Cisco Prime Service Catalog 12.0 統合ガイド

2016 年 11 月

Cisco Systems, Inc.
www.cisco.com

シスコは世界各国 200 箇所にオフィスを開設しています。
所在地、電話番号、FAX 番号
は以下のシスコ Web サイトをご覧ください。
www.cisco.com/go/offices

【注意】シスコ製品をご使用になる前に、安全上の注意（www.cisco.com/jp/go/safety_warning/）をご確認ください。

本書は、米国シスコシステムズ発行ドキュメントの参考和訳です。リンク情報につきましては、日本語版掲載時点で、英語版にアップデートがあり、リンク先のページが移動 / 変更されている場合がありますことをご了承ください。
あくまでも参考和訳となりますので、正式な内容については米国サイトのドキュメントを参照ください。

また、契約等の記述については、弊社販売パートナー、または、弊社担当者にご確認ください。

Cisco Prime Service Catalog 12.0 統合ガイド

CONTENTS

使用する前に

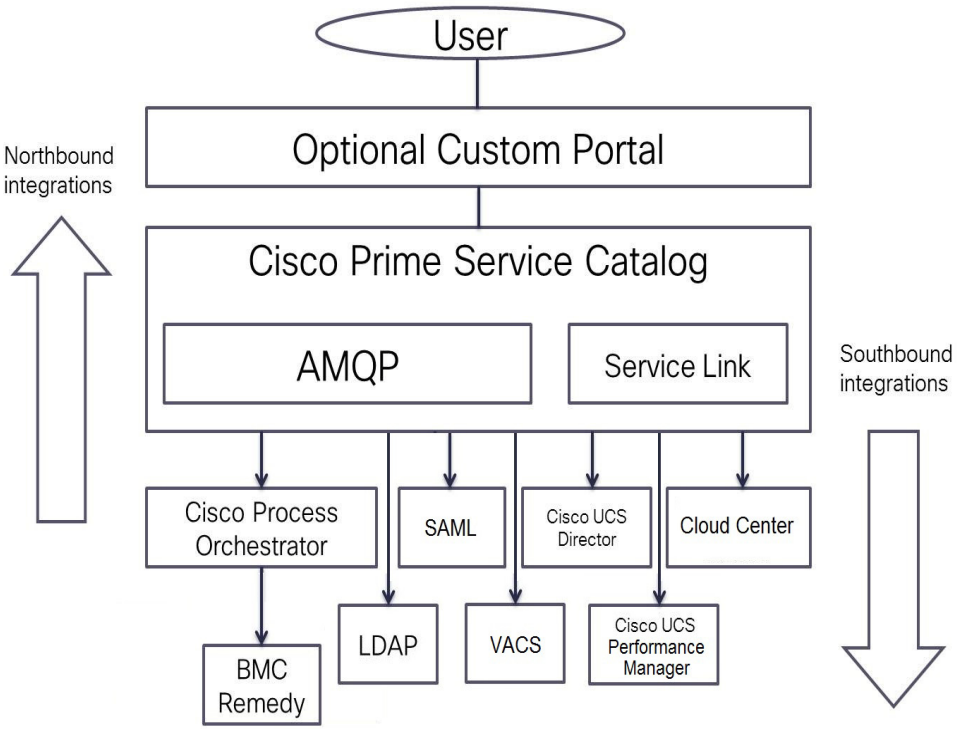
Cisco Prime Service Catalog は、企業 IT で使用される（ユーザごとに販売されるライセンス）、およびクラウド コンピューティングで使用される（サーバごとに販売されるライセンス）、セルフ サービス ポータル、サービス カタログ、およびライフサイクル管理ソフトウェアを従業員に提供します。これらのサービスはサービス設計者により Cisco Prime Service Catalog で設計され、Service Catalog ポータルでエンド ユーザが注文できるようになっています。

こうしたサービスには IT サポートが従業員にラップトップを提供するためにラップトップを注文するなどの単純なサービスもあれば、さまざまなタスクの自動化やオーケストレーションを必要とするアプリケーション スタックのプロビジョニングなどの複雑なサービスもあります。

自動化やオーケストレーションを必要とするサービスを実行できるように、Prime Service Catalog は外部システムと統合されます。外部システムには外部ディレクトリ、ドメイン マネージャ、BMC Remedy SRM、Cisco UCS Director などがあります。

サービスを要求したりサービス項目を管理したりするコールを発信するアプリケーションは統合設計のノースバウンドにあり、要求を受信し、必要な処理を実行し、応答を送信するアプリケーションは統合設計アーキテクチャのサウスバウンドにあります。たとえば、サービス要求のために Service Catalog に API コールを発信する企業のカスタム ポータルは、Prime Service Catalog のノースバウンドにあり、必要な操作を実行するために Prime Service Catalog から API コールを受信する他のアプリケーションは、Prime Service Catalog のサウスバウンドにあります。

図 1-1 統合シナリオの例



サービス要求/提供のワークフローにおいて、Prime Service Catalog は Cisco Process Orchestrator と統合されます。統合された Cisco Process Orchestrator は他のアプリケーションと対話してサービス要求を実行できるようにします。統合の設計によっては、外部アプリケーションと直接対話する場合もあります。つまり、Prime Service Catalog は、サウスバウンド コールを行って直接、または Cisco Process Orchestrator を経由して、外部アプリケーションと対話します。

Service Catalog が企業ポータルと統合される実装もいくつかあります。これは、各サービスが企業ポータル内で企業により定義されており、Prime Service Catalog を使用して設計、提供されることを意味します。したがって、エンド ユーザは企業ポータルを使用してサービス要求を送信し、ポータルは Prime Service Catalog と対話してサービスを提供します。

Prime Service Catalog 統合に関する次のシナリオを考えます。

シナリオ:ユーザは自分自身を組織グループに追加する必要があります。

ワークフロー:

1. Service Designer によりサービスが設計されます。このとき、サービス フォームには次の詳細が含まれています。

   ユーザ ID、グループ名(ドロップダウン リスト)、ユーザ名、要求名「Join Group」。
2. ユーザが Service Catalog にログインし、サービス「Join group」を要求します。
3. ユーザがサービス フォームに詳細を入力し、[送信(submit)] をクリックします。
4. Prime Service Catalog はユーザの詳細とサービス フォーム属性をサービス フォームから取得し、それを Service Link に送信します。

5. Service Link が Active Directory に対して、ユーザをグループに追加する API コールを行います。
6. Active Directory が要求を実行し、HTTP 応答 200 を返してワークフローが完了します。

（注）　Prime Service Catalog の Service Link モジュールにより、Prime Service Catalog の外部システムとの統合が実現できます。

PART 1

ノースバウンド統合

Cisco RESTful SOAP ベース API の概要

概要

Cisco Prime Service Catalog は、サービス オーダーまたは要求を送信してサービス カタログ内で定義されたエンティティにアクセスするための、一連の RESTful API を介したノースバウンド統合をサポートしています。Service Catalog には、要求を送信して管理するための、一連の古い SOAP ベースの API もあります。

シスコでは、サービス カタログ（Service Catalog）で定義されたエンティティにアクセスするために、一連の標準的な REST（Representational State Transfer）API および Java スタブを提供しています。これらは、nsAPI と総称されています。nsAPI の発信者はセッションを確立するために、有効な Service Catalog アカウントを使って最初に自身を認証する必要があります。

Service Catalog のエンティティに対するアクセス権限は、サービス カタログ（Service Catalog）アプリケーションでユーザに対して定義されたロールベース アクセス コントロール（RBAC）のオブジェクト レベル権限によって制御されます。SSO を使用してサービス カタログ（Service Catalog）アプリケーションにサインインした場合、SSO トークンは API に自動的に渡されます。

外部アプリケーションからの呼び出しをサポートしている以外に、nsAPI は Service Portal モジュール内から呼び出すこともできます。ポータル機能は、Java、JavaScript、または HTML を使用して作成されたポートレットの設計や表示をサポートします。このようなポートレット内で、nsAPI を呼び出して必要なエンティティの情報を取得し、特定のタイプのエンティティのデータを更新可能にすることができます。Portal モジュールに関する詳細については、『*Cisco Prime サービス カタログ（Service Catalog）Service Catalog Designer Guide*』を参照してください。

新規、変更、および非推奨 API

次の表は、このリリースの新規および変更 API の情報を説明します。

表 2-1　新規および変更 API 情報テーブル

API の説明	REST の URL	ステータス	参照先
サービスをコピー。	http://<ServerURL>/RequestCenter/nsapi/definition/v1/service	新規	表 3-8 サービス API テーブル（3-26 ページ）
要求の情報を取得。	http://<ServerURL>/RequestCenter/nsapi/definition/v1/ordermgmt/reqinfo/reqid/{reqId}	新規	表 3-16 オーダー管理 API テーブル（3-52 ページ）

