service Catalog）にアクセスするために使用する URL には、パラメータを含める必要があります。たとえば backDoorURLParam の上記の値について、サンプル URL は次のようになります。

```
http://prod.RequestCenter.com:<app_server_port>/RequestCenter?AdminAccess=true
```

<app_server_port> はアプリケーション サーバのポート番号です。

会社のガイドラインに従って BackDoorURLParam の値の期限切れについてポリシーを設定したり、サービス カタログ（Service Catalog）への管理アクセスの制御についてポリシーを設定したりすることは、管理者が行います。管理 URL でアクセスできるのは、対応する管理者設定によって「Site Administrator」ロールを付与されたユーザだけです。

図 8-2　管理設定

On	Off	Setting	Description
Common			
○	◉	Enable Custom Header Footer	Site will add content from the custom header and footer HTML. Default is off.
○	◉	Enable Custom Style Sheets	Site will utilize the custom stylesheet allowing for the changing of logos, color schemes, fonts and others. Default is off.
○	◉	Directory Integration	Enable the Directories feature that searches for and imports users into the site from an external datasource (e.g. LDAP). Default is off.
○	◉	Restrict Site Administrator URL	Allow only those users with the Site Administrator Role to log in using the administrator URL (i.e., bypassing Single Sign-On). Default is off.

管理者は、ユーザが URL に直接アクセス可能なことも確認する必要があります。以前は、サービス カタログ（Service Catalog）アプリケーションへのアクセスが、Web サーバまたはネットワーク設定パラメータを介した SSO ソフトウェアによるアクセスに制限されていました。

Service Catalog サービスを再起動して、このパラメータに対する変更を有効にする必要があります。

外部ユーザ認証（EUA）操作

外部認証を使用して、すべてのサービス カタログ（Service Catalog）ユーザを社内ディレクトリで認証します。このようにすれば、ユーザ パスワードの同期を心配する必要もありません。

外部ユーザ認証は、ログインの試行後に、設定済みのシングル サインオン操作またはアプリケーションのログイン画面から行う必要があります。前の操作で取得した LoginId は EUA 操作で使用できます。ただし、このユーザを外部ディレクトリで検証する場合には、BindDN を見つけられるように、追加情報が必要になります。

EUABindDN 設定により、アプリケーションで、サインオンしようとしているユーザのバインド DN を自動的に推定できます。

表 8-5　EUABindDN 設定

設定	説明

External Authentication EUABindDN	EUABindDN の形式は次のとおりです。
	`Prefix#LoginId#Suffix.`
	
	サービス カタログ（Service Catalog）は #LoginId# を、EUABindDN からサインインしようとしているユーザの loginId に置き換え、これを BindDN として認証で使用します。
	たとえば、EUABindDN を次のようにできます。
	`uid=#LoginId#,OU=People,dc=example,dc=com`
	
	このような場合、ユーザがサインアップの論理画面でログイン ID として scarter を入力すると、サービス カタログ（Service Catalog）は次の形式を使用します。
	`uid=scarter,OU=People,dc=example,dc=com`
	
	外部データソースにユーザをバインドします。

Person Lookup イベント

Person Lookup のすべてのイベント（Order on Behalf、Service Form、Authorization Delegate）は同じ動作および設定オプションを使用します。

ディレクトリ統合のイベントが有効でない場合、[個人検索（Person Search）] ウィンドウはサービス カタログ（Service Catalog）データベースにある個人情報を検索します。個人が選択された場合、その情報が使用されます。個人情報は更新されません。

ディレクトリ統合イベントが有効になっている場合、Person Lookup イベントは次の操作によって設定できます。

- 「Person Search」操作が最初の手順になっている必要があります。この操作は外部ディレクトリから個人情報を取得し、その情報を [Person Search] ウィンドウに表示します。人を選択すると、イベントに対して指定されているマッピングに従って、その人についての追加情報が取得され、コール側のコンテンツに提供されます。
- 「Import Person」操作がその次の手順です。この操作は、選択された個人のプロファイルを外部ディレクトリからサービス カタログ（Service Catalog）へインポートし、データを同期します。
- 「Import Manager」の操作は、「Import Person」手順の次の操作です。この操作は選択した個人のマネージャに関する情報を外部ディレクトリから検出して、その情報を持つサービス カタログ（Service Catalog）データベースを更新します。

Person Search 操作

Person Search 操作の設定により、検索条件を満たす人を表示するウィンドウの外観および動作が決まります。

個人をサービス カタログ（Service Catalog）にインポートするためには、すべての必須フィールドが正しい属性マッピングを持ち、空白以外の値を返す必要があります。必須の値が 1 つでも指定されていない場合は、デフォルトで、検索結果から対象となる人が除外されます。代わりに、そうした除外される人を検索結果に含めて、不完全な情報を持つ対象者としてフラグを付けることができます。

不完全な情報を持つ人は選択できません。

[個人検索（Person Search）] 操作の設定時

表 8-6　Person Search 操作

設定	値	コメント
Search Selectivity	• 不完全な情報を持つ人を表示する	デフォルトでは、不完全な情報を持つ人は検索結果ウィンドウから除外されます。
並び替え	• 名 • Last Name First Name • First Name Last Name • 姓 • No Sort	デフォルトでは姓で並び替えます。
検索結果の最大表示数（Max Results）		デフォルトでは、検索結果に表示される行数は 1000 です。

*（アスタリスク）ワイルドカード文字と個人の検索

個人の検索を設定およびテストする場合には、ワイルドカード文字としてアスタリスク（*）の使用に注意する必要があります。

ユーザに対して透過的になるよう、システムは検索文字列の最後に、常に * を付けます。したがって、ユーザが [姓（Last Name）] フィールドに john と入力して [検索（Search）] をクリックすると、ディレクトリにある「John」、「Johnson」、「Johnson」など、姓が john という語で始まるすべての個人が返されます。

[個人の検索（Search Person）] ダイアログ ボックスの検索文字列に、明示的に * の文字を入力することもできます。次に、ワイルドカード検索の使用例をいくつか示します。

- [Last Name] フィールドに「*」を入力して [Search] をクリックします。システムは、ディレクトリ内のすべての個人を返します。
- [姓（Last Name）] フィールドに **john*** と入力して [検索（Search）] をクリックします。これは実質的に、[Last Name] フィールドに **john** とだけ入力したことと同じです。システムは、姓が「john」という語で始まるディレクトリ内のすべての個人を返します。

- [Last Name] フィールドに ***john** と入力して [Search] をクリックします。システムは、姓に「john」という語が含まれる（「John」、「McJohn」、「Johnson」など）すべての個人を返します。
- [Last Name] フィールドに ***john*son** と入力して [Search] をクリックします。姓に「john」という語が含まれ、それに続いて（直後でなくてもかまいません）「son」という語が含まれているすべての個人が返されます。具体的には、「Johnson」、「Mcjohnson」、「Upjohningson」などです。

（注）　検索文字列内では、* は常にワイルドカード文字として扱われます。したがって、ユーザは文字 * を含むディレクトリの値を検索できません。その他のすべての特殊文字は、検索文字列で使用できます。

[Search Results] ウィンドウの設定

デフォルトでは、[人を選択します（Select Person）] ポップアップの [検索結果（Search Results）] ウィンドウに、ユーザの名、姓の順で表示されます。Administration モジュールで使用できる [個人（Person）] ポップアップの設定を変更すると、フィールドを表示に追加できます。

[個人のインポート（Import Person）] 操作

[個人のインポート（Import Person）] 設定は、アプリケーションの個人情報を、選択した個人に関する最新の情報で更新するか（[個人検索（Person Search）] イベントで [個人のインポート（Import Person）] を使用する場合）、または、ログインした個人に関する最新情報で更新するか（[ログイン（Login）] イベントで [個人のインポート（Import Person）] を使用する場合）を制御します。

表 8-7　Import Person 操作

設定	値	コメント
更新（Refresh）	• Refresh Person Profile • Refresh Period (Hours)	更新期間を空白のまま、またはゼロにすると、インポートごとに更新されます。こうすると、サービス カタログ（Service Catalog）データベースに外部ディレクトリの最新の変更が常に反映されることが保証されます。また、最後に更新されてから指定期間が経過したときだけ、ユーザのプロファイルを更新するように指定することもできます。
Create Associations	• Do Not Create Organizational Unit • Do Not Create Group	デフォルトでは、組織単位およびグループが作成されます（存在しない場合）。ロールはディレクトリ統合では作成できません。ロールは、個人をインポートする前に存在している必要があります。
Remove Existing Associations	• 組織 • グループ • [役割（Role）]	デフォルトでは、既存の組織単位、グループ、またはロールの関連付けは削除しません。

[マネージャのインポート（Import Manager）] 操作

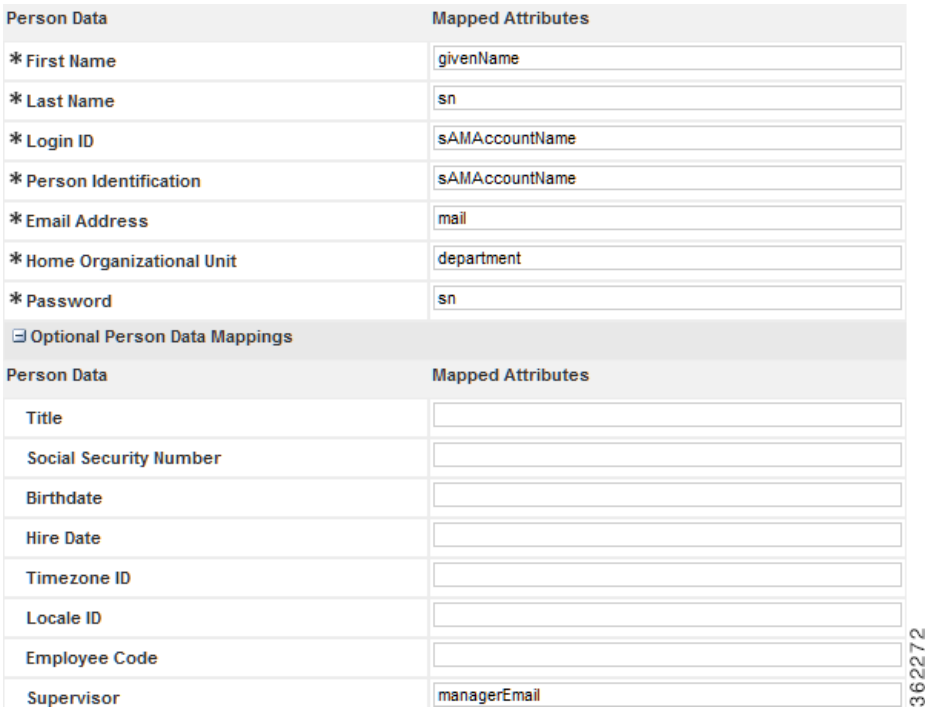
サービス カタログ（Service Catalog）では、承認と確認を動的に割り当てることができます。たとえば、要求のドル値が指定されたしきい値を超えている場合、特定の部署の部長による承認が必要な場合があります。別の要求では、要求者の直属の上司による確認が必要な場合があります。

こうしたビジネス ルールを実装するには、従業員のマネージャ（サービスを要求できる人）がサービス カタログ（Service Catalog）データベースにも存在している必要があります。[マネージャのインポート（Import Manager）] 操作は、この要件（従業員のデータとともにマネージャ（上司）のデータをインポートすること）をサポートしています。

[マネージャのインポート（Import Manager）] 操作の動作を制御するには:

- 従業員のマネージャを指定する、従業員ディレクトリ エントリの属性を指定します。
- すべての従業員について、指定された属性に、マネージャを一意に特定する値が挿入されていることを確認します。これは通常、ログイン ID または電子メール アドレスにします。
- [スーパーバイザ（Supervisor）] フィールドのマッピング（[オプションの個人データマッピング（Optional Person Data Mappings）] で一覧表示）で、マネージャ情報を保持する従業員データに属性を指定します。次の例では managerEmail 属性が使用されています。

図 8-3　個人データ



- [マネージャのインポート（Import Manager）] 設定で、元の個人に対して指定した [スーパーバイザ（Supervisor）] 属性に対応する値を持つマネージャのディレクトリ レコードのフィールドである、「マネージャ ID のキー フィールド」として指定します。

考えられるシナリオでは、個人の上司を一意に識別する単一の属性が、各個人のディレクトリ レコードに存在します。たとえば、従業員のディレクトリ レコードの属性 **manager_email** にマネージャの電子メール ID が含まれていると仮定します。マネージャの他の情報はありません。

表 8-8　マネージャ ID のキー フィールド

ソリューション	スーパーバイザ（Supervisor）	manager_email
	マネージャ ID のキー フィールド	email（マネージャのディレクトリ レコードにある電子メール属性）

別のシナリオとして、個人の上司の DN に完全に一致する属性がディレクトリ レコードに含まれている場合が想定されます。この属性の名前が **manager** であると仮定します。

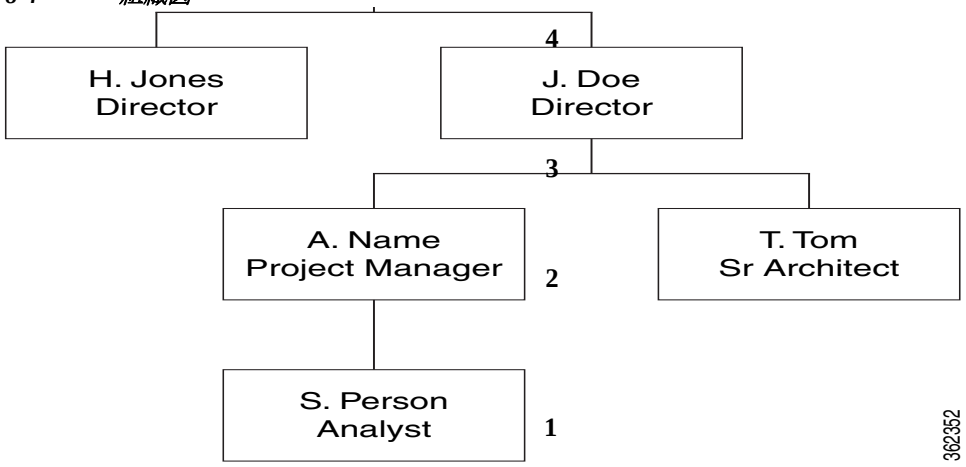
表 8-9　マネージャ ID のキー フィールド

ソリューション	スーパーバイザ（Supervisor）	manager
	マネージャ ID のキー フィールド	dn（DN は特別な属性で、検索文字列の前に付けられません）

監視階層の利用も必要になることがあります。

たとえば、次の組織図について考えてみましょう。

図 8-4　組織図



1	レベル 1	**3**	レベル 3
2	レベル 2	**4**	レベル 4

直属の上司の承認を得なければならない要求の場合、ツリーの 1 つ上のレベルに上がる、「相対的な」検索が必要です。

または、特定の要求で部長の承認が必要、といった場合には、「絶対的な」検索が必要です。現在の個人の地位が「部長」になるまで、人（マネージャ）がインポートされます。上記の例の S. Person の場合、2 名の追加の個人、つまり直属の上司である A. Name と、その上司であり部長である J. Doe が必要です。T. Tom の場合、インポートが必要なのは 1 回だけです。

絶対検索を使用している場合（指定した役職の人物を見つけるまで、連続的により高い権限レベルを持つすべてのマネージャをインポート）、その役職に相当する数字を割り当てる必要があります。

- 社内階層を分析し、すべての管理職に対応する数値を割り当てます。
- 各従業員の管理レベルを指定する属性を、従業員ディレクトリ エントリで特定します。たとえば、「paygrade」という属性を使用できます。
- すべての従業員について、指定された属性に値が入っていることを確認します。

- [管理レベル（Management Level）] フィールドのマッピング（[オプションの個人データマッピング（Optional Person Data Mappings）] で一覧表示）で、この情報を保持する属性を指定します。
- Import Manager の設定で、最も高いレベルのマネージャを「Maximum Level」としてインポートするよう入力します。

既知の値よりも上位の上司と同期しない場合は、検索の終わりを設定することができます。複数の値は、#value1#, #value2# のような形式で指定できます。

たとえば、UID が「scarter」である個人よりも上位の上司はインポートしないとします。この個人の [スーパーバイザ（Supervisor）] 属性は、その電子メール（scarter@email.com）にマップされています。このケースでは、Search Terminator を #scarter@email.com# に設定します。ディレクトリ統合は、scarter@email.com で上司としてレコードが見つかるとすぐに、上司の同期を停止します。

制約条件が Maximum Level または Search Terminator のいずれかを満たすとすぐに、上司の同期は停止します。

表 8-10　[マネージャのインポート（Import Manager）] 操作

設定	値	コメント
マネージャ ID のキー フィールド		従業員のマネージャ（スーパーバイザ）を一意に識別するディレクトリ属性。
最大レベル（Maximum Level）		絶対検索では、インポートするマネージャの最高管理レベルを示します。相対検索では、インポートする必要のある現在の従業員よりも上位階層のマネージャの数を示します。
Search Mode	• 絶対値（Absolute） • Relative	
検索終了値（Search Terminator）		マネージャのキー フィールドと一致すると、検索を終了する 1 つ以上の値。
Refresh options	• Refresh Person Profile • Refresh Period (Hours)	[個人プロファイルの更新（Refresh Person Profile）] チェック ボックスをオンにすると、Service Catalog 内のマネージャのプロファイルが更新されます。[更新期間（Refresh Period）] を空白のままにすると、同じ人に対して [マネージャのインポート（Import Manager）] イベントが発生するたびにプロファイルが更新されます。数値を指定すると、マネージャのプロファイルは指定期間内に一度だけ更新されます。
関連付け	• Do Not Create Organizational Unit • Do Not Create Group	[個人のインポート（Import Person）] の設定と同じ。
既存の関連付けの削除	• 組織 • グループ • [役割（Role）]	[個人のインポート（Import Person）] の設定と同じ。

カスタム コード操作

カスタム コード操作を使用して、アプリケーションでサポートされていないルーチンを呼び出します。カスタム コード操作は、サービス カタログ（Service Catalog）の操作を置き換えたり、補完したりできます。

表 8-11　カスタム コード操作

設定	値	コメント
Custom Code Operation Type	• シングル サインオン • 外部認証（External Authentication） • Import Person • Import Manager • Custom Code • Person Search	マッピング名を提供するために Java クラスを使用します。Java クラスの詳細については、Javadoc を参照してください。

ADS での SSO の設定

次の内容のファイル **jboss web.xml** を、ディレクトリ **ServiceCatalogServer/RequestCenter.war/'WEB-INF'** に作成します。

目次

```
<?xml version="1.0" encoding="UTF-8"?>
<jboss-web>
<security-domain>mySSO</security-domain>
</jboss-web>
```

web.xml の変更

（注）　これは、</context-param> の後ろかつフィルタの前に追加する必要があります。

```
<login-config>
<auth-method>EXTERNAL</auth-method>
</login-config>
```

standalone-full.xml の変更、セキュリティ ドメインの追加

（注）　*security–domains* を検索し、その後ろに次の内容を追加します。

```
<security-domain name="mySSO" cache-type="default">
      <authentication>
                    <login-module code="Client" flag="optional">
                    </login-module>
      </authentication>
</security-domain>
```

(注)　クラスタについては、ディレクトリ **C:\Install_dir\dist\RequestCenter.war** で、**jboss-web.xml** を作成し、**web.xml** を変更する必要があります。詳細については、『Cisco Prime Service Catalog 11.1.1』の「*Applying Patch or Customizations to the WildFly Cluster Setup*」の項を参照してください。

- **domain.xml**（wildfly-8.2.0.Final\domain\configuration）でキーワード *security-domains* で検索を行い、その下に次の内容を追加します。

```
<security-domain name="mySSO" cache-type="default">
      <authentication>
                  <login-module code="Client" flag="optional">
                  </login-module>
      </authentication>
</security-domain>
```

ディレクトリ LDAP 統合の設定

ディレクトリ統合の設定には Administration モジュールのディレクトリ オプションを使用する必要があります。基本的なプロセスは次のとおりです。

- **ディレクトリ統合を有効にします。**Administration モジュールの [設定（Settings）] タブで [ディレクトリ統合（Directory Integration）] をクリックして、ディレクトリ統合を有効にします。
- **データソース情報を設定します。**Administration モジュールの [ディレクトリ（Directories）] タブの [データソース（Datasources）] 領域を使用して、ディレクトリ サーバに接続するデータソースを設定します。データソース名、説明、プロトコル、サーバ製品、認証方法などの情報が必要です。
- **マッピングを設定します。**[Administration module's Directories] タブの [Mappings] 領域を使用して、アプリケーション データをディレクトリ サーバ データにマップします。マッピングにより、ユーザや個人の全プロファイルおよび関連するすべてのエンティティが更新されます。これらのエンティティには住所、連絡先、場所、1 つ以上のグループ関係、1 つ以上の組織単位（OU）関係、1 つ以上のロール関係があります。
- **イベントを設定します。**Administration モジュールの [ディレクトリ（Directories）] タブの [イベント（Events）] 領域を使用して、ディレクトリ統合の動作を設定します。[ログイン（Login）] イベントと [個人検索（Person Lookup）] イベントは、シングル サインオン（SSO）、エンドユーザ認証（EUA）、個人のインポート、マネージャのインポート、および個人検索などの操作を含むように設定できます。
- 必要に応じて、クライアントをカスタマイズするために、関連エンティティとの**カスタム コード インターフェイスの設定**を行います。これには、ディレクトリの Java クラス属性マッピング、ディレクトリ サーバ API、および個人のインポートなどがあります。

ディレクトリ統合の有効化

ディレクトリ統合を有効にするには:

手順 1　管理権限を持つアカウントを使用してログインし、Administration モジュールを選択します。

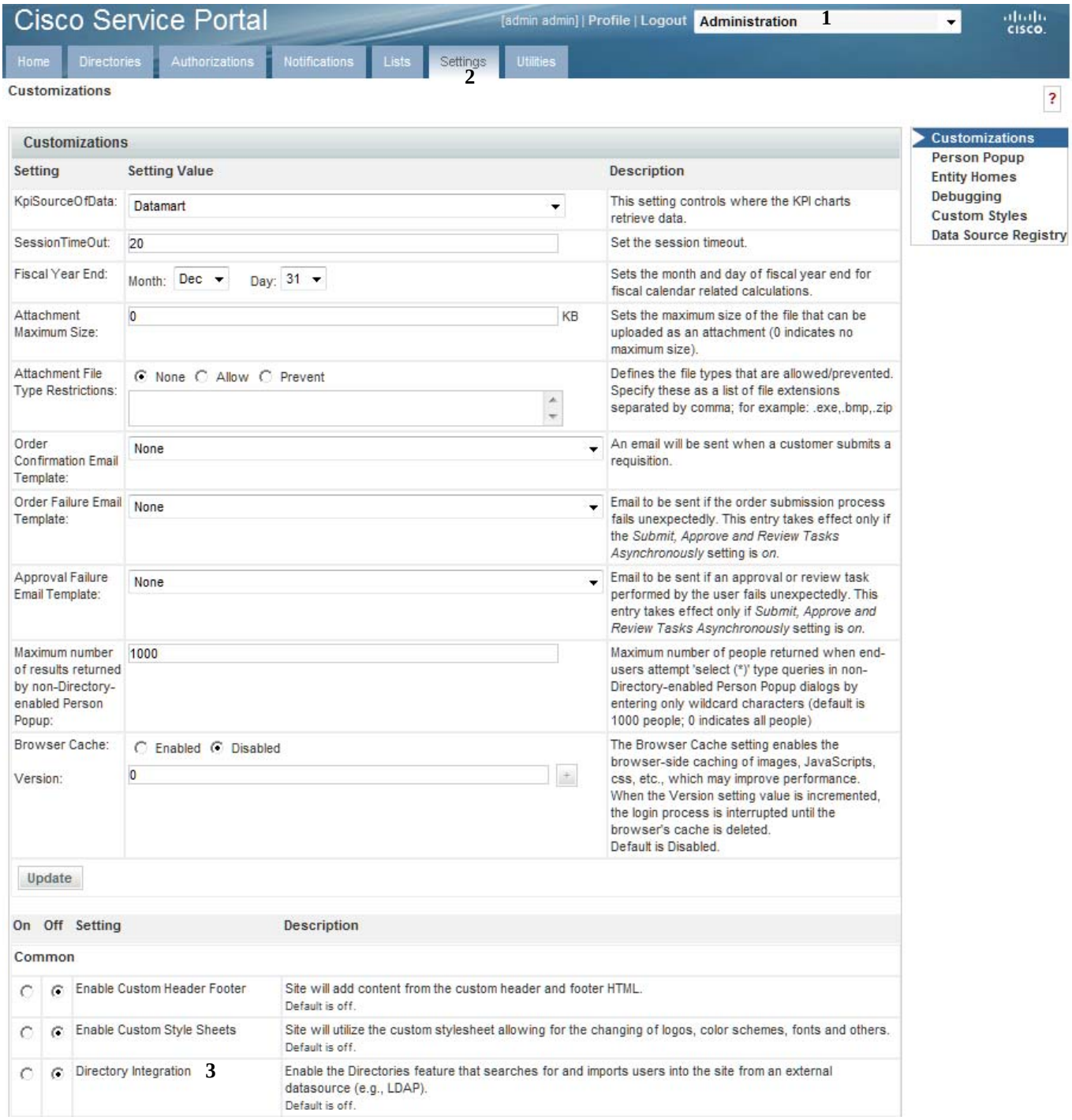
手順 2　[設定（Settings）] タブをクリックします。

手順 3　[ディレクトリ統合（Directory Integration）] の横の [オン（On）] をクリックします。

手順 4　[カスタマイゼーション（Customizations）] 画面の下部で [更新（Update）] をクリックします。

これで、ディレクトリ統合が有効になりました。（ディレクトリ統合の有効化を参照）。

図 8-5　　ディレクトリ統合の有効化



1	[Administration] モジュール	**3**	ディレクトリ統合の設定
2	[設定(Settings)] タブ		

ディレクトリ統合の設定

Administration モジュールの [Directories] タブを使用して、ディレクトリ統合の多数の設定を行います。

図 8-6　[Directory Integration] 領域

1	[Administration] モジュール
2	[ディレクトリ（Directories）] タブ

ディレクトリ統合を設定するには:

手順 1　管理権限を持つアカウントを使用してログインします。

手順 2　ドロップダウン メニューから、[Administration] を選択します。

手順 3　[ディレクトリ（Directories）] タブをクリックします。

[ディレクトリ統合（Directory Integration）] ページが表示されます。ディレクトリ統合が有効にされると、これらの設定が有効になります。

データソース情報の設定

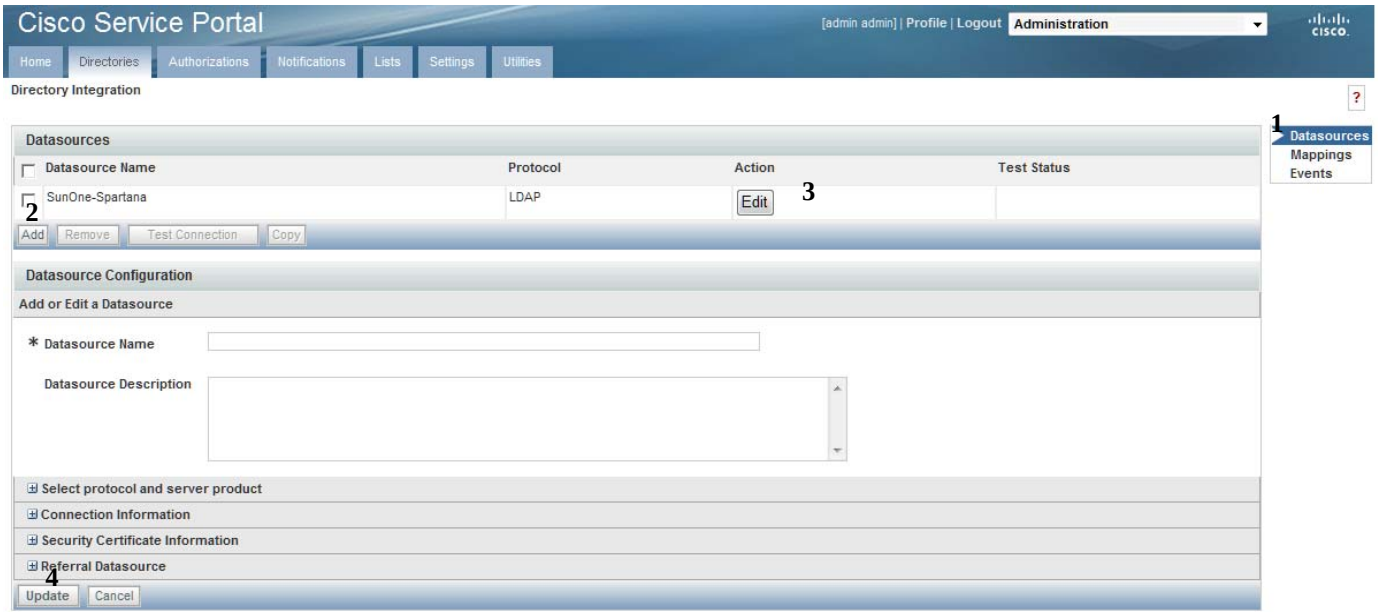
以降の項では、データソース特有の情報設定について解説します。タスクには次のものがあります。

- **データソースの追加または編集** - まだデータソースが存在しない新しいインストール環境に、データソースを追加する必要があります。データソースが存在する場合は、それを編集できます。
- **SSL 接続用のサーバ証明書の追加** - これが必要なのは、接続方式として [SSL] を選択した場合だけです。
- **参照データソースの追加** - 必要な場合のみ。
- **接続のテスト** - 接続できることを確認するために、必ず接続をテストする必要があります。

データソースの追加または編集

データソースは少なくとも 1 つ定義してください。新しいデータソースを追加するには、次の手順を実行します。

図 8-7　　データソースの追加または編集



1	[データソース(Datasources)] オプション	**3**	[データソースの編集(Edit Datasource)] ボタン
2	[Add Datasource] ボタン	**4**	[Update] ボタン

手順 1　Administration モジュールを選択してから [ディレクトリ(Directories)] タブをクリックして、[ディレクトリ統合(Directory Integration)] ページに移動します。

手順 2　ページ ナビゲータで [データソース(Datasources)] オプションをクリックします(まだ選択されていない場合)。

手順 3　[追加(Add)] をクリックします。新しいデータソースを追加する代わりに、既存のデータソースを編集するには、リストで目的のデータソースの横にある [編集(Edit)] をクリックします。

[データソース設定(Datasource Configuration)] 領域が拡大します。

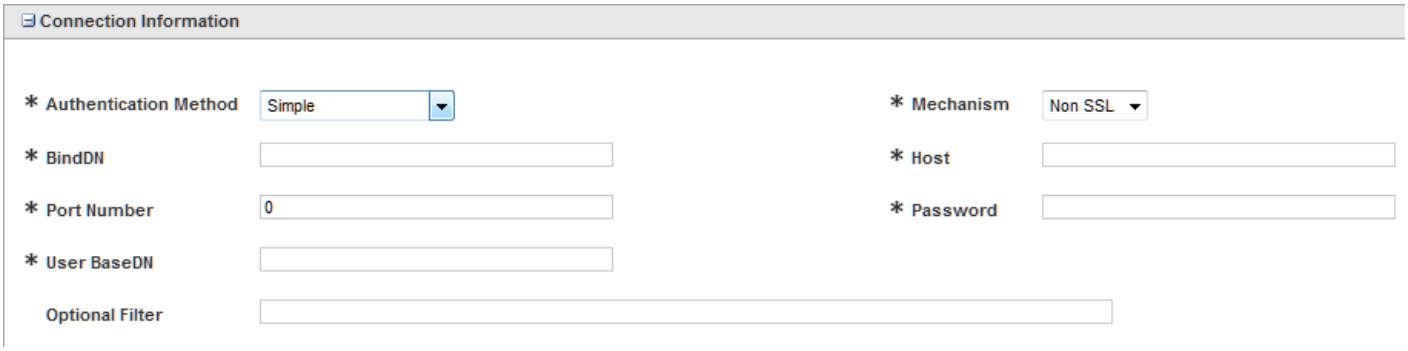
手順 4　[データソース名(Datasource name)]、[データソースの説明(Datasource Description)]、および必要な設定に入力します。隣接領域のすべての設定にアクセスするには ⊞、をクリックします。これらの設定の詳細については、データソース ワークシートを参照するか、または次の項を参照してください。

手順 5　[更新(Update)] をクリックします。

接続情報の設定

データソースへの接続に使用する接続プロトコルおよびユーザ クレデンシャルを指定します。

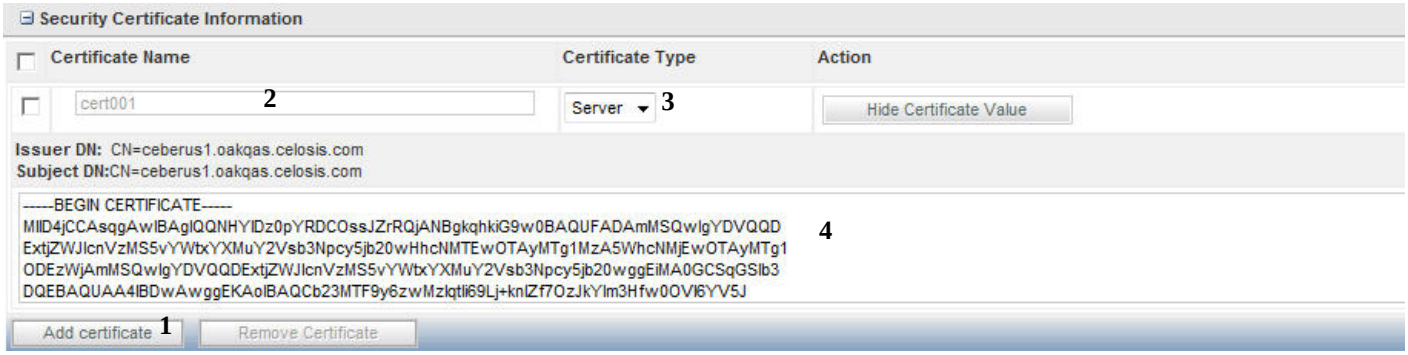
図 8-8　接続情報の設定



証明書の設定

接続方式として [SSL] を選択した場合は、ディレクトリ統合システムの証明書を指定する必要があります。

図 8-9　セキュリティ証明書の設定



1	[Add certificate] ボタン	**3**	[Certificate Type] ドロップダウン メニュー
2	[Certificate Name] フィールド	**4**	[Certificate Value] フィールド

証明書を設定するには:

手順 1　Administration モジュールを選択してから [ディレクトリ(Directories)] タブをクリックして、[ディレクトリ統合(Directory Integration)] ページに移動します。

手順 2　ページ ナビゲータで [データソース(Datasources)] オプションをクリックします(まだ選択されていない場合)。

手順 3　証明書を追加するデータソースの横にある [編集(Edit)] をクリックします。

手順 4　[証明書の追加(Add Certificate)] をクリックします。

手順 5　証明書に名前を付けます。証明書のエイリアス名には、スペースや特殊文字を使用しないでください。

手順 6　[証明書タイプ（Certificate Type）] ドロップダウン メニューから、証明書タイプを選択します。

手順 7　証明書フィールドに証明書の値（VeriSign などのベンダーから取得）を貼りつけます。

手順 8　[更新（Update）] をクリックします。

照会用データソースの設定

複数のデータソースを設定した場合は、選択したデータソースに対して、照会用システムとしてデータソースを指定することができます。この方法では、選択されたデータソースに対してシステムが検索を実行するたびに、照会用のすべてのデータソースも検索されます。

照会用データソースは、一致が見つかるまで指定された順序で検索されます。
検索条件で 1 つ以上のレコードが返されたときに、一致が見つかったことになります。

照会用データソースは、通常、ディレクトリ情報が複数のディレクトリ間で分割される場合に使用されます。たとえば、企業の複数の事業部で、それぞれ独自のディレクトリを保持することができます。

図 8-10　照会用データソースの設定

1	[照会の追加（Add Referral）] ボタン	**3**	[Mapping Name] ドロップダウン メニュー
2	[Datasource Name] ドロップダウン メニュー		

照会用データ ソースを設定するには:

手順 1　Administration モジュールの [Directory Integration] ページに移動します。

手順 2　ページ ナビゲータで [データソース（Datasources）] オプションをクリックします（まだ選択されていない場合）。

手順 3　照会用データソースを設定するデータソースの横にある [編集（Edit）] をクリックします。

手順 4　[Add Referral] をクリックします。

手順 5　[照会用データソース（Referral Datasource）] 領域が表示されます。[データソース名（Datasource Name）] ドロップダウン メニューからデータソース名を選択し、[マッピング名（Mapping Name）] ドロップダウン メニューからマッピング名を選択します。

手順 6　[更新（Update）] をクリックします。

接続のテスト

必要な設定手順をすべて完了すると、次はディレクトリ統合の接続をテストできます。

図 8-11　接続のテスト

1	[テスト接続(Test Connection)] ボタン	2	[テストステータス(Test Status)] カラム

接続をテストするには:

手順 1　Administration モジュールの [Directory Integration] ページに移動します。

手順 2　ページ ナビゲータで [データソース(Datasources)] オプションをクリックします(まだ選択されていない場合)。

手順 3　データソース名の左にあるチェック ボックスをオンにして、テストするデータソースを選択します。

手順 4　[テスト接続(Test Connection)] をクリックします。

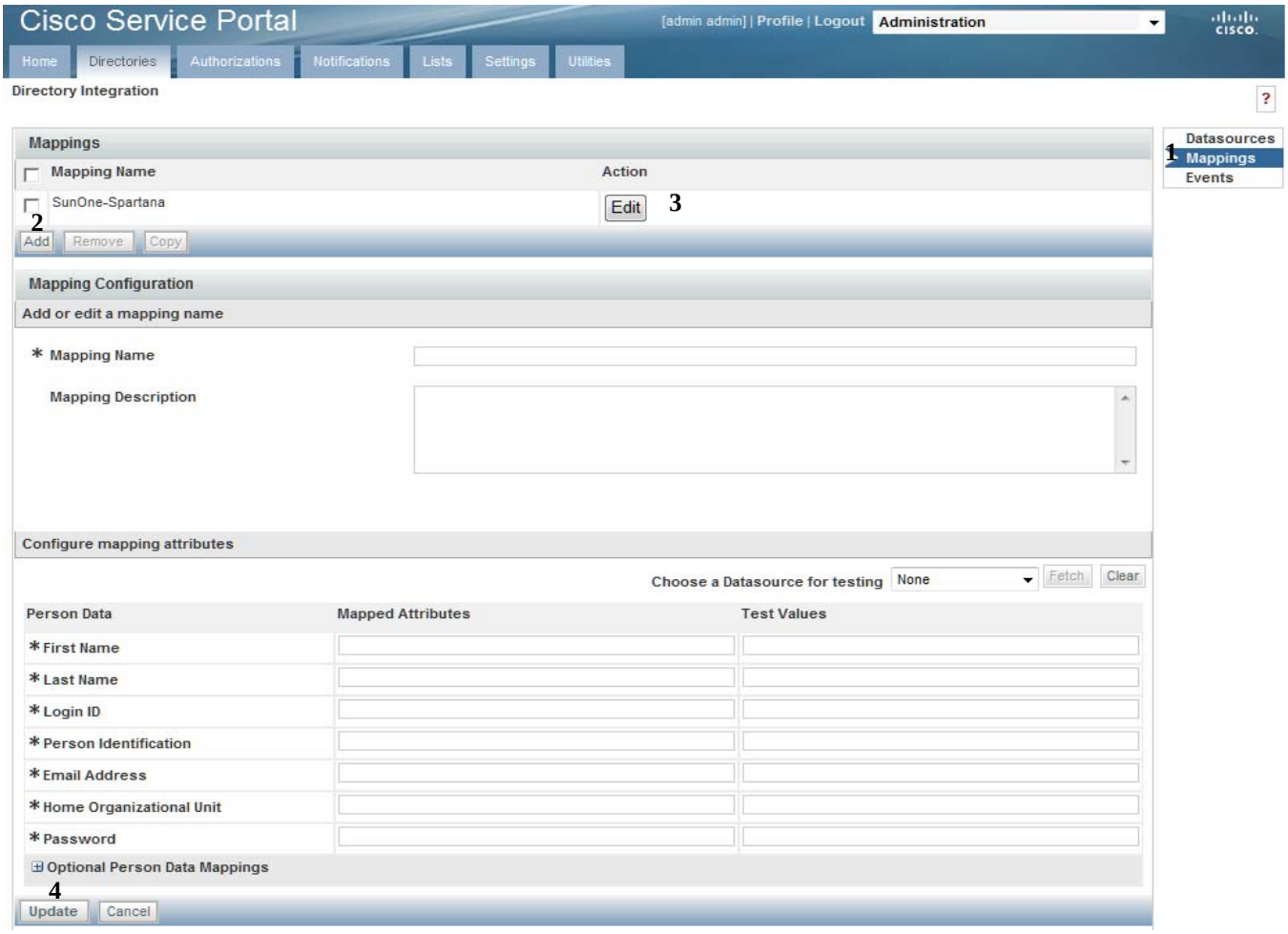
接続が成功した場合は [テストステータス(Test Status)] カラムに OK と表示され、接続が失敗した場合は  が表示されます。

マッピングの設定

Administration モジュールの [ディレクトリ(Directories)] タブで [マッピング(Mappings)] 領域を使用して、サービス カタログ(Service Catalog)データをディレクトリ サーバ データにマップします。

マッピングを設定するには、マッピングの設定を参照し、次の手順に従います。

図 8-12　　マッピングの設定



1	[マッピング(Mappings)] オプション	**3**	[マッピングの編集(Edit Mapping)] ボタン
2	[マッピングの追加(Add Mapping)] ボタン	**4**	[Update] ボタン

手順 1　Administration モジュールの [ディレクトリ統合(Directory Integration)] ページに移動します。

手順 2　ページ ナビゲータで、[マッピング(Mappings)] オプションをクリックします。

手順 3　新しいマッピングを追加するには、[追加(Add)] をクリックし、既存のマッピングを編集するにはリストの目的のマッピングの横にある [編集(Edit)] をクリックします。

[マッピング設定(Mapping Configuration)] 領域が拡大します。

手順 4　マッピング ワークシートに記載されている要件に基づいて、マッピングの名前、説明、および属性を設定します。[Person Data] セクションに表示された、先頭にアスタリスク(*)が付いているマッピングは必須です。⊞ボタンをクリックしてオプションのマッピングを設定し、[Optional Person Data Mappings] セクションを展開することもできます。

手順 5　[更新(Update)] をクリックします。

マッピング フィールドには、次に示すように、シンプル、複合、式、Java のマッピング タイプを指定できます。

マッピング タイプ

ここでは、指定可能なマッピング タイプについて説明し、正しいサンプルのマッピングを示します。また、式マッピングの例についても説明します。次の表に、サポートされているマッピング タイプを示します。

表 8-12　マッピング タイプ

マッピング タイプ	説明
[シンプル（Simple）]	1 つのディレクトリ属性をフィールドにマップします。これは単純な 1 対 1 マッピングです。次に例を示します。 Person Field: First Name Directory Attribute: givenName
Composite	属性の組み合わせをフィールドにマッピングします。# を使用して各属性名を区切ります。マッピングはリテラルを含むことができます。次に例を示します。 Person Field: Email Directory Attributes: #givenName#_#sn#@#domain#.com
式	式では、正規表現とパターン マッチングを使用してマッピングが得られます。詳細については、式マッピングを参照してください。
Java Class	簡易、複合、または式の各マッピングで目的とする機能を提供できない場合は、Java マッピングを使用します。これには、Java クラスを記述し、アプリケーション サーバ上の適切なディレクトリに、コンパイルしたクラス ファイルを配置する、という作業が含まれます。詳細については、Java クラス マッピングを参照してください。

簡易マッピングと複合マッピング

次の表は、必須フィールドに対する簡易マッピングと複合マッピングの例を示しています。

表 8-13　マッピングの例

個人データ	ディレクトリの値
名	givenName
姓	sn
ログイン ID	uid
Person Identification	uid
電子メール アドレス（Email Address）	#givenName#_#sn#@#company#.com
Home Organizational Unit	OU
[パスワード（Password）]	Uid

式マッピング

式マッピングを使用すると、式が一致するパターン（正規表現）に基づいて、値を条件付きで属性に割り当てることができます。システムの式マッピングは Perl5 Regular Expression Language を使用し、Java の条件演算子に似た構文と組み合わせて、一致させるパターンを指定します。構文:

```
expr:<expression>=(<patternlist>)?(<valuelist>):<default>
```

説明:

expr	式マッピングが使用されることを示すためのプレフィックスです。
<expression>	一致させるための式です。
<patternlist>	パイプ（|）で区切られたパターンのセットです。
<valuelist>	パターンのセットに対応する、パイプ（|）で区切られた値のセットです。それぞれの値は、式が対応するパターンと一致した場合の戻り値を指定します。
<default>	<patternlist> のパターンが <expression> と一致しなかった場合に使用される戻り値です。

次に例を示します。

```
expr:<expression>=
(<pattern1>|<pattern2>…<patternn>)?(<value1> | <value2>  <valuen>):<default>
```

<expression> が <pattern1> に一致すると、<value1> を返します。

<expression> が <pattern2> に一致すると、<value2> を返します。

<expression> がどのパターンにも一致しない場合、<default> を返します。

各要素（expression、pattern、または value）には、# 記号で区切って、ディレクトリの属性名を含めることができます。たとえば、パターンを「#givenName#_#sn#」のように指定できます。ここで #givenName# と #sn# はどちらも属性名です。

また、括弧を使用して一連のパターン要素を 1 つの要素にグループ化できます。括弧内のパターンと一致した場合、$1、$2 などの形式で後方参照を使用して、以前に一致したパターンを参照することができます。

式データ マッピングの例

ディレクトリ統合に適用される式の簡単な利用法として、ディレクトリ内でコード化されている 1 つ以上の値を、よりわかりやすい記述、または広範なカテゴリに変換することができます。たとえば、サービスによっては、従業員と請負業者の区別が必要になることがあります。costCenter 属性は、「000000」が請負業者用です。したがって、[Employee Type] フィールドに次の式を適用することができます。

```
expr:#costCenter#=(000000)?(Contractor):Employee
```

もうひとつの式の利用法は、ソース属性が空白の場合に、フィールドにデフォルト値を入力することです。これは多くの場合に、ディレクトリ データが標準化できるまでの「応急処置」となります。また、たとえば外部の請負業者に部門が割り当てられない場合などの標準にすることができます。次の式を [Home OU] フィールド（マッピングの必須フィールド）に適用することができます。

```
expr:#DeptLevel2#=(.+)?(#DeptLevel2#):Contractors
```

この式では、該当する場合に DeptLevel2 属性を使用することも、ユーザの Home OU に対して事業部門のデフォルトを「Unknown」にすることもできます。

同様に、この式を使用して、一連の入力値を、異なる戻り値のセットに変換できます。これは、case 文、またはネストされた if/then 構造と同じです。たとえば、次の式を [Locale ID] フィールドに適用し、ユーザのロケーションに基づいて、そのユーザに言語を割り当てることができます。

```
expr:#country#=(United States | Germany)?(en_US | de_DE):en_US
```

ユーザの国が United States の場合は、言語を米国英語に設定し、Germany の場合は言語をドイツ語に設定します。その他の国の場合は、言語を米国英語に設定します。

正規表現では、ソース属性の長さ、および英数字で構成されているかどうかをチェックできます。たとえば、郵便番号が数値データ タイプとして格納され、先行ゼロが切り捨てられることがあります。先行ゼロを回復するには、[Company Postal Code] フィールドに次のような式を適用できます。

```
expr:#postalCode#=(^[1-9][0-9][0-9][0-9]$)?(0#postalCode#):#postalCode#
```

postalCode 属性がちょうど 4 桁の場合は、属性に先行ゼロの値を追加します。これにより、郵便番号 **1701** は **01701** に変換され、特定のパターンに一致しないすべてのソース値は、変更されずにそのままになります。

正規表現の同様の使用法として、属性値の形式が、予想したパターンと一致するかをチェックする、というものがあります。有効なマネージャのユーザ ID は、必ず 2 文字と、それに続く一連の数字で構成されている場合について考えてみましょう。たとえば、有効な ID は fd1024 や ID3839 となります。次の式を使用できます。

```
expr:#manager#=(cn=([a-zA-Z][a-zA-Z][0-9]+),.*)?($1):None
```

式、パターン、または戻り値で属性を使用できます。

```
expr:#sn#, #givenname#=(Smith.*|Doe, John)?(All Smiths|Only John):Others
```

```
expr:#sn#, #givenname#=(Smith.*|Doe, John)?(#givenname#|Only John):Others
```

パターンと照合するためにあらゆる試行を作成する前に、Doe、Jane などのディレクトリ レコードの姓および名が 1 つの文字列に組み合わされます。

パターンの一部を抽出するには、括弧および後方参照を組み込むと役立ちます。たとえば個人が所属している組織は、識別名（dn）属性内に組み込まれることがよくあります。

```
dn: cn=plee,ou=Corporate,dc=InfoSys,dc=com
```

[Home Organizational Unit] フィールドにマップされる式は、次のような形式になります。

```
expr:#dn#=((cn=[^,]+,ou=([a-zA-Z]+),dc=InfoSys,dc=com)?($1):Default
```

戻り値「Corporate」は後方参照値 $1 で、これは最初の括弧 ([a-zA-Z]+) 内の式で照合したパターンに相当します。

複数のフィールドが含まれているオーバーロード属性を解析するために、後方参照変数の使用が必要になることがあります。たとえば、1 つの属性に、個人の勤務先住所（ビル名、階数（レベル）、オフィスなど）を含めることが可能です。

```
location=Corporate Headquarters-Fifth Floor-Office #5F
```

異なる後方参照変数を、次の値として使用すると、同じパターンを使用して、式内で 3 つの要素を照合することができます。

Office Building	expr:#Location#=(([^-])+-([^-])+-(.*))?($1): Unknown
Building Level	expr:#Location#=(([^-])+-([^-])+-(.*))?($2): Unknown
Cubic Location	expr:#Location#=(([^-])+-([^-])+-(.*))?($3): Unknown

Java クラス マッピング

ディレクトリ データをフィールドにマップするためのカスタム Java クラスを実装するには、Java プログラミングをよく理解し、Java 開発環境が用意されている必要があります。

すべてのカスタム マッピング クラスは、「ディレクトリ統合でのカスタム コードの使用」セクション（8-36 ページ）のガイドラインに従っています。マッピング クラスは IEUIAttributeMapping インターフェイスを実装する必要があります。

開発者は以下のガイドラインに従って、カスタム コード モジュールをテストおよびインストールする必要があります。

1. 最適な Java IDE をインストールし、カスタム マッピング コードを開発するためのプロジェクトをセットアップします。
2. 要件を満たすようにカスタム コード ファイルを編集します。
3. コンパイルします。
4. カスタム Java クラスは、Service Catalog サービスがアクセスできるように、サービス カタログ（Service Catalog）の Web アーカイブ（war）にインストールする必要があります。RequestCenter.war/WEB-INF/classes にディレクトリを作成し、パッケージと一致するようにします。このようなディレクトリは通常、次のような名前になります。

   com/newscale/client/<clientname>（com/newscale/client/aib など）。
5. CustomMapping.class ファイルを、前の手順で作成したディレクトリにコピーします。
6. Service Catalog サービスを再起動します。
7. クラス ファイルの完全修飾名を Mapped Attribute として指定し、フィールドに値が挿入されるようにします。
8. ディレクトリ テスト機能を使用して、カスタム コードをテストします。
9. ソースを適切なリポジトリに保存します。

マッピングのテスト

マッピング テスト機能を使用して、自身のデータ マッピングが正しく設定されていること、およびディレクトリ サーバから正しい値を取得することをテストします。

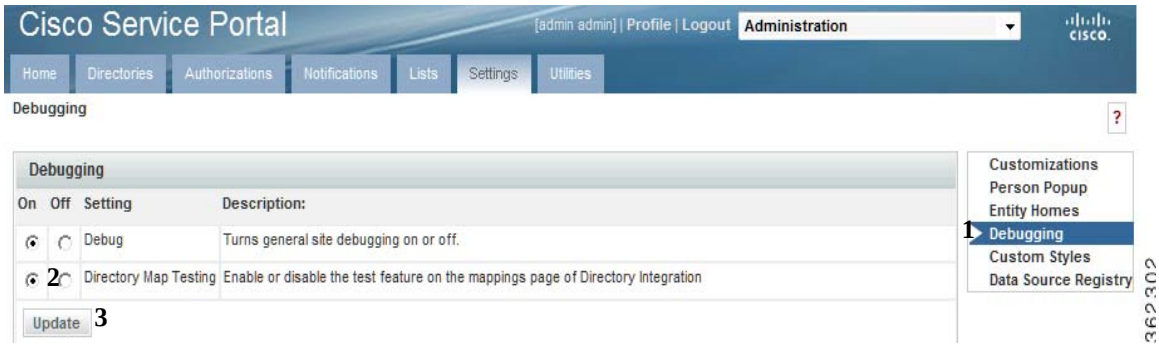
データ マッピング テスト機能の使用には、次の処理が含まれます。

- データ マッピング テスト機能を有効にする
- データ マッピング テスト コントロールの使用

ディレクトリ マップ テスト機能の有効化

ディレクトリ マップのテスト機能を有効にするには、マッピング テストの有効化を参照し、次の手順に従ってください。

図 8-13　　マッピング テストの有効化



1	[Debugging] オプション	**3**	[Update] ボタン
2	ディレクトリ マップ テストの設定		

手順 1　Administration モジュールの [Settings] タブをクリックして [Settings] ページを表示します。

手順 2　ページ ナビゲータで、[デバッグ（Debugging）] オプションをクリックします。

[Debug Settings] ページが表示されます。

手順 3　[ディレクトリマップテスト（Directory Map Testing）] 設定の横にある [オン（ON）] をクリックします。

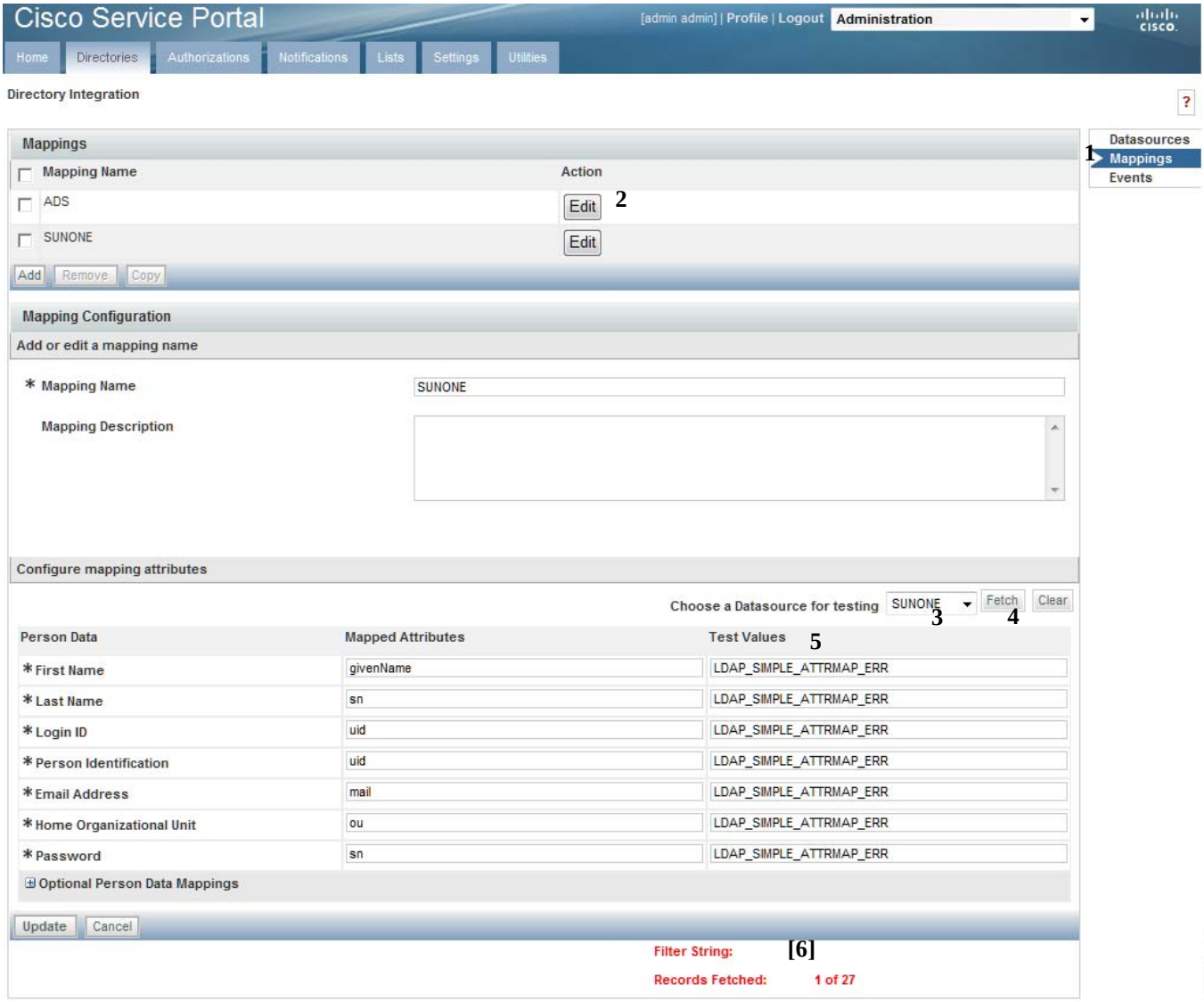
手順 4　[更新（Update）] をクリックします。

データ マッピング テスト機能が有効になります。[Data Mapping] タブにアクセスすると、マッピング テスト コントロールに示すように、次の追加機能が表示されます。