表 2-1　新規および変更 API 情報テーブル（続き）

API の説明	REST の URL	ステータス	参照先
要求のためのサービスの情報を取得。	http://<ServerURL>/RequestCenter/nsapi/definition/v1/ordermgmt/servicenfo/reqid/{reqId}	新規	表 3-16 オーダー管理 API テーブル（3-52 ページ）
要求に関するすべてのシステム コメントおよびユーザ コメントを取得。	http://<ServerURL>/RequestCenter/nsapi/definition/v1/ordermgmt/comments/reqid/{reqId}	新規	表 3-16 オーダー管理 API テーブル（3-52 ページ）
要求のすべての添付ファイルの詳細を取得。	http://<ServerURL>/RequestCenter/nsapi/definition/v1/ordermgmt/attachmentslist/reqid/{reqId}	新規	表 3-16 オーダー管理 API テーブル（3-52 ページ）
要求のドキュメント ID を使用して添付ファイルをダウンロード。	http://<ServerURL>/RequestCenter/nsapi/definition/v1/ordermgmt/attachment/viewdocid/{document Id}	新規	表 3-16 オーダー管理 API テーブル（3-52 ページ）
ログイン中のユーザがオーダーできるすべてのサービス ID を取得。	http://<ServerURL>/RequestCenter/nsapi/definition/v1/orderableserviceids	新規	表 3-16 オーダー管理 API テーブル（3-52 ページ）
ログイン中のユーザがログイン ID がわかっている顧客に代わってオーダーできるすべてのサービスを取得。	http://<ServerURL>/RequestCenter/nsapi/definition/v1/orderableserviceids/customerlogin/person	新規	表 3-16 オーダー管理 API テーブル（3-52 ページ）
サービス フォームの詳細を取得。	http://<ServerURL>/RequestCenter/nsapi/transaction/v1/orderform?serviceids=<サービス ID のリスト、カンマ区切り>	新規	表 3-21 要求 API テーブル（3-76 ページ）
すべてのサービス項目を取得する API。	http://<Server URL>/RequestCenter/nsapi/serviceitem/v1/mdrdatatable/serviceitemlist?	新規	表 3-29すべてのサービス項目 API テーブル（3-109 ページ）
すべてのサービス項目の CSV をダウンロード。	http://<ServerURL>/RequestCenter/nsapi/serviceitem/v1/mdrdatatable/serviceitem	新規	表 3-29すべてのサービス項目 API テーブル（3-109 ページ）

表 2-1　新規および変更 API 情報テーブル（続き）

API の説明	REST の URL	ステータス	参照先
履歴情報の取得	http://<Server URL>/RequestCenter/nsapi/serviceitem/v1/myservices/assetlist/history?	新規	表 3-29すべてのサービス項目 API テーブル（3-109 ページ）
分類ツリーの取得	http://<ServerURL>/RequestCenter/nsapi/serviceitem/v1/sim/serviceitemtypestree/classificationdatatypetree?	新規	表 3-29すべてのサービス項目 API テーブル（3-109 ページ）
ユーザ ロールを取得する API	http://<ServerURL>/RequestCenter/nsapi/directory/tenant/v2/userroles	新規	表 3-35 テナント管理 API テーブル（3-129 ページ）
テナント管理設定を取得する API	http://<ServerURL>/RequestCenter/nsapi/directory/tenant/v2/tenantconfigurationRelational Operators	新規	表 3-35 テナント管理 API テーブル（3-129 ページ）
管理者メンバー情報を含むカードおよびリストを表示するため、階層的チームとチーム詳細を取得する API	http://<ServerURL>/RequestCenter/nsapi/directory/tenant/v2/id/{teamId}	新規	表 3-35 テナント管理 API テーブル（3-129 ページ）
管理者メンバー情報を含むリストを表示するため、チーム リストを取得する API	http://<ServerURL>/RequestCenter/nsapi/directory/tenant/v2/teamlist?startRow={startRow}&recordSize={recordSize}&sortBy={sortBy}&sortDir={sortDir}&name={searchName}	新規	表 3-35 テナント管理 API テーブル（3-129 ページ）
親チーム ID でサブチームを取得する API	http://<ServerURL>/RequestCenter/nsapi/directory/tenant/v2/id/{teamid}/teamlist?startRow={startRow}&recordSize={recordSize}&sortBy={sortBy}&sortDir={sortDir}&name={searchName}	新規	表 3-35 テナント管理 API テーブル（3-129 ページ）
チーム メンバー リストを取得する API	http://<ServerURL>/RequestCenter/nsapi/directory/tenant/v2/id/{teamId}/memberlist?startRow={startRow}&recordSize={recordSize}&sortBy={sortBy}&sortDir={sortDir}&name={searchName}	新規	表 3-35 テナント管理 API テーブル（3-129 ページ）
メンバーを AS チーム管理者に昇格させる API	http://<ServerURL>/RequestCenter/nsapi/directory/tenant/v2/id/{teamId}/people/id/{personId}/promoteTeamAdmin	新規	表 3-35 テナント管理 API テーブル（3-129 ページ）
Cloud Center ユーザの API キーの更新	http://<ServerURL>/RequestCenter/nsapi/directory/people/id/{personId}/updateapikey	新規	表 3-35 テナント管理 API テーブル（3-129 ページ）
すべての IDP 設定の取得	http://<ServerURL>/RequestCenter/nsapi/v1/idp/configs	新規	表 9-1 SAML REST API テーブル（9-5 ページ）
IDP 設定の削除	http://<ServerURL>/RequestCenter/nsapi/v1/idp/configs/<idp configuration name>	新規	表 9-1 SAML REST API テーブル（9-5 ページ）

表 2-1　新規および変更 API 情報テーブル（続き）

API の説明	REST の URL	ステータス	参照先
IDP 設定の取得	http://<ServerURL>/RequestCenter/nsapi/v1/idp/configs/<idp configuration name>	新規	表 9-1 SAML REST API テーブル（9-5 ページ）
ノードのメタデータの更新	http://<ServerURL>/RequestCenter/nsapi/v1/idp/refreshThis	新規	表 9-1 SAML REST API テーブル（9-5 ページ）
IDP 設定の保存	http://<ServerURL>/RequestCenter/nsapi/v1/idp/configs	新規	表 9-1 SAML REST API テーブル（9-5 ページ）
IDP 設定の更新	http://<ServerURL>/RequestCenter/nsapi/v1/idp/configs	新規	表 9-1 SAML REST API テーブル（9-5 ページ）
SAML 設定の取得	http://<ServerURL>/RequestCenter/nsapi/v1/saml/configs	新規	表 9-1 SAML REST API テーブル（9-5 ページ）
SAML 設定の更新	http://<ServerURL>/RequestCenter/nsapi/v1/saml/configs	新規	表 9-1 SAML REST API テーブル（9-5 ページ）
AMQP サーバ詳細の作成	http://<ServerURL>/RequestCenter/nsapi/ucsd/discovery/saveConnection	更新しました	表 3-1 サポートされているフィルタ テーブル（3-2 ページ）
接続の削除	http://<Server URL>/RequestCenter/nsapi/ucsd/discovery/deleteConnection	新規	表 3-1 サポートされているフィルタ テーブル（3-2 ページ）
認証キー生成 API	http://<ServerURL>/RequestCenter/nsapi/messagebroker/overview?connectionIdentifier=<particular amqp connection identifier>	新規	認証キー生成 API

HTTP クライアントでの REST ベース API の使用

Cisco Prime Service Catalog がサポートしている REST API（別名 nsAPI）を使用する一般的な方法は、HTTP ベースのクライアントを使用する方法です。クライアントがこれらの API を呼び出すには、最初に呼び出し元を認証する必要があります。nsAPI では、ヘッダー ベース認証とトークン ベース認証の 2 つの認証方法がサポートされています。

ヘッダー ベース認証による nsAPI の呼び出し

RESTful nsAPI 呼び出しは、発信者が自身を認証できる場合にだけ処理されます。これは、ブラウザを介して以下のページでユーザが Prime Service Catalog に最初にログインするときに似ています。

http://<serverURL>/RequestCenter

次に、Service Catalog に正常にログインすると、ブラウザのアドレス バーに有効な nsAPI REST の URL を入力します。例を以下に示します。

```
http://<serverURL>/RequestCenter/nsapi/definition/categories/id/3
```

次のような応答 XML がブラウザに表示されます。

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<category>
  <categoryId>3</categoryId>
  <categoryName>Workplace Services</categoryName>
  <description>Services for voice and data communications, desktop, mobile devices, and
application access.</description>
  <topDescriptionEnabled>false</topDescriptionEnabled>
  <topDescription />
  <middleDescriptionEnabled>false</middleDescriptionEnabled>
  <middleDescription />
  <bottomDescriptionEnabled>false</bottomDescriptionEnabled>
  <bottomDescription />
  <catalogTypeId>1</catalogTypeId>
  <catalogType>Consumer Services Catalog</catalogType>
  <isRoot>false</isRoot>
  <associatedServices>
    <associatedService>
      <description>Order a new or refurbished laptop. Manager approval
required.</description>
      <id>20</id>
      <name>New Laptop</name>
      <status>Active</status>
    </associatedService>
    <associatedService>
      <description>Order a new iPhone or Blackberry, configured and maintained under
corporate policy.</description>
      <id>22</id>
      <name>New Mobile Device</name>
      <status>Active</status>
    </associatedService>
  </associatedServices>
  <includedCategories>
    <includedCategory>
      <id>8</id>
      <name>Email</name>
    </includedCategory>
    <includedCategory>
      <id>9</id>
      <name>Laptops</name>
    </includedCategory>
  </includedCategories>
  <categoryURLSc>
    <a
href='/RequestCenter/myservices/navigate.do?categoryid=3&amp;query=catalog&amp;layout=popu
p_p' onclick="return GB_showFullScreen('Category', this.href)">Workplace Services</a>
  </categoryURLSc>
  <categoryURLOnlySc>/RequestCenter/myservices/navigate.do?categoryid
=3&amp;query=catalog</categoryURLOnlySc>
</category>
```

REST API 要求がアプリケーションにログインする前に実行された場合、URL は次のエラーを返します。