- [テストするデータソースの選択（Choose a Datasource for Testing）] ドロップダウン メニュー
- 取得（Fetch）
- クリア（Clear）
- テスト値（Test Values）

データ マッピング テスト コントロールの使用

図 8-14　　マッピング テスト コントロール



1	[Mappings] オプション	**4**	[Fetch] ボタン
2	[編集(Edit)] ボタン	**5**	[Test Values] カラム
3	[Choose a Datasource for testing] ドロップダウン メニュー	**6**	[Test summary] 領域

データ マッピング テスト コントロールの使用方法

手順 1　[Mapping] ページが表示されていない場合は、[Mappings] をクリックします。

手順 2　テストするマッピングの横にある [編集(Edit)] をクリックします。

手順 3　[テストするデータソースの選択(Choose a Datasource for testing)] ドロップダウン メニューから目的のデータソースを選択します。

手順 4　[テスト値(Test Values)] カラムにテスト値を入力します。簡易、複合、Java、および式の各マッピングを使用できます。

手順 5　[Fetch] をクリックします。

手順 6　[Test Values] カラムにテスト値が表示され、ページの下部に結果の概要が示されます。

(注)　[Fetch] により値が返されるのは 1 つのデータ ソースのみで、照会先は検索されません。照会先の検索が統合されているとデバッグが難しいため、便宜上そのようになっています。

手順 7　[取得(Fetch)] ボタンの右にある [クリア(Clear)] をクリックして、希望するマッピングが設定されるまで、新しい値をさらに試します。

ディレクトリ統合イベントの設定

Administration モジュールの [Directories] タブで [Events] 領域を使用して、次のイベントに対するディレクトリ統合の動作を設定します。

- ログイン(Login)
- Person Lookup for Order on Behalf
- Person Lookup for Service Form
- Person Lookup for Authorization Delegate

イベントを設定するには、イベントの設定を参照し、次の手順に従います。

手順 1　Administration モジュールの [Directory Integration] ページに移動します。

手順 2　ページ ナビゲータの [イベント(Events)] をクリックし、[イベント(Events)] ページを表示します。

手順 3　設定するイベントのタイプの横にある [編集(Edit)] をクリックします。

[Event Configuration] 領域が表示されます。

手順 4　[イベントステータス(Event Status)] ドロップダウン メニューから [有効(Enabled)] を選択し、イベントを有効にします。

手順 5　[Add step] をクリックして手順を追加し、選択したイベントが発生したときにシステムで開始されるようにします。

手順 6　追加した手順に関連付ける操作を選択します。

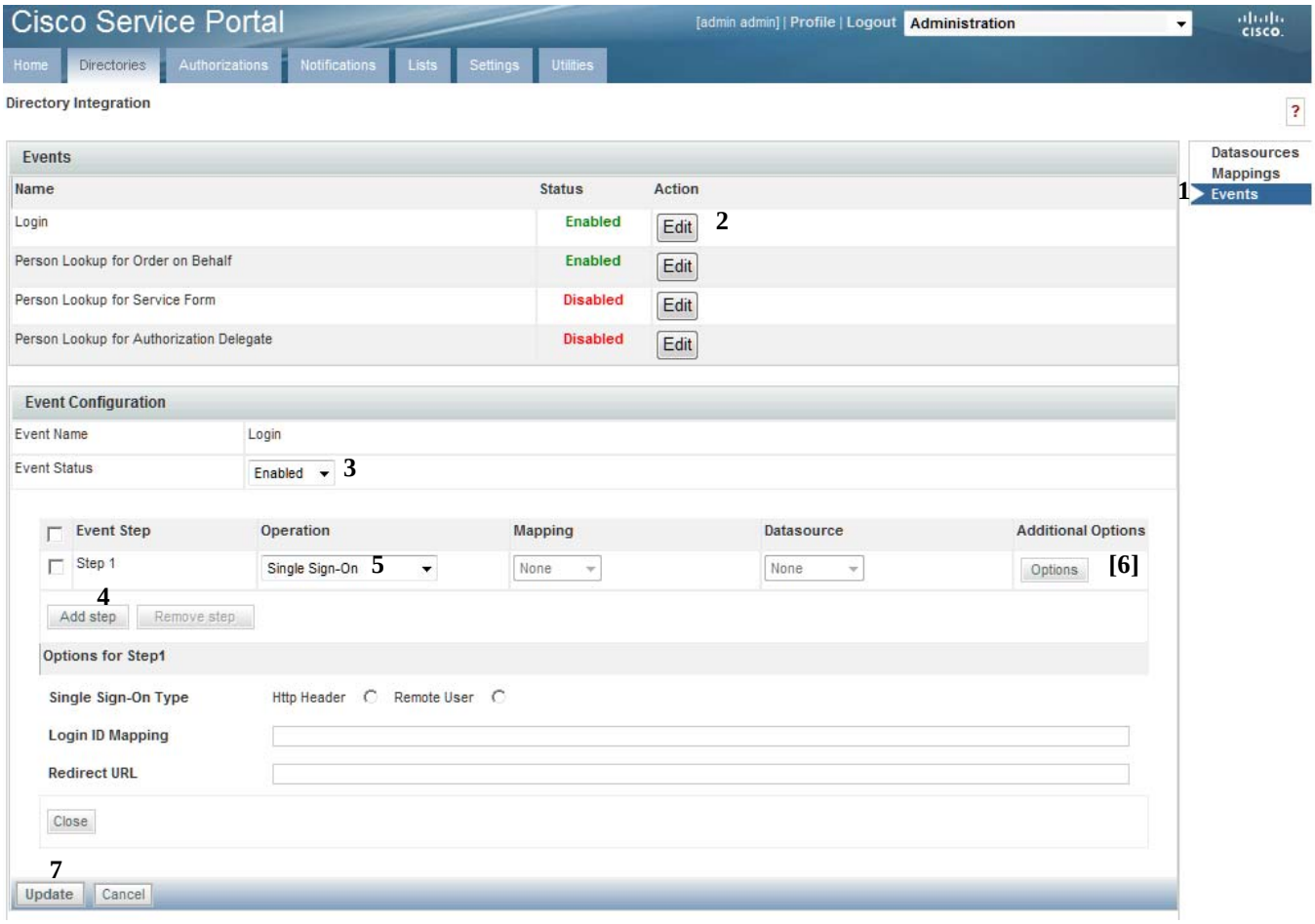
- SSO や EUA など、いくつかの操作は一部のイベント タイプに適用できませんが、このメニューではすべての操作を使用できます。

手順 7　[オプション(Options)] をクリックして、選択した操作に関連付けるオプションを設定します。[Options] 領域が表示されます。[オプション(Options)] 領域は、選択した操作によって異なります。使用できる操作、およびその操作のオプションについては、次の項で詳しく説明します。

手順 8　関連付けるオプションを設定します。使用できる操作および操作の設定用のオプションの詳細は、この章のディレクトリ イベントに関する項を参照してください。

手順 9　[Update] をクリックし、追加する各手順および操作に対して、これらの手順を繰り返します。

図 8-15　　イベントの設定



1	[Events] オプション	**5**	[Operation] ドロップダウン メニュー
2	編集（Edit）	**6**	オプション（Options）
3	[Event Status] ドロップダウン メニュー	**7**	[更新（Update）]
4	[ステップの追加（Add step）]		

ディレクトリ統合でのカスタム コードの使用

ディレクトリ統合フレームワークは、「Login」および「Person Lookup」イベントの柔軟性を利用し、カスタマイズできるよう設計されています。

Administration モジュールの [Directories] タブでは、すべてのイベントに対する標準操作を使用できます。そのような標準の操作としては、SSO、External User Authentication、Import Person、Import Manager、および Person Search などがあります。

これらの標準操作ではビジネス シナリオを満たせない場合、[Directories] タブには、カスタム Java コードを実行するためのインターフェイスも用意されています。このカスタム コードは、この章に記載されたインターフェイスに準拠している必要があり、サービス カタログ（Service Catalog）が公開している API を使用して、すべてのカスタマイズ ソリューションを開発する必要があります。

以下に、イベントの操作をカスタマイズできるシナリオの有効な使用例を示します。

表 8-14　使用例

状況	対策
HttpServletRequest による SSO ヘッダー入力の形式を解析できない	ベンダーと SSO との統合をサポートするために、ユーザのクレデンシャルを取得するカスタム コード SSO 操作を提供する。
サービス カタログ（Service Catalog）以外の Web サービスまたはデータベースでユーザを認証する	カスタム コード External Authentication 操作を提供する。
会社のメイン ユーザ リポジトリが、LDAP ディレクトリ以外のデータベースにある	カスタム コード External Authentication、およびカスタム コード Import Person 操作を提供する。

ディレクトリ統合カスタム コード フレームワークでは、外部データ ソースのレコードから個人またはユーザ プロファイルの特定のフィールドに対する、複雑な取得ロジックを提供するよう実装可能なインターフェイスも定義されます。

ディレクトリ統合の Public API およびインターフェイスには次のものが含まれています。

- **カスタム コード操作インターフェイス**。ディレクトリ統合の操作をカスタマイズするために使用します。
- **カスタム Java クラス マッピング インターフェイス**。レコードから外部データ ソースの特定の属性を取得する場合に、カスタマイズされた取得を提供します。
- **ディレクトリ サーバ API**。外部データ ソースで照会または認証を行い、レコードを取得するために使用します。
- **個人のインポート/更新 API**。サービス カタログ（Service Catalog）データベースの個人属性を更新するために使用します。

一般的なカスタム コード プロジェクトには、次のタイプのアクティビティが含まれています。

- カスタム コードの必要性を確認する。
- Administration モジュールの [ディレクトリ（Directories）] タブを設定して、カスタム コードで使用されるデータ ソースを含める。該当する場合は、カスタム コードで使用する可能性のあるマッピングを含める。
- カスタム コードを開発する。公開 API、およびディレクトリ統合タスク用にシスコから提供されたインターフェイスについて理解する必要があります。
- カスタム コードをビルドおよび導入する。
- カスタム コードを使用するよう、Administration モジュールの [Directories] タブを設定する。

次のディレクトリ統合の操作に、ディレクトリ統合の操作を詳しく示します。

表 8-15　ディレクトリ統合の操作

操作	目的	入力	出力
シングル サインオン	ユーザのログイン名を識別する	HttpServletRequest	ログイン名（Login Name）
外部認証（External Authentication）	外部データソースでユーザを認証する	ログイン名（Login Name）[パスワード（Password）]	ユーザの認証
Person Search	名または姓が一致する個人のリストを取得する	姓および名	個人のリスト
Import Person	外部データソースからサービス カタログ（Service Catalog）データベースに個人をインポートする	ログイン名（Login Name）	インポートされた個人情報（managerID も含む）
Import Manager	外部データソースからサービス カタログ（Service Catalog）データベースに、マネージャまたはマネージャのチェーンをインポートする	インポートされた個人情報（マネージャ情報も含む）	マネージャがシステムにインポートされる

標準操作とカスタム コード操作の混在と照合、または置換も、ディレクトリ統合フレームワークでサポートされています。次の表に示すように、サービス カタログ（Service Catalog）は、1 つのイベントでさまざまな操作の組み合わせをサポートしています。独自にカスタマイズしたコードと、これらのインターフェイスを実装しやすくするよう設計された、サービス カタログ（Service Catalog）の公開 API を使用できます。

カスタム コードの設計および開発エンジニアは、ディレクトリ統合フレームワーク、公開 API、およびカスタム コード インターフェイスについて理解することが重要です。これらについては、この章で詳しく説明します。

次の表 8-16 は、カスタム コード操作のメソッド、イベント、および操作タイプ間の関係を表しています。次の表 8-16 に記載されていない組み合わせはサポートされていません。

表 8-16　カスタム コード操作

イベント	処理のタイプ	インターフェイス	方法
ログイン（Login）	SSO	ISignOn	getCredentials
	EUA	ISignOn	authenticate
	Import Person	ISignOn	importPerson
	Import Manager	ISignOn	importManager
	Custom Code	ISignOn	performCustom
Person Search for: • 代理オーダー（Order On Behalf） • 承認の委任（Authorization Delegate） • サービス フォーム（Service Form）	Person Search	IPersonSearch	getCredentials
	Import Person	IPersonSearch	importPerson
	Import Manager	IPersonSearch	importManager
	Custom Code	IPersonSearch	performCustom

カスタム コード操作インターフェイス

イベント内で設定されている操作のカスタム実装を提供する場合、「カスタム コード操作インターフェイス」を実装する必要があります。

カスタム コード操作インターフェイスは、特定の操作がトリガーされたときに呼び出されるコールバック メソッドを定義します。どのメソッドが呼び出されるかは、操作で選択された操作タイプによって決まります。詳細については、カスタム コード操作のメソッド、イベント、およびタイプの表を参照してください。カスタム コードの操作インターフェイスで定義されるすべてのメソッドは、同じパターンに従っています。

パラメータ

以下のリストで、「**」は次のいずれかの操作タイプで置き換える必要があります。

- IEUISignon
- IEUIPersonSearch

1. **OperationDTO：このオブジェクトには、Administration モジュールの [Directories] タブで操作をどのように設定したか、についての情報が含まれています。これには、マッピングおよびデータ ソースの情報が含まれます。
2. **OperationContext：Context オブジェクトを使用して、メソッドの呼び出しで情報を共有します。ディレクトリ統合フレームワークは、1 つのコンテキスト オブジェクトに情報を格納しますが、これは同じ HttpServletRequest の呼び出しのときに他のコンテキスト オブジェクトで使用することができます。
   a. setLocalContextObject および getLocalContextObject を使用すると、結果の一部にはならないカスタム情報を設定できます。
   b. get**Result を使用すると結果オブジェクトが取得されます。結果オブジェクトには、イベント要求全体で発生したことに関するすべての情報が含まれています。結果には、製品化されたインポートでサポートされている情報が含まれています。LocalContext オブジェクトを使用して、製品化された操作の実装で予期しなかったオブジェクトを格納します。

3. Request：HttpServletRequest です。
4. **ImportAPI：このオブジェクトを使用して個人をインポートします。詳細については、Javadoc を参照してください。
5. **LDAPAPI：この API を使用して LDAP クエリーを作成します。詳細については、Javadoc を参照してください。

リターン

**Result。カスタム タスクを実行すると、API は、有効な戻りタイプに結果を挿入して返します。関連するプロパティを更新した後、OperationContext から取得される、同じ結果オブジェクトが返されます。結果オブジェクトの新しいインスタンスが返されると、予期しない動作が生じることがあります。

次の表 8-17 は、予想される入力または戻りと、各コールバック メソッドのパラメータにあるオブジェクトの関係を示しています。

表 8-17　カスタム コード コールバック メソッドの入力

情報	オブジェクト/プロパティ
HttpServletRequest	要求
ログイン名（Login Name）	• IEUISignOnOperationContext .IEUISignOnOperationResult.ssoLoginId • IEUIPersonSearchOperationContext .IEUIPersonSearchOperationResult.ssoLoginId
First Name と LastName	• First Name：IEUIPersonSearchOperationContext. firstNameSearchString • Last Name：IEUIPersonSearchOperationContext. lastNameSearchString
個人のリスト	IEUIPersonSearchOperationContext .EUIPersonSearchOperationResult.SearchPersonList.SearchPersonList is a collection with all elements of type IExtPersonDTO
インポートされた個人情報	• IEUISignOnOperationContext.IEUISignOnOperationResult.ImportedPersonExtDTO • IEUIPersonSearchOperationContext.EUIPersonSearchOperationResult.ImportedPersonExtDTO
マネージャ ID（Manager Id）	IEUIPersonSearchOperationContext.EUIPersonSearchOperationResult.ImportedPersonExtDTO.PersonDTO.managerID

実装クラスをコンパイルするには、すべてのメソッドを実装する必要があります。制限されている操作タイプのみをカスタマイズする場合、その操作タイプに関連しないメソッドの、空の実装を提供する必要があります。

たとえば、カスタマイズされた SSO のみを対象にする場合は、getCredentials メソッドの完全な実装を提供します。その他のすべてのメソッドについては、ヌルを返します。

システムはインターフェイスのインスタンスをプールできます。また、複数のスレッドから同時にアクセスすることができます。したがって、インスタンスはステートレスにしておくことを推奨します。

カスタム コードの操作インターフェイスには、次の 2 つのタイプがあります。

- **ISignOn** はログインのカスタマイズで使用されます。
- **IPersonSearch** は [個人検索（Person Lookup）] ダイアログ ボックスのカスタマイズで使用されます。

Login イベント カスタム コード インターフェイス：ISignOn

これは、ログイン イベント SSO、EUA、Import Person、Import Manager、およびカスタム コード操作をカスタマイズするために、カスタム コードが実装する必要のあるインターフェイスです。

SSO 操作のカスタマイズ

SSO カスタム コード操作の主な目的は、Remote NTML/IWA のタイプが Sign-On の場合に、Sign-On、または CGI ヘッダー（CGI 変数 REMOTE_USER）からログイン名を取得し、返すことです。

表 8-16 に概要を示すように、ISignOn インターフェイスを実装する Java クラスを提供する必要があります。このインターフェイスでは、getCredentials メソッドの完全な実装を提供してください。詳細な仕様については、ISignOn インターフェイスのマニュアルを参照してください。

以下に、getCredentials メソッドを実装するためのガイドラインを示します。これらのすべてのガイドラインを実装する必要はありません。以下の概説では取り上げていませんが、カスタマイズによっては追加の要件が存在する場合があります。

- IEUISignOnOperationContext から IEUISignOnOperationResult を取得します。これは、このインターフェイスから必ず返されるオブジェクトです。
- パラメータ要求を使用して処理し、個人のログイン名が得られるようにします。
- in-product ディレクトリのルックアップ機能を使用した場合は、IEUISignOnOperationResult.setSsoLoginId(<login id>) をコールすると LoginId が返されます。
- IEUISignOnOperationResult.setSsoRedirectUrl("<any url or error page>") をコールします。これは、SSO の障害時にユーザをリダイレクトするために使用します。

IEUIEventSSOOperationDTO.getEventSsoDTO() で取得された SSO 操作のオプションはヌルになる可能性があります。カスタム コード操作に対しては、Administration モジュールが SSO オプションを受け入れないからです。