```
HTTP Error 401 Unauthorized
```

RESTful nsAPI 呼び出しの認証

HTTP ヘッダーには、次のパラメータを含める必要があります。

username=<username>

password=<password>

すべての nsAPI および RAPI 要求で、**HTTP Header- acceptEncryptedPassword=true** が設定されていれば、パスワードは暗号化されていると見なされます。

パスワードは、次の場合に暗号化されます。

- Accept Encrypted Password の値が、[管理（Administration）] > [設定（Settings）] ページで有効（オンに設定）になっており、HTTP ヘッダーで false に設定されている場合。
- Accept Encrypted Password の値が、[管理（Administration）] > [設定（Settings）] ページで無効（オフに設定）になっており、HTTP ヘッダーで true に設定されている場合。

（注）　password 属性の暗号化された値と cloudpassword 属性のクリアテキスト値を非表示にするには、newscale.properties ファイルのパラメータ nsapi.directory.person.hide.secure.information を true に設定します。

認証が成功すると、HTTP 応答で JSessionID cookie が返されます。後続の nsAPI の呼び出しで認証を再度行わずにセッションを維持するには、この呼び出しの要求に同じ JSessionID cookie を含める必要があります。

認証時の応答コードは、ユーザ アカウントのパスワードの期限切れや失敗したパスワード試行により、ロックされているかどうかが示されます。ユーザ アカウントのロックを解除するには、システム管理者に問い合わせてください。

（注）　また、ユーザ認証の応答コードは、ユーザ アカウントが猶予期間であるかどうかと、ユーザ パスワードの更新期限の日付も示します。

パスワード ポリシーの詳細情報については、『Cisco Prime Service Catalog Administration and Operations Guide』の「Enforcing Password Policies」を参照してください。

API セッションは、セッション タイムアウトが発生するか、または任意のユーザに nsapi ログアウトが呼び出された場合に、非アクティブになります。nsapi を使用したログアウトの URL は次のとおりです。

```
RequestCenter/nsapi/authentication/logout
```

次の API は、セッションがまだ有効であるかどうかを確認するために使用されます。クライアント インターフェイスは、セッションが有効な場合は HTTP コード 200 を、それ以外の場合は HTTP 401 コードを返します。

```
http://<ServerURL>:8088/RequestCenter/nsapi/authentication/session
```

次のような応答 XML がブラウザに表示されます。

```
<nsapi-response utid="235c3faa2bbade2ebc85afc2b7f29bd3">User is
authenticated.</nsapi-response>
```

Service Catalog アプリケーションのセッションのタイム アウト設定に関する詳細情報については、『*Cisco Prime Service Catalog Administration and Operations Guide*』の「Site Administration」の章を参照してください。

(注)　いずれかの nsAPIs が（nsAPI ログインを呼び出すことなく）クレデンシャルにより直接呼び出された場合、セッションは応答の送信後に自動的に終了する必要があります。nsAPI ログインが明示的に呼び出された場合、nsAPI ログアウトが手動で呼び出されるか、API タイムアウトの新しい管理設定（[管理設定（Administration Settings）] > [APIタイムアウト（API Timeout）]）に基づいてセッションがタイムアウトしない限り、nsAPI セッションは自動的に終了しません。

(注)　nsAPI をクロスサイト リクエスト フォージェリ（CSRF）によるセキュリティ脆弱性から保護するには、newscale.properties ファイルの session.token.validation の値を 1 に設定します。さらに、セッションに認証トークンを提供する場合、後続の nsAPI 呼び出しにユーザ ID とパスワードを入力する必要はありません。

トークン ベース認証の使用

Prime Service Catalog はトークン ベースの認証メカニズムをサポートしています。要求（呼び出し）ごとにユーザ名とパスワードで認証を行うのではなく、1 回認証して、期間限定のトークンを代わりに取得できます。トークンを（GET 要求を使用して）入手したら、ユーザ名とパスワードを入力しなくても、後続の RESTful 呼び出しにそのトークンを使用できます。

トークン ID の取得

次の nsAPI GET 呼び出しでトークンを要求します。HTTP ヘッダーに有効なユーザ名とパスワードが指定されていることを確認してください。

```
http://<ServerURL/RequestCenter/nsapi/authentication/token?persistent=true
```

サンプル応答:

```
<sessiontoken utid="P_D887A7375EC640D725A1D042FBFBEAFE"/>
```

トークン検証パラメータ値の設定

次のように、newscale.properties ファイルで *session.token.validation* パラメータの値を設定します。

```
session.token.validation=1
```

さらに、[管理（Administration）] > [設定（Settings）] タブの [API セッションタイムアウト（API Session Timeout）] フィールドで、トークン有効性のタイムアウト時間を設定します。

nsAPI の呼び出しに対するトークン検証の設定

Service Catalog からの nsAPI の呼び出しに対し、トークン ベースの検証を有効にできます。この設定は任意です。検証を制御するには、newscale.propertiesファイルでパラメータ *session.token.nsapisc.validation* を 1 に設定します。

アップグレードを行う予定で、外部システムからの既存の nsAPI 呼び出しがこのトークン ベース検証の影響を受ける場合は、このプロパティの値を 0 に設定することを検討してください。

(注)　newscale.properties ファイルを変更したら、その都度、Service Catalog サーバを再起動する必要があります。

トークンの有効期限の指定

nsAPI で使用されるトークンが期限切れの場合、nsAPI の発信者に次のエラーが返されます。

<nsapi-error-response errorcode="AUTH_0014"> Token Invalid</nsapi-error-response>

トークンの使用

後続の REST ベースの nsApi 呼び出しには、REST 呼び出しから取得されたトークン ID を HTTP ヘッダーとして追加する必要があります(ユーザ名とパスワードは不要です)。

utid=<token_id>

サポートされているエンティティ

nsAPI でサポートされるエンティティは、次のカテゴリに分けられます。

表 2-2　サポートされているエンティティ

エンティティ グループ	エンティティ タイプ
定義データ	カテゴリ(Categories) サービス エージェント(Agents) 課金レート ポリシー
ディレクトリ データ	組織 人物 グループ(Groups) アカウント
トランザクション データ	契約 要求 要求エントリ 承認 タスク
サービス項目および標準	サービス項目の詳細 すべてのサービス項目 標準(Standards) 課金履歴 ポリシー アラート
Portal Designer データ	コンテンツのカスタマイズ テーブル

サポートされる操作

次のタイプの操作が、nsAPI でサポートされています。

- すべてのエンティティのデータ取得に対する HTTP GET 操作
- 個人、アカウント、契約、課金レート、ポリシー、およびサービス項目のインスタンスを作成または更新するための HTTP POST 操作と PUT 操作。
- タスク処理を実行するため、およびサービス項目権限を付与/取り消すための HTTP POST 操作。
- アカウント、契約、課金レート、ポリシー、およびサービス項目インスタンスを削除するための HTTP DELETE 操作。

要求と応答の形式

すべての nsAPI は要求のコンテンツ タイプに「application/xml」の使用をサポートしています。次のエンティティは、application/json コンテンツ タイプもサポートしています。

- サービス項目（Service Items）
- アカウント、契約
- 課金レート、課金履歴
- ポリシーおよびポリシー アラート

デフォルトの応答のコンテンツ タイプは要求のコンテンツ タイプと同じですが、クエリ パラメータ「responseType」（xml または json）で上書きできます。

例:

サービスを追加して、新しい要求/カートを送信します。

Method: POST

REST URL:

/RequestCenter/nsapi/transaction/requisitions

ペイロード:

```
{
  "requisition": {
    "customerLoginName": "admin",
    "billToOU": "H_OU",
    "services":[{
          "name": "TestServiceRest"
          "quantity": "1",
          "version": "0",
          "dictionaries": [{
              "name": "TestNonGrid",
              "data": {
                  "FullName": "AAB",
                  "HireDate": "02/20/1978",
"MultiSelect":["MS3","MS4"],
}
},{
              "name": "TestG",
              "data":
                [
```

```
                {"city":"Bangalore", "country": "India"},
                {"city":"Mysore", "country":"India"}
            ]
        }]
    }]
  }
}
```

成功コード:201

応答エラー コード:REST/Web サービスのエラー メッセージの表を参照してください。

サービス オーダーまたはサービス要求の送信と管理

ここでは、RESTful クライアントが Prime Service Catalog にオーダーを出す方法、確認のためのステータスの読み取り、そしてオーダーのキャンセルについて説明します。サービス オーダーやサービス要求、および要求エントリでサポートされる API は他にもあり、そうした API はこのマニュアルの参照の項に記載されています。

サービス オーダーでの RESTful API の使用

サービス オーダーまたはサービス要求の送信

Cisco Prime Service Catalog でサービス オーダーまたはサービス要求を送信するよう、RESTful クライアントを書き込むときには、次の手順を実行する必要があります。

1. サービスを発注するための RBAC 権限がある有効なユーザ アカウントを使用して、クライアントを認証します。

   コマンドの発行

   http://<ServerURL>/RequestCenter/nsapi/authentication/token?persistent=true

   HTTP ヘッダーでユーザ名とパスワードを指定する

   呼び出しが成功した場合、認証トークンを受信します。
   <sessiontoken utid="P_D887A7375EC640D725A1D042FBFBEAFE"/>

2. JSON 形式で API ペイロードを作成します。オーダーが UI で送信された場合、これらのデータは通常、ユーザがオーダー フォームに入力したデータです。

   どの Prime Service Catalog サービスに要求を送信する必要があるか、どのフォーム フィールド値を入力する必要があるかが把握されているものとして、サービス要求または要求のペイロードを作成します。

   ペイロード:

```
{
    "requisition": {
        "services": [
            {
                "version": "0",
                "dictionaries": [
                    {
                        "data": {
                            "Name": "DockerTr9"
                        },
                        "name": "Application_Information"
                    }
                ],
                "name": "Docker1",
```

```
                "quantity": "1"
            }
        ],
        "customerLoginName": "combo1user1",
        "billToOU": "UCSD::uc1::Fenced_Group1"
    }
}
```

サービスが他のユーザに対してオーダーされたことを示すには、「customerLoginName」フィールドを使用します。

(注)　RBAC の設定で定義された権限に基づいて、別のユーザの代わりにオーダーできます。

前の手順で受信した認証トークン（sessiontoken utid=<token_id>）を、HTTP ヘッダーで指定します。

オーダーが正常に送信されると、Prime Service Catalog は成功コード：201 を返します。

応答のペイロードは次のとおりです。

```
{
  "RequisitionSubmit": {
     "id": 96,
     "customer": "combo1user1 combo1user1",
     "initiator": "combo1user1 combo1user1",
     "startedDateRaw": 1444267887000,
     "startedDate": "10/07/2015 6:31 PM",
     "status": "Ordered"
  }
}
```

応答エラー コード：REST/Web サービスのエラー メッセージの表を参照してください。

3. オーダーが入ったため、RESTful API クライアントは、要求 ID を使用します。この要求 ID は、オーダーのステータスを監視するために前の手順で受信したものです。
オーダーはサービス オーケストレーションと配信プロセスに応じて、完了までに数分から数日かかる場合があります。nsAPI クライアントは以前に受信した同じ utid=<token_id> トークンを利用する場合も、新しいトークンを取得するために認証を行う場合もあります。

   オーダーのステータスを取得するために RESTful 呼び出しを実行する

   HTTP Method: GET

   http://<ServerURL>/RequestCenter/nsapi/transaction/requisitions/id/<requisitionId>

   または、ユーザがオーダーを行ったときに受け取った応答のペイロードには、要求のステータスを確認するために使用できる URL も含まれています。

カスタム ポータルに対して行ったオーダー/要求をキャンセルする場合は、カスタム ポータルのキャンセルを呼び出すことができます。

この場合、ポータルの基礎となる RESTful API クライアントが、Prime Service Catalog に対して対応する要求を発行します。

HTTP メソッド：DELETE

http://<ServerURL>/RequestCenter/nsapi/transaction/requisitionentries/<reqEntryId>?

ペイロードは不要です。

成功コード：200

応答エラー コード：REST/Web サービスのエラー メッセージの表を参照してください。

サービス オーダーのレガシー SOAP ベースの RAPI

ここでは、Service Catalog の Web サービスの使用方法について説明します。これには SOAP ベース版の API 要求（外部システムが Service Catalog 内のサービス要求を作成および管理できるようにする API である RAPI 2）を実装する Web サービスが含まれます。Web サービスには、サービス要求内の提供タスクや承認タスクの管理を許可し、Service Catalog の内容の確認を行う要求なども含まれます。

WSDL

作成した要求を検証するには、SOAP ベースの Web サービスの WSDL が使用できる必要があります。WSDL は次の場所にあります。

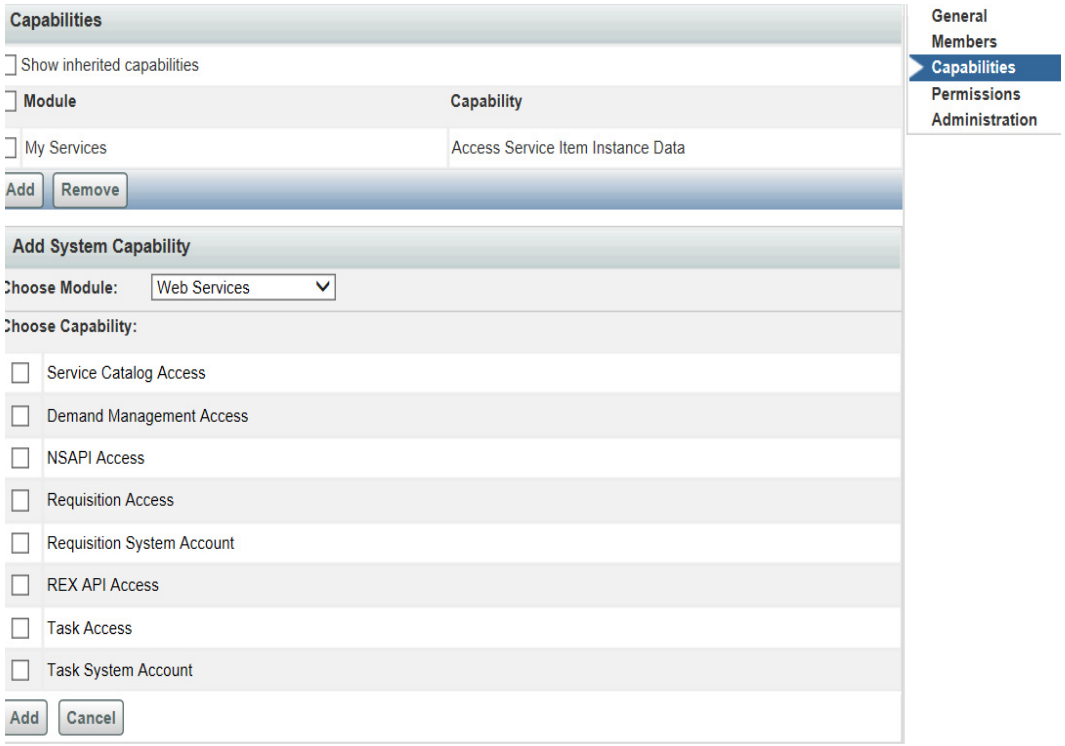
http://<ServerName>/RequestCenter/webservices/wsdl/

使用可能な WSDL について、次の表にまとめます。

WSDL	内容
AuthenticationService.wsdl	指定したユーザをサービス カタログ（Service Catalog）に対して認証する要求。
RequisitionService.wsdl	要求の送信、要求のキャンセル、またはそのステータスの確認を求める要求。
ServiceCatalog.wsdl	内部使用専用。
ServiceManagerTaskService.wsdl	承認の許可または拒否の要求、あるいは確認が実行済みであるとの通知を求める要求。

ロールと機能の設定

Web サービスを利用できるのは、Web サービス モジュールに対する適切な権限が含まれるロールを持つユーザです。事前に設定されたロールにはこのような権限は含まれていないため、管理者は Organization Designer を使用して、1 つ以上のカスタム ロールを作成する必要があります。ロールを作成したら、Web サービスの権限を追加できます。



Web サービスの権限を次に示します。

- **Service Catalog アクセス（Service Catalog Access）：**この権限を持つユーザは、Service Catalog の Web サービスにアクセスできます。
- **デマンド管理アクセス（Demand Management Access）：** この権限を持つユーザは、自分のデマンド管理 Web サービスにアクセスできます。
- **NSAPI アクセス（NSAPI Access）：**この権限を持つユーザは、NSAPI Web サービスにアクセスできます。
- 要求アクセス（Requisition Access）：この権限を持つユーザのみが、自分の RequisitionService Web サービス要求にアクセスできます。認証されたユーザとイニシエータは同一人である必要があります。同一人でない場合、対応するエラー応答がスローされます。
- 要求システムアカウント（Requisition System Account）：この権限を持つユーザは、自分だけでなく他のユーザの RequisitionService Web サービス要求にもアクセスできます。認証されたユーザとイニシエータは同一人である必要はありません。
- REX APIアクセス（REX API Access）：この権限を持つユーザは、Catalog Deployer の機能にアクセスできます。
- タスクアクセス（Task Access）：この権限を持つユーザのみが、自分の ServiceManager TaskService Web サービス要求にアクセスできます。この権限は必須です。
- タスクシステムアカウント（Task System Account）：この権限を持つユーザは、自分だけでなく他のユーザの ServiceManager TaskService Web サービス要求にもアクセスできます。認証されたユーザとイニシエータは同一人である必要はありません。

Web サービスのクライアント コードの作成

Web サービスのクライアントは、Apache の CXF や Axis などのツールを使用してコーディングできます。

Axis 2 によるクライアント コードの作成

Axis 2 を使用して Web サービスのクライアントを生成するための詳細な説明とユーザ ガイドについては、Apache の Web サイトを参照してください。

soapUI を使用して axis2 クライアントを作成する手順の概要は次のとおりです。

手順 1　Axis 2 のライブラリをダウンロードします。

手順 2　Axis 2 ライブラリの場所を soapUI の [Preferences] メニューで設定します。

手順 3　[Tools] > [Axis 2 Artifacts] を使用して、クライアント コードを作成します。

クライアント コードを作成するときには、データバインディング方式として **adb**（Axis のデフォルト バインディング）を選択する必要があります。テスト ケース オプションも作成する必要があります。

クライアント コードが生成されます。テスト ケース内にメソッドのスタブが作成されます。このオブジェクトに正確に入力する必要があります。

Apache CXF によるクライアント コードの生成

CXF を使用して Web サービス クライアントを生成するための手順については、Apache の Web サイトを参照してください。

soapUI を使用して CXF クライアントを作成するために必要な手順は、次のとおりです。

手順 1　Apache CXF のライブラリをダウンロードします。

手順 2　CXF ライブラリの場所を soapUI の [Preferences] メニューで設定します。

手順 3　[ツール（Tools）] > [Apache CXF] を使用して、クライアント コードを作成します。

クライアント コードが生成されます。対象のクラスは次のとおりです。

```
RequisitionServicePortType_RequisitionServiceHttpPort_Client.java
```

このクラスには main メソッドがあり、WSDL で定義されているすべての処理をここから起動できます。これらの処理を起動するコードは、すでに用意されています。必要な各種変数を入力する必要があります。メソッドのスタブが作成されます。必要な操作は、オブジェクトを正しく入力することだけです。

要求管理のための Web サービス

RAPI 2 要求管理によって実行できる処理を、以下の表に要約します。

要求	説明
addComment	オープン状態の要求にコメントを追加します。 要求へのコメントの追加を参照してください。
cancelRequisition	要求内のすべてのサービス要求を含め、オープン状態の要求をキャンセルします。 要求のキャンセルを参照してください。
cancelRequisitionEntry	サービス要求をキャンセルします。要求内の最後のサービス要求の場合は、要求がキャンセルされます。 要求のキャンセルを参照してください。
getOpenRequisitions	オープン状態のすべての要求のリストを入手します。 要求のリストの入手を参照してください。
getRequisitions	オープン状態のすべての要求のリストを入手します。オプションでリストの内容を制限できます。 要求のリストの入手を参照してください。
getRequisitionStatus	指定した要求のステータスを入手します。 要求のステータスの入手を参照してください。
getServiceDefinition	要求を行う対象のサービスの定義を入手します。 getServiceDefinition 応答を参照してください。
submitRequisition	新しい要求を送信します。 submitRequisition の応答の例を参照してください。

(注)　RAPI 2 では、要求の送信にグローバル OOB 設定ではなく、OOB 権限が必要です。OOB 権限の詳細については、『Cisco Prime Service Catalog Designer Guide』の「Organization Design」の章を参照してください。

Web サービスの認証

Service Catalog で公開される Web サービスは、すべて認証される必要があります。認証されていない Web サービス コールは、遮断および停止する必要があります。

Service Catalog の Web サービスによる認証は、以下の方法で実行できます。

- セッションごとの認証
- 要求ごとの認証

認証されてないユーザは、Web サービス呼び出しを正しく実行することができません。グローバル設定の [Web サービスの有効化(Enable Web Service)] をオフにすると、Service Catalog の Web サービスにアクセスできなくなります。デフォルトでは、この設定はオフになっています。

セッションごとの認証

この方法では、ユーザは最初に Authentication Web サービス コールを実行して、ユーザを認証します。その後、サーバがそのユーザに対してセッションを確立します。このセッションが有効な間、ユーザはその他の Web サービス コールを実行できます。セッションごとの認証要求は、AuthenticationService WSDL に含まれています。

認証要求のフォーマットは以下のとおりです。

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:aut="http://authentication.api.newscale.com">
  <soapenv:Header/>
  <soapenv:Body>
   <aut:authenticate>
     <aut:userName>?</aut:userName>
     <aut:password>?</aut:password>
   </aut:authenticate>
  </soapenv:Body>
</soapenv:Envelope>
```

認証時のエラー コードは、ユーザ アカウントのパスワードの期限切れや失敗したパスワード試行により、ロックされているかどうかが示されます。ユーザ アカウントのロックを解除するには、システム管理者に問い合わせてください。

(注)　また、ユーザ認証のエラー コードは、ユーザ アカウントが猶予期間であるかどうかと、ユーザ パスワードの更新期限の日付も示します。エラー コードの詳細については、REST/Web サービスのエラー メッセージを参照してください。

要求ごとの認証

この方法では、Web サービスを認証するための個別のコールはありません。ユーザは各 Web サービス コールの一部として SOAP ヘッダーに認証情報を入れて送信します。Service Catalog の Web サービスに対する認証ハンドラが、そのユーザを認証するかどうかを確認します。このユーザに対してセッションがまだ確立されていない場合は、ハンドラが認証情報を SOAP ヘッダーから読み取ります。認証情報が存在すれば、ハンドラがユーザの認証を試みます。認証情報がない場合や、有効でない場合、ハンドラはクライアントに対して該当するエラー コードおよびエラー メッセージと共に例外をスローします。

暗号化

SOAP ヘッダーに指定されるパスワードは、暗号化された形式だけを受け入れるように設定されている場合があります。暗号化パスワードを強制するには、Administration モジュールの [暗号化されたパスワードを受け入れる（Accept Encrypted Password）] 設定を有効にします。暗号化ユーティリティは、パスワードの暗号化された値を取得するサイト管理者ロールを持つユーザが使用できます。このユーティリティにアクセスするには、次のブラウザ ページを開きます。

http://<server>:<port>/RequestCenter/EncryptedPassword.jsp

Web サービスの認証メカニズム

Service Catalog で公開されている各 Web サービスには、関連付けられたシステム権限があります。認証ハンドラでも、指定したユーザが Web サービスにアクセス（または実行）できるかどうかを確認します。ユーザに適切なシステム権限があれば、ユーザは処理を続行できます。適切なシステム権限がない場合は、クライアントに対してエラー コードとエラー メッセージと共に例外をスローします。

SOAP 認証と Directory Integration との相互関係

Directory Integration がイネーブルでない場合は、SOAP 要求が発行される前に、指定したユーザが個人ディレクトリに存在している必要があります。

Directory Integration がイネーブルで、[ログイン（Login）] イベントに [個人のインポート（Import Person）] の処理が含まれている場合は、ユーザのプロファイルを入手する際に外部ディレクトリが確認の対象となり、その情報が個人ディレクトリに挿入されます。この方法を取るには、ディレクトリ情報に適切な Web サービス権限を付与するロールが含まれているか、ユーザが属する事業部（またはサービス チーム）にそのようなロールが事前に割り当てられている必要があります。したがって、使用する SOAP アカウントはデータベースに事前に入力し、これらのアカウントが要求を送信する前に、アカウントに対して適切な権限を割り当てておくことを推奨します。

Directory Integration が有効で、シングル サインオン（SSO）だけを行うように [ログイン（Login）] イベントが設定されている場合は、ディレクトリ イベントを完全にバイパスし、個人ディレクトリに対する単純認証にフォールバックするオプションがあります。デフォルトでは Web サーバへの SOAP 要求があり、SSO が成功した場合、SSO のユーザは、Web サービスのセッション ユーザになります。そのためには、SOAP ヘッダーにユーザ クレデンシャルを含めない必要があります。ただし、優先クレデンシャルが SOAP 要求ヘッダーに指定されている場合は、そのクレデンシャルを使用して、外部ディレクトリではなく、個人ディレクトリに対して認証を行います。つまり、SOAP ヘッダーのユーザ クレデンシャルの存在は、認証をローカルで行うか、または外部で行うかを制御します。

[ログイン（Login）] イベントの Directory Integration に外部認証の手順（SSO との組み合わせの有無にかかわらず）が含まれる場合、認証は必ずディレクトリ データソースに対して実行されます。

サービス定義の入手

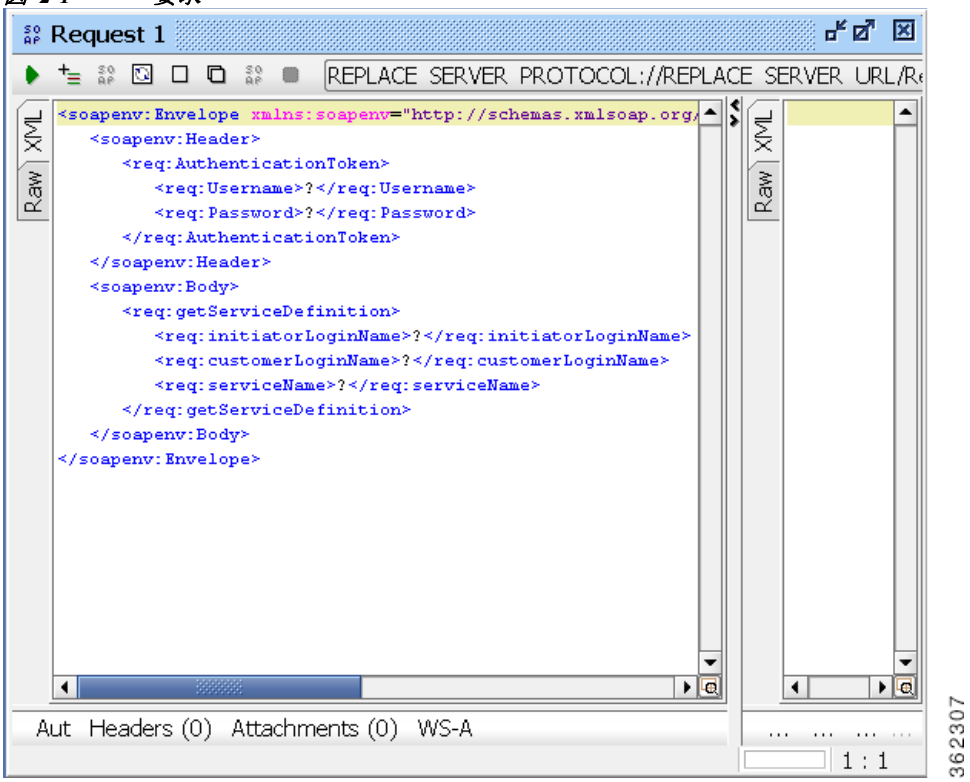
getServiceDefinition 要求は、指定したサービスが記述されたメタデータを返します。要求を送信するには、このメタデータが必要です。グリッド ディクショナリを含むサービスにこの操作を使用することは、このリリースではサポートされていません。そのようなサービスに対して操作を呼び出すと、エラーが返されます。