Login イベントに対する EUA 操作のカスタマイズ

EUA カスタム コード操作の主な目的は、外部システムに対してユーザを認証することです。

表 8-16 に概要を示すように、ISignOn インターフェイスを実装する Java クラスを提供する必要があります。このインターフェイスでは、authenticate メソッドの完全な実装を提供してください。詳細な仕様については、ISignOn インターフェイスのマニュアルを参照してください。

以下に、EUA 操作を実装するためのガイドラインを示します。これらのすべてのガイドラインを実装する必要はありません。以下では取り上げていませんが、カスタマイズによっては追加の要件が存在する場合があります。

- IEUISignOnOperationContext から IEUISignOnOperationResult を取得します。これは、このインターフェイスから必ず返されるオブジェクトです。
- IEUIEventEUAOperationDTO オブジェクトの EUIDatasourceDTO オブジェクトには、この操作に対して Administration モジュールで設定された Datasource へのインターフェイスが含まれています。
- EUIUtil の LDAPConfigInfo オブジェクトに値を挿入し、EUIDatasourceDT に渡します。これは、LDAP Server に接続情報を使用して LDAP API をコールするために必要です。
- IEUISignOnOperationResult.getSsoLoginId() をコールして Login Name を取得します。
- BindDN を形成し、setBindDN() をコールして LDAPConfigInfo に設定します。

- IEUISignOnOperationResult.getEuaPassword() をコールして、[ログイン（Login）] ページでユーザが入力したパスワードを取得します。この関数は、BASE64 でエンコードされたパスワードを返します。この文字列をプレーンなパスワードに変換するには、プログラムを使用して次の手順に従います。
  - 文字列から、プレフィックス「!@^_」およびサフィックス「_^@!」を削除します。
  - 上記から得られたテキストに対し、Java の Base64 復号化方式を使用します。
- setBindPassword() をコールして、それを LDAPConfigInfo に設定します。
- LDAPConfigInfo オブジェクト ILDAPApi.authenticate() API を渡して、Directory Server に対してユーザを認証します。
- ユーザが認証されたら、IEUISignOnOperationResult.setEuaAuthenticated(true) をコールします。
- ユーザの認証が失敗したか、何らかの例外が発生した場合は、IEUISignOnOperationResult.setEuaAuthenticated(false) をコールします。

IEUIEventEUAOperationDTO.getEventEuaDTO() で取得した EUA 操作オプションは空になります。カスタム コードの操作に対しては、Administration モジュールが EUA オプションを受け入れないからです。

Login イベントに対する Import Person 操作のカスタマイズ

[個人のインポート（Import Person）] 操作の主な目的は、外部システム（ディレクトリ サーバや外部データベースなど）からサービス カタログ（Service Catalog）アプリケーションにユーザをインポートまたは更新することです。

表 8-16 に概要を示すように、ISignOn インターフェイスを実装する Java クラスを提供する必要があります。このインターフェイスでは、importPerson メソッドの完全な実装を提供してください。詳細な仕様については、ISignOn インターフェイスのマニュアルを参照してください。

以下に、Import Person 操作を実装するためのガイドラインを示します。これらのすべてのガイドラインを実装する必要はありません。以下では取り上げていませんが、カスタマイズによっては追加の要件が存在する場合があります。

- IEUISignOnOperationContext から IEUISignOnOperationResult を取得します。これは、このインターフェイスから必ず返されるオブジェクトです。
- IEUIEventImportPersonOperationDTO オブジェクトの EUIDatasourceDTO オブジェクトには、この操作に対して Administration モジュールで設定されたデータソースへのインターフェイスが含まれています。
- IEUIEventImportPersonOperationDTO オブジェクトの EUIDataMappingDTO オブジェクトには、この操作に対して Administration モジュールで設定されたマッピングへのインターフェイスが含まれています。
- IEUISignOnOperationResult.getSsoLoginId() メソッドから取得したログイン名を使用して、LDAP サーバまたは外部データベースから外部システム上のユーザに対して問い合わせを行い、個人のプロファイルに関するすべての情報（組織単位、グループ、ロールなど）を収集します。
- ISignOnImportPersonAPI.getPersonByLoginName(<Login Id>) をコールして、ユーザがすでにサービス カタログ（Service Catalog）データベースに存在しているかどうかチェックします。すでに存在している場合、このメソッドは IPersonDTO オブジェクトを返します。存在していない場合、メソッドは signOnImportPersonAPIException をスローします。
- 個人が見つからない場合は、個人のインポートを準備するために、PersonFactory.createPersonDTO() メソッドによって IPersonDTO オブジェクトを作成します。

- 外部システムからフェッチしたデータから、PersonFactory を使用してこれらの DTO を作成して値を挿入します。IPersonDTO、ILoginInfo、IContactDTO、IAddressDTO、および IPersonExtensionDTO についても同様にします。
- ISignOnImportPersonAPI.beginTransaction() をコールしてデータベース トランザクションを開始します。
- ISignOnImportPersonAPI.getOrgUnitByName(<OU Name>) をコールして、組織単位（OU）が存在するかどうかチェックします。存在する場合、このメソッドは IOrganizationalUnitDTO オブジェクトを返します。組織単位が存在しない場合、メソッドは signOnImportPersonAPIException をスローします。
- OU が存在しない場合は、ISignOnImportPersonAPI.createOrgUnit(<IOrganizationalUnitDTO>) をコールして作成することができます。
- ユーザがすでに存在する場合、ISignOnImportPersonAPI.updatePerson(<IPersonDTO>) をコールします。これにより、個人の基本プロファイル、ログイン情報、プリファレンス、ホーム OU、および拡張が更新されます。
- ユーザがすでに存在する場合は、ISignOnImportPersonAPI.linkAddresses(<IAddressDTO collection>) および ISignOnImportPersonAPI.linkContact(<IContactDTO> をコールして、住所/ロケーションおよび連絡先をリンクまたは更新します。
- 個人が外部システムの 1 つ以上のグループに関連付けられている場合は、ISignOnImportPersonAPI.getGroupByName (<ou name>) をコールして、最初に、存在するすべてのグループを取得します。関連付けられていない場合は、ISignOnImportPersonAPI.createGroup(<IOrganizationalUnitDTO>) をコールして新しいグループをすべて作成します。
- 個人が新規の場合は、ISignOnImportPersonAPI.linkPersonToOrgUnit() および ISignOnImportPersonAPI.linkPersonToGroup() をコールして、OU およびグループのすべてのリストをユーザにリンクします。
- 個人がすでに存在する場合は、ISignOnImportPersonAPI.unlinkPersonToOrgUnit() および ISignOnImportPersonAPI.unlinkPersonToGroup() をコールして、ユーザからすべての OU およびグループをリンク解除することができます。
- OU の既存の関係（個人のホーム OU やグループなど）を見つけるには、ISignOnImportPersonAPI.getOrgUnitsForPerson() および ISignOnImportPersonAPI.getGroupsForPerson() methods をコールします。
- ロールへの既存の関係を見つけるには、ISignOnImportPersonAPI.getRolesForPerson() method をコールします。
- インポートされた個人をロールに関連付ける必要がある場合は、最初に ISignOnImportPersonAPI.getRBACRoleByLogicName(<roleLogicName>) を使用してロールを取得します。
- ISignOnImportPersonAPI.linkPersonToRole() または ISignOnImportPersonAPI.unlinkPersonToRole() をコールして、個人にロールをリンクまたはリンク解除します。
- 個人のインポートまたは更新が正常に終了したら、IEUISignOnOperationResult にフラグ ImportPersonDone = true を設定します。
- インポートまたは更新が正常に終了したら、PersonFactory.createExtUserDTO() で IExtUserDTO のオブジェクトを作成し、IPersonDTO および HomeOUDTO (IOrganizationalUnitDTO) を IExtUserDTO に設定し、IEUISignOnOperationResult.setImportedPersonExtDTO(<IExtUserDTO>) をコールして、インポートした個人の IExtUserDTO を返します。
- インポートまたは更新操作が失敗したら、IEUISignOnOperationResult にフラグ ImportPersonDone = false を設定します。

- ISignOnImportPersonAPI.commitTransaction() をコールしてデータベース トランザクションを終了またはコミットします。
- トランザクションが失敗した場合は、ISignOnImportPersonAPI.rollbackTransaction() をコールして `exception` ブロックでトランザクションをロールバックし、ISignOnImportPersonAPI.releaseTransaction() をコールして `finally` ブロックでトランザクションをリリースします。

IEUIEventImportPersonOperationDTO.getImportPersonDTO() メソッドによる Import Person 操作のオプションは空です。カスタム コードの操作に対しては、Administration モジュールが Import Person オプションを受け入れないからです

Login イベントに対する Import Manager 操作のカスタマイズ

[マネージャのインポート（Import Manager）] 操作の主な目的は、ディレクトリ サーバなどの外部システムから、個人の上司チェーンをサービス カタログ（Service Catalog）にインポートまたは更新することです。

表 8-16 に概要を示すように、ISignOn インターフェイスを実装する Java クラスを提供する必要があります。このインターフェイスでは、importPerson メソッドの完全な実装を提供してください。詳細な仕様については、ISignOn インターフェイスのマニュアルを参照してください。

以下に、Import Manager 操作のガイドラインを示します。

- IEUISignOnOperationContext から IEUISignOnOperationResult を取得します。これは、このインターフェイスから必ず返されるオブジェクトです。
- IEUISignOnOperationResult から、インポートまたは更新されたユーザの ImportedPersonExtDTO を取得します。
- IEUISignOnOperationResult.getImportedPersonExtDTO() method メソッドでインポートされた個人を取得します。IExtUserDTO オブジェクトが返され、それから PersonDTO オブジェクトを取得します。
- 外部システムから、必要に応じてすべてのマネージャをインポートし、前述の Import Person の例で説明しているのと同じ方法で、各マネージャを作成または更新します。
- personDTO は、インポートされたマネージャの IPersonDTO への参照であり、managerDTO は、マネージャのインポート後に返された IPersonDTO への参照であることを前提に、マネージャを個人にリンクします。
- personDTO.setManagerId(managerDTO.getId() を使用して、personDTO に対するマネージャの関係を設定します。
- [個人のインポート（Import Person）] 操作に説明のあるいずれかのメカニズムを使用して personDTO を保存し、関係を保存します。

マネージャ チェーンをインポートする場合は、個人よりも先に最上位のマネージャをインポートすることを推奨します。これにより、personDTO が個人のマネージャとのリンクを更新するための不要な更新が防止されます。

IEUIEventImportManagerOperationDTO. getImportManagerDTO() で取得した Import Manager 操作オプションは空になります。カスタム コードの操作に対しては、Administration モジュールが Import Manager オプションを受け入れないからです。

Login イベントに対するカスタム操作のカスタマイズ

カスタム コード操作の主な目的は、アプリケーションの他のどの場所にも示されていない、必要なカスタム操作を実行することです。

以下に、カスタム コード操作のガイドラインを示します。

- IEUISignOnOperationContext から IEUISignOnOperationResult を取得します。これは、このインターフェイスから必ず返されるオブジェクトです。
- IEUIEventCustomOperationDTO から EUIDatasourceDTO を取得します。このオブジェクトには、この操作の Administration モジュールで設定されたデータソースが含まれています。
- IEUIEventCustomOperationDTO から EUIDataMappingDTO を取得します。このオブジェクトには、この操作の Administration モジュールで設定されたマッピングが含まれています。
- 必要に応じて、すべてのカスタム操作を実行します。
- 前の例に基づいて、IEUISignOnOperationResult には値が適切に挿入されるはずです。

Person Lookup のカスタム コード インターフェイス：IPersonSearch

これは、Person Search イベントの Person Search、Import Person、Import Manager、およびカスタム コード操作をカスタマイズするために、カスタム コードが実装する必要のあるインターフェイスです。

実装クラスは、Administration モジュール > [ディレクトリ（Directories）] タブ > [イベント（Events）] で設定されます。サービス カタログ（Service Catalog）アプリケーション内の次の場所で個人を検索するために設定することができます。

- Person Search for Order On Behalf
- Person Search for Authorization Delegate
- Person Search for Service Form

Person Search 操作のカスタマイズ

Person Search 操作の主な目的は、ディレクトリ サーバなどの外部システムからユーザを検索することです。

カスタム コード操作の表に概要を示すように、IPersonSearch インターフェイスを実装する Java クラスを提供する必要があります。このインターフェイスでは、search メソッドの完全な実装を提供してください。詳細な仕様については、ISignOn インターフェイスのマニュアルを参照してください。

以下に、Person Search 操作のガイドラインを示します。

- IEUISignOnOperationContext から IEUISignOnOperationResult を取得します。これは、このインターフェイスから必ず返されるオブジェクトです。
- カスタム Person Search 操作は、Person Search を使用して設定できるため、Search Event の以前の操作から、検索結果に対して追加したり、操作したりできます。このようにするには、IEUISignOnOperationResult.getSearchPersonList() をコールして、すでに検索結果内に存在する個人のリストを取得します。
- 外部システムのユーザを検索します。ディレクトリ サーバの場合は、インターフェイス ILDAPApi で API メソッドを使用し、外部データベースの場合は、SQL データソースに接続するために ISignOnImportPersonAPI で API を使用します。
- 外部システムで見つかった個人ごとに IExtUserDTO を作成します。

- IExtUserDTO に、IPersonDTO、IOrganizationalUnitDTO（ホーム OU の場合）、および ILoginInfoDTO を挿入します。
- 任意：個人のポップアップ グローバル設定に基づいて、コレクション IContactDTO、IAddressDTO のコレクション、IPersonExtensionDTO にも挿入します。
- ISignOnImportPersonAPI getCustomParam("ShowAllUsersForOrderOnBehalf") を使用して、フラグ「All Users For Order On Behalf」を取得します。
- カスタム コードと標準プラットフォームの動作との整合性を保つため、フラグがオフで、個人に対していずれかの必須属性が指定されていない場合は、そのエントリを削除します。これにより、不完全な個人はポップアップに表示されなくなります。
- カスタム コードと標準プラットフォームの動作との整合性を保つため、フラグがオンで、いずれかの必須属性に対して個人が指定されていない場合は、IExtUserDTO.setResultHasError(true) をコールします。これにより、不完全な個人がポップアップに含まれるようになりますが、オプション ボタンの代わりに赤いアスタリスク「*」が表示されます。星印の付いたユーザは、エンド ユーザが選択することも、インポートすることもできません。
- IEUISignOnOperationResult. setSearchPersonList(<List of all IExtUserDTO>) をコールして、検索されたすべての個人のリストを返します。

IEUIEventPersonSearchOperationDTO.getPersonSearchOperationDTO() メソッドで取得した Person Search 操作のオプションは空です。これは、カスタム コードの操作に対しては、Administration モジュールが Person Search オプションを受け入れないからです。

Person Search イベントに対する Import Person 操作のカスタマイズ

カスタム コード操作の表に概要を示すように、IPersonSearch インターフェイスを実装する Java クラスを提供する必要があります。このインターフェイスでは、importPerson メソッドの完全な実装を提供してください。詳細な仕様については、IPersonSearch インターフェイスのマニュアルを参照してください。

カスタマイズする手順は、Login イベントに対する Import Person 操作のカスタマイズと同様です。

Person Search イベントに対する Import Manager 操作のカスタマイズ

カスタム コード操作に概要を示すように、IPersonSearch インターフェイスを実装する Java クラスを提供する必要があります。このインターフェイスでは、search メソッドの完全な実装を提供してください。詳細な仕様については、IPersonSearch インターフェイスのマニュアルを参照してください。

カスタマイズする手順は、Login イベントに対する Import Manager 操作のカスタマイズと同様です。

Person Search イベントに対するカスタム操作のカスタマイズ

カスタム コード操作に概要を示すように、IPersonSearch インターフェイスを実装する Java クラスを提供する必要があります。このインターフェイスでは、performCustom メソッドの完全な実装を提供してください。詳細な仕様については、IPersonSearch インターフェイスのマニュアルを参照してください。

カスタマイズする手順は、Login イベントに対するカスタム操作のカスタマイズと同様です。

カスタム Java クラス マッピング インターフェイス

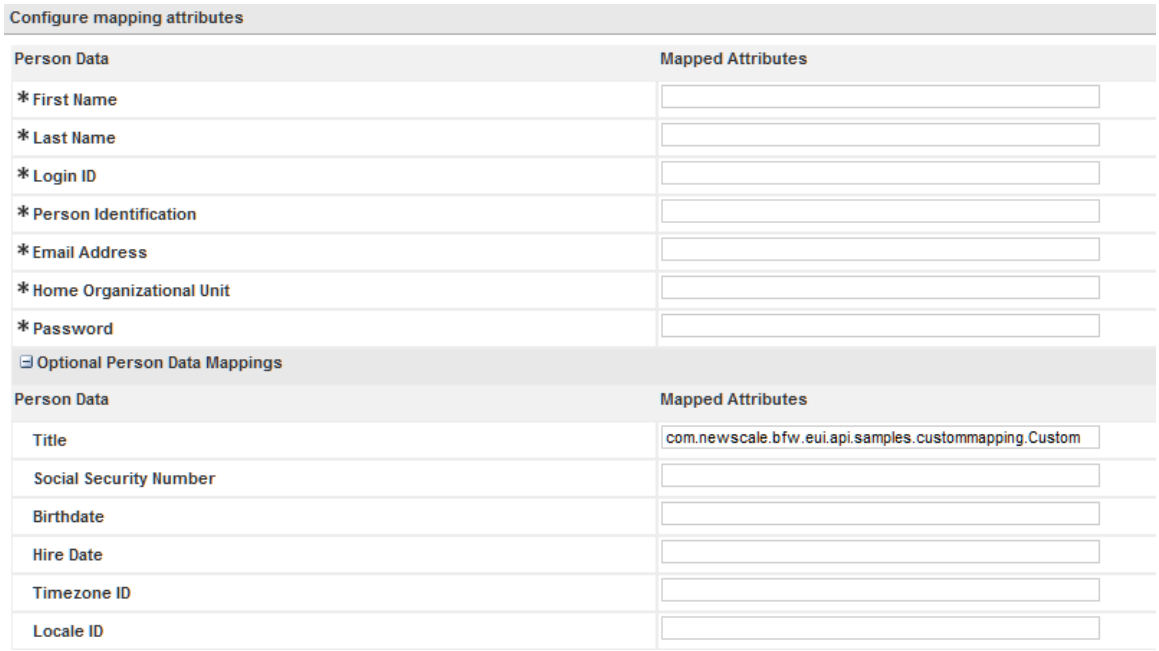
簡易、複合、または正規表現の各属性マッピングでは不十分な場合は、ディレクトリ統合属性マッピングでカスタム Java クラスを使用することができます。

属性マッピングに対するカスタム Java クラス：IEUIAttributeMapping

これは、ディレクトリ属性マッピングをカスタマイズするために、カスタム コードが実装する必要のあるインターフェイスです。カスタム マッピング クラスの主な目的は、ディレクトリ サーバから取得した属性値をカスタマイズすることです。

実装クラスは、Administration モジュール > [Directories] タブ > [Mappings] で設定され、マッピングのすべての属性に対して設定できます。