getServiceDefinition 要求

この要求では、その定義を必要とするサービスの名前を指定します。

soapUI で、[getServiceDefinition] ノードの下にある要求の例（Request1）を右クリックして、[要求エディタの表示（Show Request Editor）] をクリックします。生成された要求が表示されます。疑問符（?）は、値を必要とするすべての XML 要素を示します。

図 2-1　　要求



SOAP 要求にはエンドポイントを指定する必要があります。この要求のプロパティ（および上図のメニューバー）をよく見ると、エンドポイントがまだ定義されていないことがわかります。これを以下の RAPI 2 サービスのエンドポイントで置き換えます。

http://<ServerName>/RequestCenter/services/**RequisitionService**

ここで、**RequisitionService** は wsdl 名です。

次に、Request Editor のメニューバーにある [Creates a copy of this request] アイコン（ ）を使用して、この要求をコピーします。このプロトタイプの要求は参照用に残しておきます。コピーを使用して、疑問符の部分を置き換え、認証条件のほか、イニシエータやカスタマーのログイン名、対象とするサービスの名前などを入力します。

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:req="http://requisition.api.newscale.com">
  <soapenv:Header>
   <req:AuthenticationToken>
     <req:Username>admin</req:Username>
     <req:Password>admin</req:Password>
   </req:AuthenticationToken>
  </soapenv:Header>
  <soapenv:Body>
   <req:getServiceDefinition>
     <req:initiatorLoginName>admin</req:initiatorLoginName>
     <req:customerLoginName>mthurston</req:customerLoginName>
     <req:serviceName>New Standard Laptop Computer</req:serviceName>
   </req:getServiceDefinition>
  </soapenv:Body>
</soapenv:Envelope>
```

getServiceDefinition 応答

getServiceDefinition 要求を送信するには、Request Editor ウィンドウの左上にある [Submit request to specified URL] ボタン( )をクリックします。応答が [要求エディタ(Request Editor)] 内の要求の右側に表示されます。

getServiceDefinition に対する応答では、サービスについてのメタデータが返されます。以下の表に要約を示します。

表 2-3　getServiceDefinition に対する応答

XML 要素(およびドキュメントの階層)	説明
Service	
name	サービスの名前
pricingmodel	
quantity	オーダーするサービスの数量
version	サービスのバージョン番号
Dictionaries	
Dictionary	各サービスには 1 つ以上のディクショナリが含まれます。
name	ディクショナリの名前
readable	オーダー時点で、Service Designer のアクティブ フォーム コンポーネントにあるアクセス コントロールに従って、ディクショナリが読み取り可能であれば true、そうでない場合は false。
writable	オーダー時点で、Service Designer のアクティブ フォーム コンポーネントにあるアクセス コントロールに従って、ディクショナリが書き込み可能であれば true、そうでない場合は false。
Fields >	
DictionaryField	各ディクショナリには 1 つ以上のフィールドが含まれます。
canSelectMultiple	このフィールドに対して複数の値を選択できるかどうかを示します。
defaultValue	フィールドのデフォルト値
fieldDataType	フィールドのデータ型(数値、日付など)
fieldName	フィールドの名前
inputType	フィールドの html の入力タイプ
label	フィールドのラベル
mandatory	このフィールドが必須の場合は true、必須でない場合は false。
maxLength	フィールドの最大長
selectableValues	フィールドに対して選択可能な値

各ディクショナリと、ディクショナリ内の各フィールドが記述されています。オーダー時点でディクショナリに対して指定されているアクセス コントロールは、正確な形式の submitRequisition 要求を記述するのに不可欠です。オーダー時点でカスタマーから読み込みおよび書き込み可能なディクショナリだけが応答に取り込まれます。これは、submitRequisition 要求に取り込む必要があります。

```
  <name>Customer_Information</name>
  <readable>true</readable>
  <writable>true</writable>
</Dictionary>
```

「New Standard Laptop Computer」に対する完全な getServiceDefinitionResponse は、「要求と応答の例」に記載してあります。

要求の送信

submitRequisition 要求を送信する前に getServiceDefinition 要求を実行する必要はありません。ただし、getServiceDefinition では、現在のサービスのバージョンに関する有効な submitRequisition メッセージの記述に必要な情報が返されます。

- サービスの現在のバージョンは必須です。バージョン番号は、サービス定義自体、または含まれているアクティブ フォーム コンポーネント、またはディクショナリが更新されたときに、必ず増分されます。
- getServiceDefinition 要求は、どのフィールドが必須であるか指定します。送信要求にはすべての必須フィールドのデータが含まれる必要があります。
- 送信する要求には、すべての必須ディクショナリとフィールドがリストされている必要があります。ディクショナリ、またはそれぞれのディクショナリ内のフィールドの表示順序は任意です。
- ブラウザ側でトリガされるように設定されたフォーム ルールは、Web サービスに影響しません。フォーム ルールによって入力される必要がある選択リストやデフォルト値がある場合、それらのルールが検証およびワークフローの開始前に実行されるように、After Submission イベントと関連付ける必要があります。
- getServiceDefinition 要求では、各フィールドに指定されているデフォルト値も返されます。これには、カスタマーまたはイニシエータ情報のための解決済みの Lightweight 名前空間などが含まれます。これらの値は通常は必須であり、submitRequisition 要求で指定する必要があります。
- サービスの定義にオプション（単一選択、複数選択、およびオプション ボタン）を伴うフィールドがあり、それらのオプションがアクティブ フォーム コンポーネントの [Display Options]（HTML Representation）ページを使用して定義されている場合、getServiceDefinition 要求を使用して、そのサービスに対する要求を送信できます。オプションがデータ取得ルールによって指定されているときも、サービス要求は送信できますが、そのフィールドの値が有効なオプションであるかどうかの確認は、送信側のプログラムの責任になります。

submitRequisition 要求では、My Services から要求が送信されるときに発生するオーダーの時間が、基本的に省略されます。条件付きルール、データ取得ルール、または ISF は、送信された要求と一緒には実行されません。したがって、このような仕組みがディクショナリ フィールドの値の指定や評価の実行に使用されている場合は、これらの値を指定するための別の手段を探す必要があります。グリッド ディクショナリを含むサービスにこの操作を使用することは、このリリースではサポートされていません。そのようなサービスに対して操作を呼び出すと、エラーが返されます。

submitRequisition 要求

オーダー時点で表示または編集が可能なディクショナリは、submitRequisition 要求の <section> ノードとして組み込む必要があります。すべての必須フィールドとその値を指定する必要があります。オプションのフィールドには値を指定する必要はありません（しかし、soapUI によって挿入された疑問符は削除する必要があります）。ディクショナリの順序とフィールドの順序は、サービス定義での順序と一致する必要はありませんが、フィールドは対応するディクショナリノードの下にある必要があります。

表 2-4　submitRequisition 要求

XML 要素（およびドキュメントの階層）	説明
initiatorLoginName	イニシエータのログイン名
customerLoginName	カスタマーのログイン名
serviceRequests > ServiceRequest	複数のサービス要求を指定できます。
name	サービスの名前
quantity	オーダーされるサービスの数量
version	サービスのバージョン
Sections > Section	各サービスには複数のディクショナリ（セクション）を指定できます。
name	ディクショナリの名前
Fields > Field	各ディクショナリには複数のフィールドを指定できます。
name	フィールドの名前
value > string	このフィールドに設定する値

たとえば、ディクショナリ RC_ServiceLocation 内の ZipCode フィールドの値を「07201」に設定する XML は、以下のようになります。