図 8-16　　属性マッピングに対するカスタム Java クラス



以下に、カスタム Java クラスのマッピング クラスを使用するためのガイドラインを示します。

- マッピング クラスは、ディレクトリから取得された値に適用される、簡単なロジックに対してのみ使用されます。
- パフォーマンス上の理由から、Directory Server API を使用してディレクトリ サーバに対するコールを実行したり、何らかのデータベース操作を実行するためにマッピング クラスを使用することはできません。これらの用途に対しては、Person Search または Login インターフェイスを使用する必要があります。
- マップされた属性に対して単一の値を返すには、IEUIAttributeMapping.getAttributeValue() を実装します。このメソッドは、OU List、Group List、または Role List mapping フィールドに対して実装しないでください。
- マップされた属性に対して複数の値を返すには、IEUIAttributeMapping.getAttributeValueArray() を実装します。このメソッドは、OU List、Group List、および Role List マッピング フィールドに対してのみ実装されます。

ディレクトリ サーバ API

これは、製品に組み込まれているディレクトリ サーバ（LDAP）接続機能と統合するために、シスコが提供する API ラッパーです。

この API が提供する機能は、ディレクトリ サーバに対する認証、およびクエリーのみです。この API は、サービス カタログ（Service Catalog）でサポートされているすべてのディレクトリ サーバをサポートします。

通常、Directory Server API はディレクトリ統合のデータソースおよびマッピング設定から機能し、クエリー用の接続情報、フィルタ、および属性を手作業でコーディングする必要がありません。

一般的に、LDAP API を使用するには、LDAPConfigInfo オブジェクトも必要です。この目的のためには、すべてのデータソースおよびマッピングから EUIUtil.get LDAPConfigInfo() を使用します。

LDAP API の javadoc は、製品パッケージの javadocs フォルダにあります。

ILDAPApi のインスタンスの取得：API 実装

ILDAPApi のインスタンスを作成する必要はありません。両方のカスタム コード API インターフェイス（ISignOn と IPersonSearch）のすべてのメソッド引数で使用できます。

ディレクトリ統合ユーティリティ（EUIUtil）クラス

ディレクトリ統合ユーティリティクラス（EUIUtil）は、Administration モジュールで設定されたデータソースおよびマッピングを、Directory Server API が認証、検索、およびクエリー機能の入力として使用できる形式に変換します。

LDAP 設定情報（LDAPConfigInfo）クラス

LDAPConfigInfo クラスのオブジェクトは、ディレクトリ サーバ API に渡す必要がある、次のすべての設定オプションをカプセル化します。

- 認証情報
- 接続情報
- 属性のクエリー
- サーチ フィルタ

上級ユーザの場合は、何らかの設定を上書きする必要がある場合に、すべての設定に対する getter および setter が LDAPConfigInfo から提供されます。これらのメソッドの詳細については、このクラスに関する Javadoc を参照してください。

API のメイン インターフェイス：ILDAPApi

ILDAPApi は、ディレクトリ サーバ上で次の 2 つの基本操作を提供するメイン インターフェイスです。

- 認証
- Search/Query

ILDAPApi インターフェイスは、サービス カタログ（Service Catalog）全体で LDAP と一貫性を保って対話を行うためのメソッドを提供します。

LDAPEntryBean

ILDAPApi.query(…) メソッドを使用してディレクトリ サーバに対してクエリーまたは検索を行うと、LDAPEntryBean のコレクションとして結果が返されます。

個人のインポート/更新 API

この API を使用すると、個人プロファイルのインポートまたは更新、OU またはグループの作成、OU、グループ、またはロールへの個人のリンクまたはリンク解除を行えます。この API は、個人をインポートするためのトランザクション管理、および SQL データソースへの接続もサポートしています。この API には、CnfParams テーブルから読み込むためのメソッドも含まれています。

個人のインポート/更新 API インターフェイス：ISignOnImportPersonAPI

個人のインポート/更新 API インターフェイスは、次のためのメソッドを提供します。

- PersonID または LoginName によって Person オブジェクトを取得する。Person とログイン情報、プリファレンス、ホーム OU、住所、連絡先、場所、および拡張が返されます。
- ログイン情報、プリファレンス、ホーム OU、住所、連絡先、場所、および拡張を使用して Person を作成する。
- ログイン、プリファレンス、ホーム OU、および拡張を使用して Person を更新する。
- OrganizationalUnitID、Name により OU を取得する。OU のメンバは返されません。
- 特定の Person についてすべての OU を取得する。OU のメンバは返されません。
- OU を作成します。
- Person と OU をリンク/リンク解除する。
- GroupID、Name によって Group を取得する。Group のすべてのメンバは返されません。
- 特定の個人についてすべてのグループを取得する。
- グループを作成します。
- 個人とグループをリンク/リンク解除する。
- 名前によりユーザ定義ロールを取得する。
- システム定義ロールに対する LogicName オブジェクトを取得する。
- LogicName オブジェクトによりシステム定義ロールを取得する。
- 指定された個人のすべてのロールを取得する。
- 個人とロールをリンク/リンク解除する。
- 個人の住所や場所をリンク/更新する。
- 個人の連絡先を追加/更新/削除する。
- Import Person に対するトランザクションの開始、トランザクションのコミット、およびトランザクション リソースのリリースを行う。
- SQL データソースへの接続を取得する。
- SQL データソース接続でトランザクションをロールバックする。
- SQL データソースへの接続を接続プールに戻す。
- CnfParams テーブルからパラメータ値を取得する。

詳細については、Java のマニュアルを参照してください。

SQL データソースに接続するための Java クラスのカスタマイズ

SQL データソースに接続するために Java クラスをカスタマイズするには、次の手順に従います。

手順 1　DatasourceName を渡すことで、ISignOnImportPersonAPI から SQL データソース データベースへの接続を取得します。DatasourceName には、newscale.properties ファイルの「DatasourceJNDIPrefix」プロパティで定義されているように、JNDI プレフィックスが付加されます。

手順 2　JDBC ステートメントを使用してクエリーを実行するには、上記の接続を使用します。

手順 3　**try** ブロックの最後で、接続オブジェクトを直接コミットします。

手順 4　障害または例外が発生したときに接続をロールバックするには、ISignOnImportPersonAPI をコールします。

手順 5　final ブロックで、ステートメントを直接閉じて ISignOnImportPersonAPI をコールし、接続を解放して接続プールに戻します。

ベストプラクティス

カスタム コード Java ファイルのコンパイル

カスタム コードをコンパイルおよび導入する手順を、次に示します。

手順 1　build.xml ファイルの例に示す build.xml ファイルをコピーして、任意のフォルダ（C:\CustomCode など）に貼り付けます。

手順 2　build.xml ファイルを編集して、RequestCenter.war を使用できるフル パスを指すようプロパティ「rcwar.dir」を変更します。

手順 3　build.xml を編集して、servlet-api.jar を使用できるフル パスを指すようプロパティ「javax.servlet.dir」を変更します。これはアプリケーション サーバに特有の設定です。

手順 4　カスタム コード Java ファイル用に、（C:\CustomCode\src などの）サブフォルダを作成します。

手順 5　「com.newscale.SignOnCustomCode」などのパッケージ名を付けてカスタム コードを作成し、SignOnCustomCode.java ファイルを C:\CustomCode\src\com\newscale\SignOnCustomCode.java ディレクトリに配置します。

手順 6　C:\CusomCode フォルダのコマンドラインから「ant」を実行します。

手順 7　ant のビルド ファイルは、「src」サブフォルダの下のすべての java ファイルをコンパイルし、クラス ファイルを「out」サブフォルダに配置します。

手順 8　ant のビルド ファイルは、「RequestCenter.war\WEB-INF\classes」フォルダにもクラス ファイルを導入します。

手順 9　アプリケーション サーバを再起動します。

コーディングのガイドライン

パッケージ名

- パッケージ名は com.newscale.[yourcompanyname].* にすることを推奨します。
- すべての ContextLocalAttributes の格納には、キー名「com.yourcompanyname.*」を使用します。これにより、内部名前空間でのクラッシュが防止されます。

ログ

- メッセージをサーバ ログに記録するには、System.out.println ではなく Logger を使用します。
- デバッグ ログについては、必ずデバッグが有効になっているかどうかを最初に確認します。これはパフォーマンス上、必要です。
- 例外ブロックをコール元に伝搬する前に、必ずエラーを例外ブロックに記録します。

例外処理

- EUIException が取得された場合は、そのままスローして返します。
- その他のすべての例外は EUIException としてラップし、スローして返します。

Administration モジュールでのカスタム コードの設定

カスタム コードの開発、コンパイル、および導入が終了したら、そのコードを使用するように Administration モジュールを設定する必要があります。設定には、いつ（どのイベントで）、どの操作で、どの順序（ステップ）でカスタム コードを呼び出すかを指定します。

ステップ 1: グローバル設定の実行

Administration モジュールの [Settings] タブでこの設定をオンにして、Directory Integration が有効になっていることを確認します。Directory Integration をオンにする方法は、ディレクトリ統合の有効化に説明があります。

ステップ 2: データソースの設定

カスタマイズされているかどうかに関係なく、大半の操作ではデータソースとマッピングが必要です。このため、最初に Directory Administration の 2 つの領域を設定する必要があります。

データソースは LDAP などの外部サーバであり、ユーザのデータが現在格納されています。サービス カタログ（Service Catalog）はデータソースにアクセスする必要があります。データソースが必要ないカスタム操作は SSO だけです。

データソースの設定の詳細については、データソースの定義およびデータソース情報の設定を参照してください。

ステップ 3:属性マッピングの設定

外部データソースを設定したら、サービス カタログ（Service Catalog）で使用できる個人関連のデータを、LDAP ディレクトリ（または他の外部データソース）内のデータにマップする必要があります。これらのマッピングでは、イベントおよび操作のシーケンスで、どこを検索するか、および何を取得するかがサービス カタログ（Service Catalog）に指示されます。

マッピングを設定するには、マッピングの定義およびマッピングの設定のガイドラインと説明に従ってください。

ステップ 4:イベント/カスタマイズ イベントの設定

Login 以外のすべてのイベントに対するシングル サインオン（SSO）および認証の操作のカスタマイズは、不正なアクションとみなされます。これらの操作が必要になるタイミングは他にありません。外部の LDAP サーバから、アプリケーションにユーザがサインインして認証された後は、プロセスを複製する必要がありません。

外部データソースへの接続が必要なすべてのイベントは、ここで設定されます。このガイドに記載されているカスタム コード API を呼び出す場合には、カスタム操作が不正な順序で発生したり、失敗したりしないよう、各イベントに対する操作の順序を考えることが重要です。

手順 1　ナビゲーション ペインで [イベント（Event）] をクリックします。

手順 2　カスタマイズするイベントの [編集（Edit）] をクリックします。

手順 3　イベントが無効になっている場合は、ドロップダウン メニューを使用して [有効（Enabled）] を選択します。

手順 4　[Add step] をクリックして操作を追加します。ここでは、ステップを必要なだけ追加できます。また、各ステップの詳細を設定してから、その次のステップを追加および設定してください。

手順 5　ドロップダウン メニューから [操作（Operation）] を選択します。

- SSO のコードを呼び出すだけの場合は、たとえばメニューから SSO を選択できます。SSO のコードを*カスタマイズ*するには、[カスタムコード（Custom Code）] を選択し、次のステップでカスタマイズする操作を選択します。
- カスタマイズした操作を設定するには、[カスタムコード（Custom Code）] を選択します。

手順 6　ドロップダウン メニューから [マッピング（mapping）] および [データソース（datasource）] を選択します。

手順 7　[追加オプション（Additional Options）] の見出しの下で、[オプション（Options）] をクリックします。

手順 8　そのステップのオプションを設定します。

- カスタム コード操作タイプでは、ドロップダウン メニューを使用して、カスタマイズする操作を選択します。
- Java クラスでは、その操作用の全体のパッケージ名を入力してから、その後ろにクラス名を入力します。たとえば、com.newscale.bfw.eui.api.samples.operations.*CustomCodeTester* のようになります。
- この例では、Java クラス名を*イタリック体*で示してあります。これらは両方がコード自体の中にあり、コピーすることができます。

手順 9　ステップの追加オプションを閉じるには、[Close] をクリックします。

手順 10　必要に応じて、ステップの追加と設定を続けます。

手順 11　イベント用のすべてのステップを保存するには、[Update] をクリックします。

操作タイプとしてのカスタム コードの使用

前述のステップで、操作として [カスタムコード（Custom Code）] を選択し、操作タイプとして [カスタムコード（Custom Code）] を選択した場合は、未定義のカスタム コードをコールしているため、設計する必要があります。

シスコが提供しているカスタム コード テストの例では、Java クラス「performCustom」を使用して、独自のカスタム コードを定義できます。

カスタム コードの導入

すべてのカスタム コードは、サービス カタログ（Service Catalog）インストーラに対するカスタマイズとしてパッケージ化する必要があります。これにより、インストールの更新が必要な場合や、新しいサイトをインストールする場合に、カスタマイズを再適用できます。

カスタム コードをパッケージおよび導入するための方法は、サービス カタログ（Service Catalog）をホストしているアプリケーション サーバによって異なります。カスタム コードの設定の詳細については、『*Cisco Prime Service Catalog 12.0 アダプタ統合ガイド*』および『*Cisco Prime Service Catalog Administration and Operations Guide*』を参照してください。

API のビュー/用途の例

このソリューションは、次の用途に適しています。

- コンテナで管理される SQL データソースから収集されたデータを使用して、個人を検索するイベント クラスを作成する。
- コンテナで管理される SQL データソースから収集されたデータを使用して、個人をインポートするイベント クラスを作成する。
- コンテナで管理される SQL データソースから収集されたデータを使用して、個人を修正するイベント クラスを作成する。
- UI から設定パラメータを受け取ることができるイベント クラスを作成する。この例ではマッピング インターフェイスを使用して、設定パラメータをクラスに渡します。

また、ホーム OU がサービス カタログ（Service Catalog）に存在しない場合に、個人のホーム OU を事業部門として作成します。

（注）　このソリューションでは、データ ソースをアプリケーション サーバ上に設定する必要があります。以降では、EUIPersonSearchSQL クラスの設定および使用法について説明します。

SQL データ ソース

個人プロファイルの必須フィールドのデータが含まれている（または、これらのフィールドの値を得ることができる）SQL テーブルは、データ ソースとして使用できます。次に、この例で使用されるテーブルの定義を示します。

```
CREATE TABLE [psgextusers] (
  [login]     [nvarchar] (100) COLLATE Latin1_General_CI_AI NOT NULL,
  [firstname] [nvarchar] (100) COLLATE Latin1_General_CI_AI NULL,
  [lastname]  [nvarchar] (100) COLLATE Latin1_General_CI_AI NULL,
  [password]  [nvarchar] (100) COLLATE Latin1_General_CI_AI NULL,
  [email]     [nvarchar] (100) COLLATE Latin1_General_CI_AI NULL,
```

```
  [homeOU]    [nvarchar] (100) COLLATE Latin1_General_CI_AI NULL,
  CONSTRAINT [PK_extuser] PRIMARY KEY CLUSTERED
  (
     [login]
  )  ON [PRIMARY]
) ON [PRIMARY]
GO
```

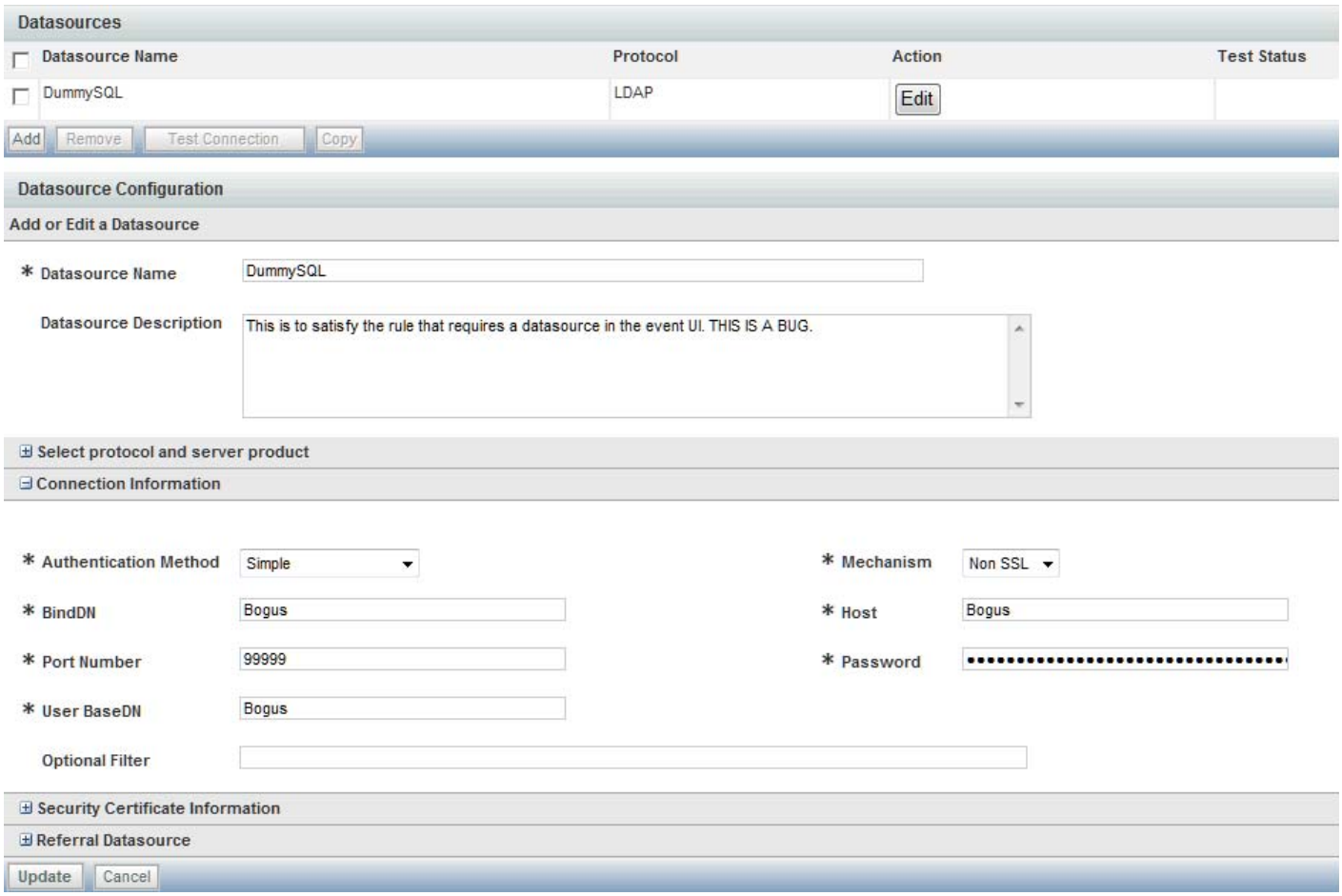
次に、上記のテーブル定義で使用するサンプル データの一部を示します。

```
INSERT INTO [RequestCenter].[dbo].[psgextusers]([login], [firstname], [lastname],
[password], [email], [homeOU])VALUES('Moe', 'Moe', 'Howard', 'Moe', 'moe@stooge.com',
'Nyuk Nyuk Nyuk')
INSERT INTO [RequestCenter].[dbo].[psgextusers]([login], [firstname], [lastname],
[password], [email], [homeOU])VALUES('Larry', 'Larry', 'Fine', 'Larry',
'larry@stooge.com', 'Nyuk Nyuk Nyuk')
INSERT INTO [RequestCenter].[dbo].[psgextusers]([login], [firstname], [lastname],
[password], [email], [homeOU])VALUES('Curly', 'Curly', 'Howard', 'Curly',
'curly@stooge.com', 'Nyuk Nyuk Nyuk')
INSERT INTO [RequestCenter].[dbo].[psgextusers]([login], [firstname], [lastname],
[password], [email], [homeOU])VALUES('Shemp', 'Shemp', 'Howard', 'Shemp',
'shemp@stooge.com', 'Nyuk Nyuk Nyuk')
```

データソースの定義

Directory Integration インターフェイスを使用するには、LDAP データソースを設定しておく必要があります。LDAP は、データソースでサポートされている唯一の UI です。データソースなしでマップを作成できますが、LDAP データソースなしではテストできません。

図 8-17　　データソース設定の例

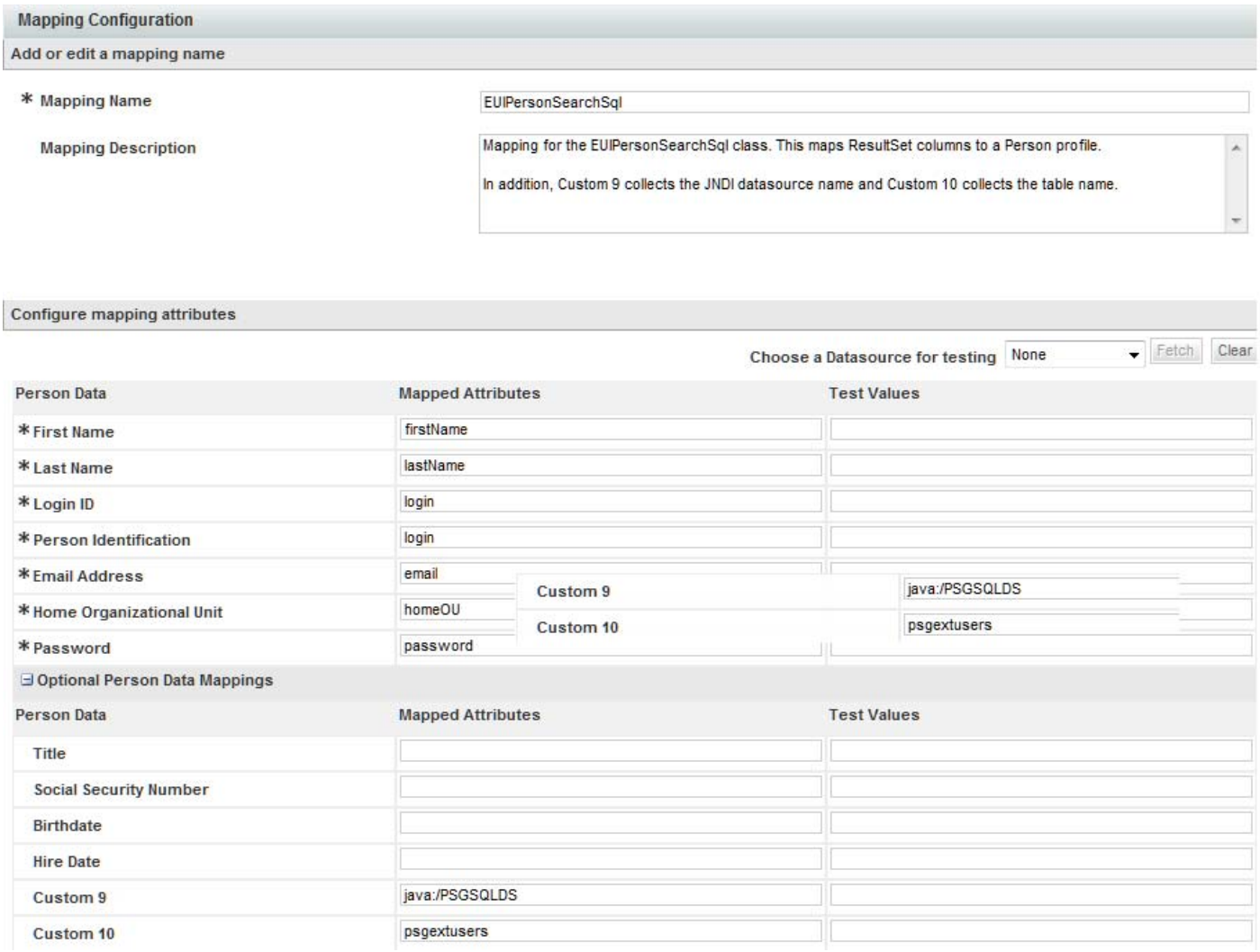


コンテナで管理されるデータソースの設定は、コンテナによって異なります。データソースの設定の詳細については、『*Cisco Prime Service Catalog Installation and Upgrade Guide*』を参照してください。

マッピングの例

EUIPersonSearchSQL クラスに対するマッピングを作成する必要があります。

図 8-18　　マッピングの設定例

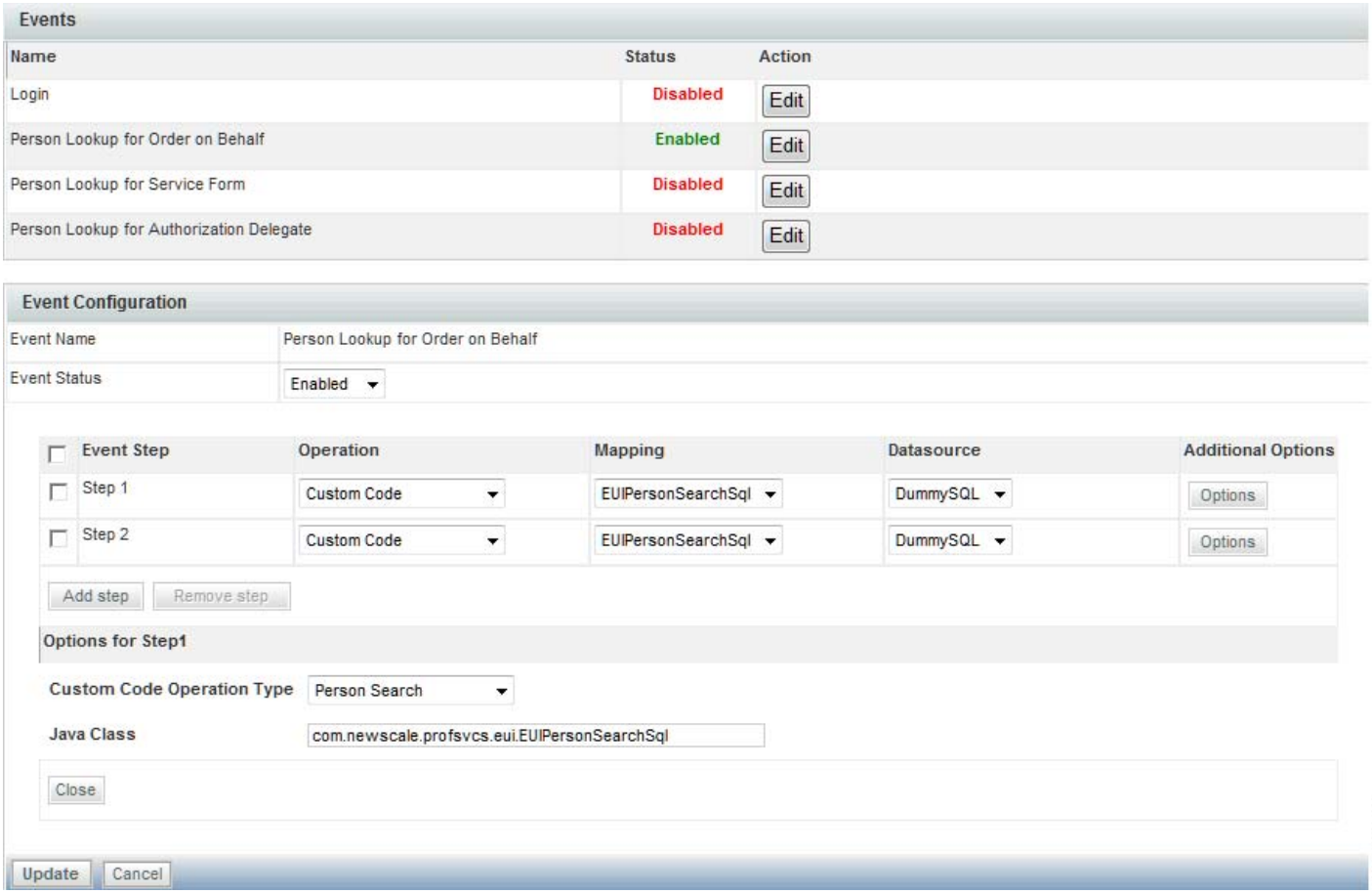


このマッピングには、Custom 9 として JNDI への参照が含まれ、Custom 10 にはテーブル名の参照が含まれています。このようなマッピングを使用すると、「select * from tablename」のような簡単なクエリーを実行し、JDBC のメタデータ機能を使用して、マッピングに基づいてカラムを選択することができます。

イベントの設定例

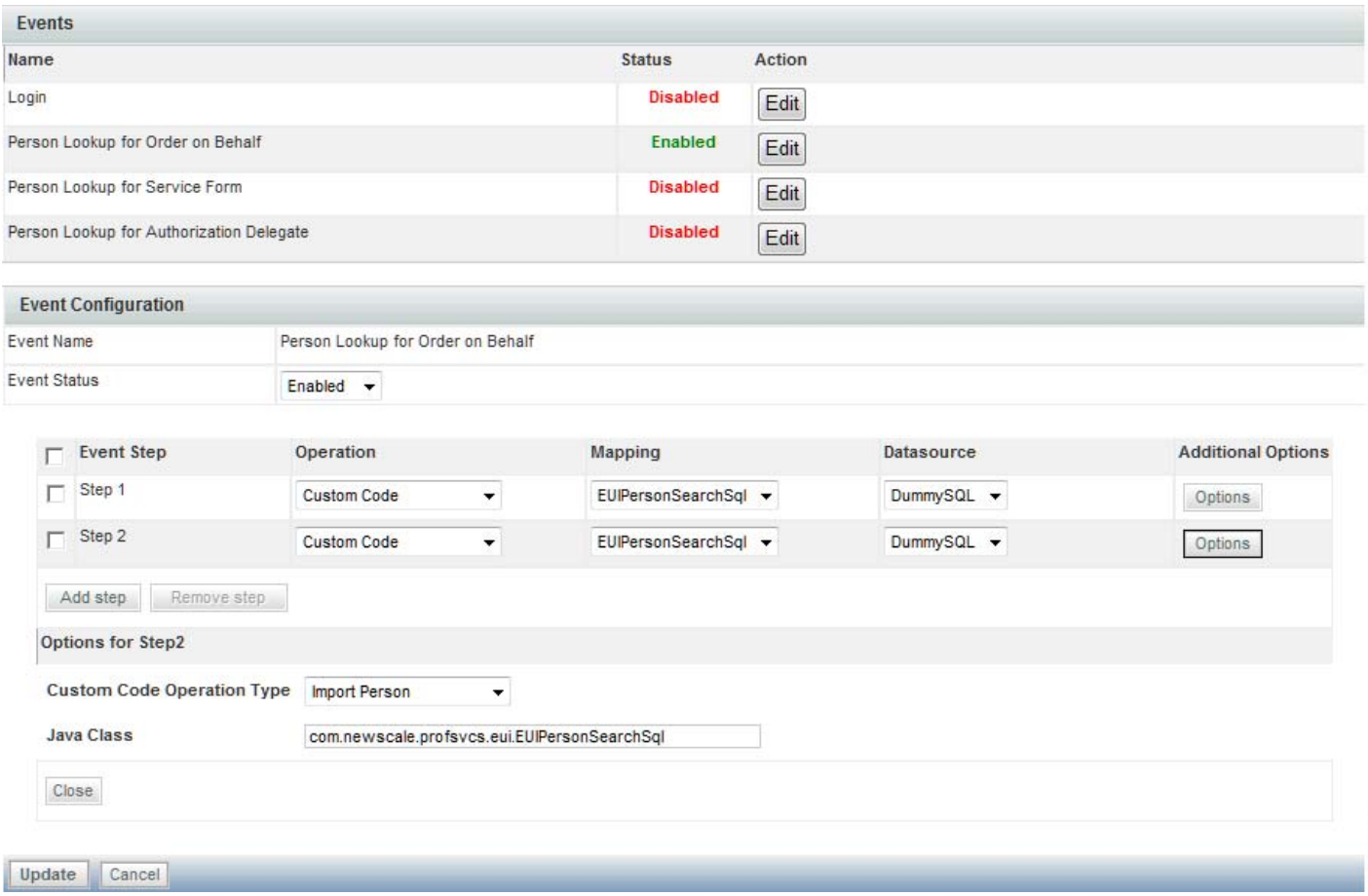
「Person Lookup for Order on Behalf」イベントには 2 つのステップがあり、最初のステップでは「Person Search」操作を実行する必要があります。クラスの名前はマッピングとして与えられます。パッケージの仕様全体は、Java クラスとして与えられます。

図 8-19　　カスタム Person Search 操作



「Person Lookup for Order on Behalf」イベントの 2 番めのステップは、選択した個人のインポート（「Import Person」）です。この設定では同じ Java クラスを使用しますが、カスタム コード操作タイプは異なります。ドロップダウン メニューのカスタム コード操作タイプは、インターフェイス クラスでコールされるメソッドに対応したものとなります。