```
<req:Section>
.   .   .
    <req:fields>
.   .   .
        <req:Field>
            <req:name>ZipCode</req:name>
            <req:value>
                <req:string 07201/>
            </req:value>
        </req:Field>
    </req:fields>
    <req:name>RC_ServiceLocation</req:name>
</req:Section>
```

submitrequisition の応答

要求を送信する要求が成功すると、応答には作成された要求の要求 ID と、その要求の他の属性が含まれます。

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soap:Body>
   <ns1:submitRequisitionResponse xmlns:ns1="http://requisition.api.newscale.com">
```

```
     <ns1:submitRequisitionResult ns1:customer="admin admin"
         ns1:dueDate="2009-05-08T16:14:26.267-07:00"
         ns1:requisitionId="186"
         ns1:initiator="admin admin"
         ns1:startedDate="2009-04-30T18:14:26.110-07:00"
         ns1:status="Ongoing"/>
   </ns1:submitRequisitionResponse>
  </soap:Body>
</soap:Envelope>
```

submitRequisitionResult 応答の属性を、次の表に要約します。

表 2-5　submitrequisition の応答

XML 要素	説明
submitRequisitionResponse > submitRequisitionResult	この応答には、サービス要求内にあるサービスと同じ数のエントリが含まれます。
customer	要求のカスタマー名
dueDate	要求の期限
requisitionId	要求の要求 ID
initiator	要求のイニシエータ
startedDate	要求が開始された日付
status	要求のステータス

要求が失敗すると、エラー メッセージが返されます。発生する可能性のあるエラーは、「付録 B: RAPI のエラー メッセージ」に記載されています。エラー メッセージは、次の例に示すように、必ず「SOAP fault」という書式で表されます。

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soap:Body>
   <soap:Fault>
     <faultcode>soap:Server</faultcode>
     <faultstring>The version specified in the request does not match the version in the
database for service 'New Standard Laptop Computer'. Please get the latest service
definition.</faultstring>
     <detail>
      <RequisitionFault xmlns="http://requisition.api.newscale.com">
        <errorCode>REQ_0018</errorCode>
        <errorMessage>The version specified in the request does not match the version in
the database for service 'New Standard Laptop Computer'. Please get the latest service
definition.</errorMessage>
      </RequisitionFault>
     </detail>
   </soap:Fault>
  </soap:Body>
</soap:Envelope>
```

要求のリストの入手

getRequisitions と getOpenRequisitions の処理では、オープン状態の要求についての情報を返します。これらは要求に含めることのできる引数が異なります。

これらの処理は、要求の管理に役立つ場合があります。たとえば、オープンな状態の要求のリストが返されたら、その中から、ユーザ定義の期限を超えた特定のタイプ（特定のサービスについて）を調べることができます。

getOpenRequisitions 要求

getOpenRequisitions は、指定した最大数まで、オープン状態の要求をすべて返します。要求は、要求 ID の降順で返されます。この要求は、一部の古い Service Catalog 統合ポイントとの下位互換性のためだけにサポートされているため、Web サービスで使用しないでください。

getRequisitions 要求

getRequisitions は、指定した最大数まで、要求をすべて返します。返される要求の表示タイプやステータスを指定することもできます。この要求は、一部の古い Service Catalog 統合ポイントとの下位互換性のためだけにサポートされているため、Web サービスで使用しないでください。

要求のステータスの入手

getRequisitionStatus の処理では、指定した要求の承認やタスク プランのステータスに関する情報を返します。詳細度は、My Services ユーザが提供計画を表示した場合の内容と同程度です。

図 2-2　getRequisitionStatus

Delivery Process

	Process Milestone	Due Date	Completed On	Status
✔	Service Group Review	12/07/2011 10:00 AM	12/07/2011 3:06 AM	Completed
✔	Service Group Authorization	12/07/2011 11:00 AM	12/07/2011 3:07 AM	Completed
	Delivery project for Testemail_Order_Mobile Device_Service_at_doorstep	12/08/2011 11:00 AM		In Progress

362308

getRequisitionStatus 要求

この要求は、要求の現在のステータスに関する情報を返します。

表 2-6　getRequisitionStatus

XML 要素（およびドキュメントの階層）	説明
loginUserName	情報を要求するユーザの名前。このユーザは要求を表示する権限を持っている必要があります。
requisitionid	紹介される要求の ID。

GetRequisitionStatus 応答

要求を取得する要求が成功すると、その応答には要求に関する情報が含まれます。

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soap:Body>
  </soap:Body>
</soap:Envelope>
```

get*RequisitionsResult 応答の属性を、以下の表に要約します。

表 2-7　GetRequisitionStatus 応答

XML 要素	説明
getRequisitionStatusResponse > get*RequisitionsResult	
RequisitionEntryStatuses > RequisitionEntryStatus	要求内の各サービスに対する 1 つのステータス ブロック
itemNumber	要求内でサービスに割り当てられた連続番号
quantity	サービス オーダーの数
requisitionEntryId	サービスの要求エントリ ID
serviceName	サービスの名前
status	要求エントリの現在のステータス
requisitionStepStatuses > RequisitionStepStatus	サービスの提供計画に設定されている各時点についての StepStatus
dueDate	現在の承認、確認、またはタスクの期限の日付
name	提供計画での時点。(「Service Group Authorization」または「Delivery project for <service name>」など)
stepStatus	タスクのステータス。(「In Progress」、「Pending」、「Completed」など)

要求へのコメントの追加

addComments によって、指定した要求にユーザのコメントを追加します。

addComments 要求

この要求では、指定したコメントを指定した要求に追加します。指定されたユーザは、要求にアクセスする権限を持っている必要があります。

表 2-8　addComments 要求

XML 要素(およびドキュメントの階層)	説明
loginUserName	コメントを追加するユーザの名前
requisitionid	対象となる要求の ID
commentText	ユーザ コメントのテキスト

要求のキャンセル

cancelRequisition は、指定したサービス要求をキャンセルするために使用します。要求に含まれるすべてのサービスがキャンセルされます。

cancelrequisitionentry は、サービス要求内にある指定したサービス(要求エントリ)をキャンセルします。要求内の唯一(または最後)のサービスの場合は、要求がキャンセルされます。そうでない場合は、ステータスは変更されないままになります。

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:req=>
  <soapenv:Header>
   <req:AuthenticationToken>
     <req:Username>admin</req:Username>
```

```
      <req:Password>admin</req:Password>
    </req:AuthenticationToken>
  </soapenv:Header>
  <soapenv:Body>
   <req:cancelRequisition>
     <req:loginUserName>ltierstein</req:loginUserName>
     <req:requisitionId>99</req:requisitionId>
   </req:cancelRequisition>
  </soapenv:Body>
</soapenv:Envelope>
The response is shown below (with formatting added for clarity):
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soap:Body>
   <ns1:cancelRequisitionResponse xmlns:ns1="http://requisition.api.newscale.com">
     <ns1:cancelRequisitionResult
       ns1:closedDate="2009-06-02T12:04:32.837-07:00"
       ns1:customer="Leslie Tierstein"
       ns1:dueDate="2009-04-03T15:00:00-07:00"
       ns1:id="99"
       ns1:initiator="Leslie Tierstein"
       ns1:startedDate="2009-04-03T09:55:54.843-07:00"
       ns1:status="Cancelled"/>
   </ns1:cancelRequisitionResponse>
  </soap:Body>
</soap:Envelope>
```

タスク管理用の Web サービス

概要

タスク管理用の Web サービスから実行できる処理を、以下の表に要約します。

表 2-9　タスク管理用の Web サービス

要求	説明
approveTask	承認/許可の許可
getAuthorizations	承認の入手
getAuthorizationsForUser	指定したユーザの承認の入手
getMyAuthorizations	指定した人に割り当てられている承認の入手
rejectSelectedReqEntry	指定したサービス（要求エントリ）の拒否
rejectTask	承認/許可の拒否
reviewTask	「確認」タスクを確認済みとしてマーク

承認のリストの入手

getAuthorizations および getMyAuthorizations 操作は、「In Progress」である承認についての情報を返します。これらは要求に含めることのできる引数が異なります。

要求サービスの操作では、操作の対象となる Web Services Administrative ユーザ（SOAP ヘッダーで指定）や Service Catalog ユーザごとに個別のプロビジョニングを行いますが、それとは異なり、これらのタスク サービスの操作では、ユーザを 1 名のみ SOAP ヘッダーに指定できます。したがって、承認を入手または処理されるユーザは、Web サービス モジュールの Task Access 権限を持っている必要があります。手順は次のとおりです。

- この権限が含まれるロールを作成します。承認を実行する権限は My Services Professional ロールに含まれているので、このロールの子を作成します。

- Web サービスを使用して、自身の承認を確認または処理する必要のあるユーザに、このロールを割り当てます(追加するか、または My Services Professional の代わりとします)。

getMyAuthorizations 要求

getMyAuthorizations は、すべてのオープン状態の要求を、Service Catalog のクレデンシャルが SOAP ヘッダーに指定されているユーザに対して、指定されている最大数まで返します。要求は、要求 ID の降順で返されます。この要求は、JSR168 準拠の承認ポートレットでの使用だけがサポートされており、Web サービスでは使用できません。

getAuthorizations 要求

getAuthorizations は、リストで指定した承認から開始して、特定の承認の最大数まで、すべての承認を返します。返される要求の表示タイプやステータスを指定することもできます。この要求は、JSR168 準拠の承認ポートレットでの使用だけがサポートされており、Web サービスでは使用できません。

getAuthorizationsForUser 要求(内部用途のみ、非サポート)

これは上記の getAuthorizations とよく似ていますが、承認を入手する対象のユーザを指定できる userLoginName パラメータが追加されています。

getAuthorizationsForUser SOAP 要求の例

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:smt="http://smtask.api.newscale.com">   <soapenv:Header>
      <smt:AuthenticationToken>
         <smt:Username>admin</smt:Username>
         <smt:Password>admin</smt:Password>
      </smt:AuthenticationToken>
   </soapenv:Header>
   <soapenv:Body>
      <smt:getAuthorizationsForUser>
         <smt:userLoginName>qreviewer</smt:userLoginName>
         <smt:startRow>0</smt:startRow>
         <smt:numberOfRows>5</smt:numberOfRows>
         <smt:status>1</smt:status>
         <smt:viewType>2</smt:viewType>
      </smt:getAuthorizationsForUser>
   </soapenv:Body>
</soapenv:Envelope>
```

getAuthorizations 要求および getAuthorizationsForUser 要求に役立つパラメータ

表 2-10　getAuthorizations および getAuthorizationsForUser 要求のパラメータと値

パラメータ	値
Status	処理中:1
	キャンセル済み:2
	承認済み:3
	拒否済み:4
	確認済み:5
	すべて:6
ViewType	自身の承認:1
	自身の割り当ておよび非割り当て:2