図 8-20　　イベントのステップ2：カスタム Import Person 操作



SQL ベースの個人検索のサンプル コード

次に、カスタム クラスのソースを示します。

```
package com.newscale.profsvcs.eui;

import com.newscale.api.person.*;
import com.newscale.bfw.eui.EUIException;
import com.newscale.bfw.eui.api.*;
import com.newscale.bfw.ldap.ILDAPApi;
import com.newscale.bfw.logging.ILogUtil;
import com.newscale.bfw.logging.LogUtilFactory;
import com.newscale.comps.extuserintegration.session.*;

import javax.servlet.http.HttpServletRequest;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.*;

/**
 * Person Search to an external SQL datasource
```

```
 *
 * @author Lee Weisz
 * @version $Revision$
 */
public class EUIPersonSearchSql implements IPersonSearch {
  /**
   * Logger instance
   */
  private ILogUtil log = LogUtilFactory.getLogUtil(EUIPersonSearchSql.class);

  /**
   * Implement Person Search Operation and fetch users from an external system
   *
   * @param euiOperationDTO       .
   * @param euiPersonSearchOperationContext
   *
   * @param request               .
   * @param signOnImportPersonAPI
   * @param ldapApi
   * @return .
   * @throws EUIException .
   */
  public IEUIPersonSearchOperationResult search(IEUIEventPersonSearchOperationDTO
euiOperationDTO,
                                                IEUIPersonSearchOperationContext
euiPersonSearchOperationContext,
                                                HttpServletRequest request,
                                                ISignOnImportPersonAPI
signOnImportPersonAPI, ILDAPApi ldapApi)
      throws EUIException {

    log.debug("search: Entering search method...");
    IEUIPersonSearchOperationResult euiOperationResult = euiPersonSearchOperationContext
        .getEUIPersonSearchOperationResult();

    // Check if there is any SearchPerson List already available, if so we
    // can append to the existing List

    // Typically if there is a productized Person Search Operation is
    // configured before the custom code, this list would be populated

    // TODO Why is this an ArrayList? Can't it be a List?
    ArrayList personList = euiOperationResult.getSearchPersonList();

    if (null == personList) {
      personList = new ArrayList();
    }

    // Get the search criteria from the dialog box
    String searchFirstName = euiPersonSearchOperationContext.getFirstNameSearchString();
    String searchLastName = euiPersonSearchOperationContext.getLastNameSearchString();

    log.debug("search: Looking for " + searchFirstName + " " + searchLastName);

    EUIDataMappingDTO dataMappingDTO = euiOperationDTO.getEuiMappingDTO();
    Map attributeMap = dataMappingDTO.getAllAttributeMap();

    // What's in this map?
    if (log.isDebugEnabled()) {
      Set ks = attributeMap.keySet();
      for (Iterator it = ks.iterator(); it.hasNext();) {
        Object key = it.next();
        log.debug("search: " + key + " is " + attributeMap.get(key));
      }
```

```
    }

    // Use the map to map the columns to Person fields
    String firstNameColumn = (String)
attributeMap.get(EUIAPIConstants.EUIMAPFIELDTYPE.ATTR_FIRSTNAME);
    String lastNameColumn  = (String)
attributeMap.get(EUIAPIConstants.EUIMAPFIELDTYPE.ATTR_LASTNAME);
    String loginColumn     = (String)
attributeMap.get(EUIAPIConstants.EUIMAPFIELDTYPE.ATTR_LOGINID);

    // Use the custom9 mapping to hold the datasource value and custom10 to
    // hold the tablename. Since we control the import as well, it won't show up
    // in the imported Person's profile
    String ds = (String) attributeMap.get("custom9");
    String sourceTable = (String) attributeMap.get("custom10");


    StringBuffer searchSQL;
    searchSQL = new StringBuffer().append("select ")
        .append(firstNameColumn).append(", ")
        .append(lastNameColumn).append(", ")
        .append(loginColumn).append(" from ")
        .append(sourceTable);

    if (searchFirstName != null && searchFirstName.trim().length() > 0 ||
        searchLastName != null && searchLastName.trim().length() > 0) {
      searchSQL.append(" where ");

      if (searchFirstName != null && searchFirstName.trim().length() > 0) {
        searchSQL.append(firstNameColumn).append(" like
'").append(searchFirstName.trim()).append("%'");
      }

      if (searchFirstName != null && searchFirstName.trim().length() > 0 &&
          searchLastName != null && searchLastName.trim().length() > 0) {
        searchSQL.append(" and ");
      }

      if (searchLastName != null && searchLastName.trim().length() > 0) {
        searchSQL.append(lastNameColumn).append(" like
'").append(searchLastName.trim()).append("%'");
      }
    }

    log.debug("search: " + searchSQL.toString());

    Connection conn = null;
    Statement s = null;

    // get a connection to the external db
    try {
      conn = signOnImportPersonAPI.getExternalDBConnection(ds);

      s = conn.createStatement();
      ResultSet rs = s.executeQuery(searchSQL.toString());

      while (rs.next()) {
        String fname = rs.getString(firstNameColumn);
        String lname = rs.getString(lastNameColumn);
        String login = rs.getString(loginColumn);

        IExtUserDTO extUserDTO = PersonFactory.createExtUserDTO();
        IPersonDTO personDTO = PersonFactory.createPersonDTO();
```

```
            personDTO.setFirstName(fname);
            personDTO.setLastName(lname);
            personDTO.setPersonIdentification(login);

            // Make the IPersonDTO into an IExtPersonDTO
            extUserDTO.setPersonDTO(personDTO);
            // Add IExtUserDTO to the collection of searched persons
            personList.add(extUserDTO);
          }
        } catch (SQLException e) {
          log.error("search: " + searchSQL.toString(), e);
        } catch (SignOnImportPersonAPIException e) {
          log.error("search: Cannot get a connection to " + ds, e);
        } finally {
          try {
            s.close();
          } catch (SQLException e) {
            e.printStackTrace();
          }
          try {
            conn.close();
          } catch (SQLException e) {
            e.printStackTrace();
          }
        }

        // Set the list of Persons Searched into the Result to be returned
        euiOperationResult.setSearchPersonList(personList);

        log.debug("search: Leaving search method...");
        return euiOperationResult;
      }

      /**
       * Implement the Import Person Operation to Import a user from External
       * system
       *
       * @param euiOperationDTO       .
       * @param euiPersonSearchOperationContext
       *                              .
       * @param request               .
       * @param signOnImportPersonAPI
       * @param ldapApi
       * @return .
       * @throws EUIException .
       */
      public IEUIPersonSearchOperationResult importPerson(IEUIEventImportPersonOperationDTO
euiOperationDTO,
                                                          IEUIPersonSearchOperationContext
euiPersonSearchOperationContext,
                                                          HttpServletRequest request,
                                                          ISignOnImportPersonAPI
signOnImportPersonAPI, ILDAPApi ldapApi)
          throws EUIException {

        log.debug("importPerson: Entering importPerson method...");

        /* Potentially useful stuff on the request...
          Name : isOOB          Value : true/false
          Name : customerid     Value : personDTO.setPersonIdentification() from search
          Name : customerId     Value : personDTO.setPersonIdentification() from search
          Name : lDAPCustomerId Value : personDTO.setPersonIdentification() from search
        */
```

```
    // What's on this request?
    if (log.isDebugEnabled()) {
      log.debug("importPerson: Parameters collected from the search window...");
      Enumeration paramNames = request.getParameterNames();

      if (paramNames.hasMoreElements()) {
        while (paramNames.hasMoreElements()) {
          String paramName = (String) paramNames.nextElement();
          String paramValues[] = request.getParameterValues(paramName);
          if (paramValues != null) {
            log.debug("importPerson: Name : " + paramName);
            for (int i = 0; i < paramValues.length; i++) {
              log.debug("importPerson: Value : " + paramValues[i]);
            }
          }
        }
      }
    }

    boolean refreshPerson = true;
    String login = request.getParameter("customerId");

    // Defaults
    String homeOU    = "";
    String firstName = "";
    String lastName  = "";
    String email     = "";
    String password  = "password";

    // Get the UI mapping
    EUIDataMappingDTO dataMappingDTO = euiOperationDTO.getEuiMappingDTO();
    Map attributeMap = dataMappingDTO.getAllAttributeMap();

    // Use the map to map the columns to Person fields
    String firstNameColumn = (String)
attributeMap.get(EUIAPIConstants.EUIMAPFIELDTYPE.ATTR_FIRSTNAME);
    String lastNameColumn  = (String)
attributeMap.get(EUIAPIConstants.EUIMAPFIELDTYPE.ATTR_LASTNAME);
    String loginColumn     = (String)
attributeMap.get(EUIAPIConstants.EUIMAPFIELDTYPE.ATTR_LOGINID);
    String passwordColumn  = (String)
attributeMap.get(EUIAPIConstants.EUIMAPFIELDTYPE.ATTR_PASSWORD);
    String emailColumn     = (String)
attributeMap.get(EUIAPIConstants.EUIMAPFIELDTYPE.ATTR_EMAILADDRESS);
    String homeOUColumn    = (String)
attributeMap.get(EUIAPIConstants.EUIMAPFIELDTYPE.ATTR_HOMEORGANIZATIONALUNIT);

    // Use the custom9 mapping to hold the datasource value and custom10 to
    // hold the tablename. Since we control the import as well, it won't show up
    // in the imported Person's profile unless we screw up somehow and put it there...
    String ds = (String) attributeMap.get("custom9");
    String sourceTable = (String) attributeMap.get("custom10");

    StringBuffer importSQL;
    importSQL = new StringBuffer().append("select ")
        .append(firstNameColumn).append(", ")
        .append(lastNameColumn).append(", ")
        .append(loginColumn).append(", ")
        .append(passwordColumn).append(", ")
        .append(emailColumn).append(", ")
        .append(homeOUColumn)
        .append(" from ").append(sourceTable).append(" where ")
        .append(loginColumn).append("='").append(login).append("'");
```

```
    log.debug("import: " + importSQL.toString());

    Connection conn = null;
    Statement s = null;

    try {
      // get a connection to the external db
      conn = signOnImportPersonAPI.getExternalDBConnection(ds);
      s = conn.createStatement();
      ResultSet rs = s.executeQuery(importSQL.toString());

      while (rs.next()) {
        homeOU = rs.getString(homeOUColumn);
        firstName = rs.getString(firstNameColumn);
        lastName = rs.getString(lastNameColumn);
        email = rs.getString(emailColumn);
        password = rs.getString(passwordColumn);
      }
    } catch (SQLException e) {
      log.error("import: " + importSQL.toString(), e);
    } catch (SignOnImportPersonAPIException e) {
      log.error("import: Cannot get a connection to " + ds, e);
    } finally {
      try {
        s.close();
      } catch (SQLException e) {
        log.error("import: ", e);
      }
      try {
        conn.close();
      } catch (SQLException e) {
        log.error("import: ", e);
      }
    }

    log.debug("import : Got " + login + "," + firstName + "," + lastName + "," + email +
"," + password + "," + homeOU);

    IPersonDTO personDTO = PersonFactory.createPersonDTO();
    try {
      // Get or Create the Person
      // This API throws an exception if the Person is not found in Service Catalog
      try {
        personDTO = signOnImportPersonAPI.getPersonByLoginName(login);
        log.info("importPerson: " + login + " exists in Request Center");
      } catch (SignOnImportPersonAPIException impEx) {
        log.info("importPerson: Creating new Person for " + login);
        refreshPerson = false;
        personDTO.setLogin(login);
      }

      // Get or Create the Home OU that the Person should be associated with
      // This API throws an exception if the OU is not found in Service Catalog
      IOrganizationalUnitDTO homeOUDTO;
      try {
        homeOUDTO = signOnImportPersonAPI.getOrgUnitByName(homeOU);
        log.info("importPerson: " + homeOU + " exists in Request Center");
      } catch (SignOnImportPersonAPIException impEx) {
        log.info("importPerson: Creating new OU " + homeOU + " for " + login);
        homeOUDTO = PersonFactory.createOrganizationalUnitDTO();
        homeOUDTO.setName(homeOU);
        homeOUDTO.setBillable(false);
        homeOUDTO.setOrganizationalUnitTypeId(2); // business unit.
        homeOUDTO.setRecordStateId(1); // active
```

```
        homeOUDTO.setLocaleId(EUIAPIConstants.LOCALEID.USEN);
        try {
          homeOUDTO = signOnImportPersonAPI.createOrgUnit(homeOUDTO);
        } catch (SignOnImportPersonAPIException crEx) {
          log.error("importPerson: Can't create " + homeOU + " for " + login);
          throw crEx;
        }
      }
      personDTO.setHomeOrganizationalUnitId(homeOUDTO.getId());

      // Populate the Login Object...
      // Modify the login information only if this is a new Person
      if (!refreshPerson) {
        ILoginInfoDTO loginInfoDTO = PersonFactory.createLoginInfoDTO();

        loginInfoDTO.setLoginname(personDTO.getLogin());
        loginInfoDTO.setPrivateKey(personDTO.getLogin());
        // Set the un-encrypted password
        loginInfoDTO.setPassword(password);

        // Set ILoginInfoDTO to IPersonDTO
        personDTO.setILoginInfoDTO(loginInfoDTO);
      }

      // Populate the rest of the essential fields
// Presumably, any expression on the mapping will have already been executed
// and the result is what's returned in the personDTO
      personDTO.setFirstName(firstName);
      personDTO.setLastName(lastName);
      personDTO.setEmail(email);

      // Set the active status
      // TODO These methods are bogus...
//    personDTO.setIsInactive(false);
//    personDTO.setIsActive(true);
      // TODO What do these numbers mean? Is there a constants library to convert these
codes into something meaningful?
      personDTO.setRecordStateId(1);

      // Upsert the Person
      signOnImportPersonAPI.beginTransaction();
      if (refreshPerson) {
        // Update the existing Person
        // This method updates only Basic Info, LoginInfo, Preferences, Home OU and Person
Extension
        signOnImportPersonAPI.updatePerson(personDTO);
      } else {
        // Create the Person
        // This creates a Person with Basic Info, LoginInfo, Preferences, Home OU and
Person Extension
        personDTO = signOnImportPersonAPI.createPerson(personDTO);
        // From here on out it's a refresh
        refreshPerson = true;
      }
      signOnImportPersonAPI.commitTransaction();
    } catch (Exception e) {
      log.error("importPerson: Exception during Import Person", e);
      try {
        // Rollback Transaction
        signOnImportPersonAPI.rollbackTransaction();
      } catch (SignOnImportPersonAPIException se) {
        log.error("importPerson: Error while Rolling back transaction", se);
      }
    } finally {
```

```
      // Release Transaction
      signOnImportPersonAPI.releaseTransaction();
    }

    IExtUserDTO extUserDTO = PersonFactory.createExtUserDTO();
    extUserDTO.setPersonDTO(personDTO);

    IEUIPersonSearchOperationResult psor =
euiPersonSearchOperationContext.getEUIPersonSearchOperationResult();
    psor.setImportedPersonExtDTO(extUserDTO);

    log.debug("importPerson: Leaving importPerson method...");

    return psor;
  }

  /**
   * Implement Import Manager Operation and Import all the Supervisors chain
   * of the Person being imported
   *
   * @param euiOperationDTO       .
   * @param euiPersonSearchOperationContext
   *                              .
   * @param request               .
   * @param signOnImportPersonAPI
   * @param ldapApi
   * @return .
   * @throws EUIException .
   */
  public IEUIPersonSearchOperationResult importManager(IEUIEventImportManagerOperationDTO
euiOperationDTO,
                                                       IEUIPersonSearchOperationContext
euiPersonSearchOperationContext,
                                                       HttpServletRequest request,
                                                       ISignOnImportPersonAPI
signOnImportPersonAPI, ILDAPApi ldapApi)
      throws EUIException {
    return null;
  }

  /**
   * Implement any Custom Operation
   *
   * @param euiOperationDTO       .
   * @param euiPersonSearchOperationContext
   *
   * @param request               .
   * @param signOnImportPersonAPI
   * @param ldapApi
   * @return .
   * @throws EUIException .
   */
  public IEUIPersonSearchOperationResult performCustom(IEUIEventCustomOperationDTO
euiOperationDTO,
                                                       IEUIPersonSearchOperationContext
euiPersonSearchOperationContext,
                                                       HttpServletRequest request,
                                                       ISignOnImportPersonAPI
signOnImportPersonAPI, ILDAPApi ldapApi)
      throws EUIException {
    return null;
  }
}
```

サポートされるタイム ゾーン

タイム ゾーンのマッピングにサポートされているタイム ゾーンは、次のとおりです。

表 8-18　　サポートされている時間帯

タイム ゾーン名	GMT 相当
Etc/GMT+12	(GMT-12:00) 日付変更線、西側
Pacific/Apia	(GMT-11:00) サモア
US/Hawaii	(GMT-10:00) ハワイ
US/Aleutian	(GMT-10:00) ハワイ アリューシャン夏時間
US/Alaska	(GMT-09:00) アラスカ
America/Tijuana	(GMT-08:00) 太平洋標準時（米国およびカナダ）
America/Chihuahua	(GMT-07:00) チワワ、ラパス、マサトラン
US/Arizona	(GMT-07:00) アリゾナ
Canada/Mountain	(GMT-07:00) 山岳部標準時（米国およびカナダ）
Canada/Saskatchewan	(GMT-06:00) サスカチュワン州
US/Central	(GMT-06:00) 中央アメリカ
Canada/Central	(GMT-06:00) 中央標準時（米国およびカナダ）
America/Mexico_City	(GMT-06:00) グアダラハラ、メキシコシティー、モ
America/Bogota	(GMT-05:00) ボゴタ、リマ、キト
Canada/Eastern	(GMT-05:00) 東部夏時間（米国およびカナダ）
America/Jamaica	(GMT-05:00) 東部標準時（米国およびカナダ）
US/East-Indiana	(GMT-05:00) インディアナ（東部）
America/Antigua	(GMT-04:00) 大西洋標準時（カナダ）
Canada/Atlantic	(GMT-04:00) 大西洋夏時間（カナダ）
America/Manaus	(GMT-04:00) マナウス
America/Santiago	(GMT-04:00) サンチャゴ
America/Caracas	(GMT-04:30) カラカス
America/La_Paz	(GMT-04:00) ラパス（ボリビア）
America/Sao_Paulo	(GMT-03:00) ブラジリア
America/Godthab	(GMT-03:00) グリーンランド
America/Argentina/Buenos_Aires	(GMT-03:00) ブエノスアイレス
America/Guyana	(GMT-04:00) ジョージタウン
America/St_Johns	(GMT-03:30) ニューファンドランドおよびラブラ
Atlantic/South_Georgia	(GMT-02:00) 中部大西洋
Atlantic/Azores	(GMT-01:00) アゾレス諸島
Atlantic/Cape_Verde	(GMT-01:00) カーボベルデ諸島
Etc/Greenwich	(GMT) グリニッジ標準時、ダブリン、エディンバ
Africa/Casablanca	(GMT) カサブランカ、モンロビア
Europe/Sarajevo	(GMT+01:00) サラエボ、スコピエ、ワルシャワ、ザグ
Europe/Brussels	(GMT+01:00) ブリュッセル、コペンハーゲン、マド
Africa/Brazzaville	(GMT+01:00) アフリカ中西部
Europe/Amsterdam	(GMT+01:00) アムステルダム、ベルリン、ベルン、

表 8-18　サポートされている時間帯（続き）

タイム ゾーン名	GMT 相当
Europe/Belgrade	(GMT+01:00)ベオグラード、ブラチスラバ、ブダペ
Africa/Cairo	(GMT+02:00)カイロ
Europe/Helsinki	(GMT+02:00)ヘルシンキ、キエフ、リガ、ソフィア、
Europe/Minsk	(GMT+02:00)ミンスク
Europe/Athens	(GMT+02:00)アテネ、ブカレスト、イスタンブール
Asia/Jerusalem	(GMT+02:00)エルサレム
Africa/Windhoek	(GMT+02:00)ビントフック
Africa/Harare	(GMT+02:00)ハラーレ、プレトリア
Asia/Baghdad	(GMT+03:00)バグダッド
Africa/Nairob	(GMT+03:00)ナイロビ
Europe/Moscow	(GMT+03:00)モスクワ、サンクトペテルスブルク、
Asia/Kuwait	(GMT+03:00)クウェート、リヤド
Asia/Tehran	(GMT+03:30)テヘラン
Asia/Baku	(GMT+04:00)バクー
Asia/Muscat	(GMT+04:00)アブダビ、マスカット
Asia/Yerevan	(GMT+04:00)エレヴァン
Asia/Tbilisi	(GMT+04:00)トビリシ
Asia/Kabul	(GMT+04:30)カブール
Asia/Karachi	(GMT+05:00)イスラマバード、カラチ、タシケント
Asia/Yekaterinburg	(GMT+05:00)エカチェリンブルグ
Asia/Kolkata	(GMT+05:30)チェンナイ、コルカタ、ムンバイ、
Asia/Kathmandu	(GMT+05:45)カトマンズ
Asia/Dhaka	(GMT+06:00)アスタナ、ダッカ
Asia/Novosibirsk	(GMT+07:00)ノボシビルスク
Asia/Colombo	(GMT+05:30)スリジャヤワルダナプラコッテ
Asia/Rangoon	(GMT+06:30)ヤンゴン(ラングーン)
Asia/Bangkok	(GMT+07:00)バンコク、ハノイ、ジャカルタ
Asia/Krasnoyarsk	(GMT+08:00)クラスノヤルスク
Asia/Irkutsk	(GMT+09:00)イルクーツク
Asia/Kuala_Lumpur	(GMT+08:00)クアラルンプール、シンガポール
Asia/Taipei	(GMT+08:00)タイペイ
Australia/Perth	(GMT+08:00)パース
Asia/Chongqing	(GMT+08:00)北京、重慶、香港特別自治区、ウルムチ
Asia/Seoul	(GMT+09:00)ソウル
Asia/Tokyo	(GMT+09:00)大阪、札幌、東京
Asia/Yakutsk	(GMT+09:00)ヤクーツク
Australia/Darwin	(GMT+09:30)ダーウィン
Australia/Adelaide	(GMT+09:30)アデレード
Australia/Hobart	(GMT+10:00)ホーバート
Australia/Canberra	(GMT+10:00)キャンベラ、メルボルン、シドニー

表 8-18　サポートされている時間帯(続き)

タイム ゾーン名	GMT 相当
Australia/Brisbane	(GMT+10:00)ブリズベン
Asia/Vladivostok	(GMT+10:00)ウラジオストク
Pacific/Guam	(GMT+10:00)グアム、ポートモレスビー
Pacific/Guadalcanal	(GMT+11:00)ソロモン諸島、ニューカレドニア
Pacific/Auckland	(GMT+12:00)オークランド、ウェリントン
Pacific/Fiji	(GMT+12:00)フィジー島
Pacific/Tongatapu	(GMT+13:00)ヌークアロファ

build.xml ファイルの例

```
<?xml version="1.0" ?>
<project name="Sample Project" default="all" basedir=".">
- <!-- Main target -->
  <target name="all" depends="init,build,deploy" />
  <!- Set the following properties to point to appropriate folders -->
  <property name="rcwar.dir" value="<apps server path where Request Center application WAR
file is deployed>" />
  <property name=" javax.servlet.dir" value="<path where
jboss-servlet-api_3.0_spec-1.0.0.Final.jar is available in the app server>" />
  <property name="rcwar_webinf_classes.dir"value="${rcwar.dir}/WEB-INF/classes" />
  <target name="init">
    <property name="dirs.base" value="${basedir}" />
    <mkdir dir="${dirs.base}/out" />
    <property name="src" value="${dirs.base}/src" />
    <property name="out" value="${dirs.base}/out" />
  </target>
  <path id="classpath">
    <fileset dir="${rcwar.dir}" includes="*.jar" />
    <fileset dir="${javax.servlet.dir}"
includes="jboss-servlet-api_3.0_spec-1.0.0.Final.jar" />
    <pathelement path="${rcwar_webinf_classes.dir}" />
  </path>
- <!-- Compile Java Files -->
  <target name="build" depends="init">
    <javac srcdir="${src}" destdir="${out}" debug="true" includes="**/*.java"
classpathref="classpath" deprecation="true" fork="true" memoryinitialsize="256M"
memorymaximumsize="512M" />
  </target>
  <target name="deploy" depends="init">
    <copy todir="${rcwar_webinf_classes.dir}">
      <fileset dir="${out}">
        <include name="**/*.class" />
      </fileset>
    </copy>
  </target>
</project>
```