承認の許可または拒否

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:smt="http://smtask.api.newscale.com">
  <soapenv:Header>
   <smt:AuthenticationToken>
     <!--Optional:-->
     <smt:Username>admin</smt:Username>
     <!--Optional:-->
     <smt:Password>admin</smt:Password>
   </smt:AuthenticationToken>
  </soapenv:Header>
  <soapenv:Body>
   <smt:approveTask>
     <smt:approverLoginName>maria</smt:approverLoginName>
     <smt:taskID>281</smt:taskID>
   </smt:approveTask>
  </soapenv:Body>
</soapenv:Envelope>
```

応答は以下に示すとおりです(読みやすくするために書式化してあります)。

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soap:Body>
   <ns1:approveTaskResponse xmlns:ns1="http://smtask.api.newscale.com">
     <ns1:approveTaskResult
       ns1:actionID="5"
       ns1:requisitionId="103"
       ns1:status="approved"
       ns1:taskName="Computer Memory - Upgrade - APPROVAL NEEDED"/>
   </ns1:approveTaskResponse>
  </soap:Body>
</soap:Envelope>
```

要求と応答の例

getServiceDefinition 応答

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soap:Body>
   <ns1:getServiceDefinitionResponse xmlns:ns1=>
     <ns1:getServiceDefinitionResult>
      <dictionaries>
        <Dictionary>
         <fields>
           <DictionaryField>
            <canSelectMultiple>false</canSelectMultiple>
            <defaultValue>
              <string/>
            </defaultValue>
            <fieldDataType>Text</fieldDataType>
            <fieldName>ModelNumber</fieldName>
            <inputType>text</inputType>
            <label>Model Number</label>
            <mandatory>false</mandatory>
            <maxLength>50</maxLength>
            <selectableValues>
              <string/>
```

```
      </selectableValues>
     </DictionaryField>
     <DictionaryField>
      <canSelectMultiple>false</canSelectMultiple>
      <defaultValue>
        <string/>
      </defaultValue>
      <fieldDataType>Text</fieldDataType>
      <fieldName>AssetTag</fieldName>
      <inputType>text</inputType>
      <label>Asset Tag</label>
      <mandatory>false</mandatory>
      <maxLength>50</maxLength>
      <selectableValues>
        <string/>
      </selectableValues>
     </DictionaryField>
   </fields>
   <name>NewLaptop</name>
   <readable>true</readable>
   <writable>true</writable>
  </Dictionary>
  <Dictionary>
   <fields>
     <DictionaryField>
      <canSelectMultiple>false</canSelectMultiple>
      <defaultValue>
        <string>admin</string>
      </defaultValue>
      <fieldDataType>Text</fieldDataType>
      <fieldName>First_Name</fieldName>
      <inputType>text</inputType>
      <label>First Name</label>
      <mandatory>false</mandatory>
      <maxLength>100</maxLength>
      <selectableValues>
        <string/>
      </selectableValues>
     </DictionaryField>
     <DictionaryField>
      <canSelectMultiple>false</canSelectMultiple>
      <defaultValue>
        <string>admin</string>
      </defaultValue>
      <fieldDataType>Text</fieldDataType>
      <fieldName>Last_Name</fieldName>
      <inputType>text</inputType>
      <label>Last Name</label>
      <mandatory>false</mandatory>
      <maxLength>100</maxLength>
      <selectableValues>
        <string/>
      </selectableValues>
     </DictionaryField>
     <DictionaryField>
      <canSelectMultiple>false</canSelectMultiple>
      <defaultValue>
        <string>admin</string>
      </defaultValue>
      <fieldDataType>Text</fieldDataType>
      <fieldName>Login_ID</fieldName>
      <inputType>hidden</inputType>
      <label>Login ID</label>
      <mandatory>false</mandatory>
```

```
 <maxLength>200</maxLength>
 <selectableValues>
   <string/>
 </selectableValues>
</DictionaryField>
<DictionaryField>
 <canSelectMultiple>false</canSelectMultiple>
 <defaultValue>
   <string/>
 </defaultValue>
 <fieldDataType>Text</fieldDataType>
 <fieldName>Personal_Identification</fieldName>
 <inputType>text</inputType>
 <label>Personal_Identification</label>
 <mandatory>false</mandatory>
 <maxLength>510</maxLength>
 <selectableValues>
   <string/>
 </selectableValues>
</DictionaryField>
<DictionaryField>
 <canSelectMultiple>false</canSelectMultiple>
 <defaultValue>
   <string>ed @cisco.com</string>
 </defaultValue>
 <fieldDataType>Text</fieldDataType>
 <fieldName>Email_Address</fieldName>
 <inputType>text</inputType>
 <label>Email Address</label>
 <mandatory>false</mandatory>
 <maxLength>1024</maxLength>
 <selectableValues>
   <string/>
 </selectableValues>
</DictionaryField>
<DictionaryField>
 <canSelectMultiple>false</canSelectMultiple>
 <defaultValue>
   <string>Site Administration</string>
 </defaultValue>
 <fieldDataType>Text</fieldDataType>
 <fieldName>Home_Organizational_Unit</fieldName>
 <inputType>text</inputType>
 <label>Department</label>
 <mandatory>false</mandatory>
 <maxLength>50</maxLength>
 <selectableValues>
   <string/>
 </selectableValues>
</DictionaryField>
<DictionaryField>
 <canSelectMultiple>false</canSelectMultiple>
 <defaultValue>
   <string/>
 </defaultValue>
 <fieldDataType>Text</fieldDataType>
 <fieldName>Company_State</fieldName>
 <inputType>text</inputType>
 <label>State</label>
 <mandatory>false</mandatory>
 <maxLength>100</maxLength>
 <selectableValues>
   <string/>
 </selectableValues>
```

```
  </DictionaryField>
  <DictionaryField>
   <canSelectMultiple>false</canSelectMultiple>
   <defaultValue>
     <string/>
   </defaultValue>
   <fieldDataType>Text</fieldDataType>
   <fieldName>Supervisor</fieldName>
   <inputType>hidden</inputType>
   <label>Supervisor</label>
   <mandatory>false</mandatory>
   <maxLength>100</maxLength>
   <selectableValues>
     <string/>
   </selectableValues>
  </DictionaryField>
  <DictionaryField>
   <canSelectMultiple>false</canSelectMultiple>
   <defaultValue>
     <string/>
   </defaultValue>
   <fieldDataType>Text</fieldDataType>
   <fieldName>Supervisor_Email</fieldName>
   <inputType>hidden</inputType>
   <label>Supervisor Email</label>
   <mandatory>false</mandatory>
   <maxLength>1024</maxLength>
   <selectableValues>
     <string/>
   </selectableValues>
  </DictionaryField>
  <DictionaryField>
   <canSelectMultiple>false</canSelectMultiple>
   <defaultValue>
     <string/>
   </defaultValue>
   <fieldDataType>Text</fieldDataType>
   <fieldName>Custom_1</fieldName>
   <inputType>text</inputType>
   <label>Custom_1</label>
   <mandatory>false</mandatory>
   <maxLength>200</maxLength>
   <selectableValues>
     <string/>
   </selectableValues>
  </DictionaryField>
  <DictionaryField>
   <canSelectMultiple>false</canSelectMultiple>
   <defaultValue>
     <string/>
   </defaultValue>
   <fieldDataType>Text</fieldDataType>
   <fieldName>Custom_2</fieldName>
   <inputType>text</inputType>
   <label>Custom_2</label>
   <mandatory>false</mandatory>
   <maxLength>200</maxLength>
   <selectableValues>
     <string/>
   </selectableValues>
  </DictionaryField>
 </fields>
 <name>Customer_Information</name>
 <readable>true</readable>
```

```
 <writable>true</writable>
</Dictionary>
<Dictionary>
 <fields>
   <DictionaryField>
    <canSelectMultiple>false</canSelectMultiple>
    <defaultValue>
      <string>admin</string>
    </defaultValue>
    <fieldDataType>Text</fieldDataType>
    <fieldName>First_Name</fieldName>
    <inputType>text</inputType>
    <label>First Name</label>
    <mandatory>false</mandatory>
    <maxLength>100</maxLength>
    <selectableValues>
      <string/>
    </selectableValues>
   </DictionaryField>
   <DictionaryField>
    <canSelectMultiple>false</canSelectMultiple>
    <defaultValue>
      <string>admin</string>
    </defaultValue>
    <fieldDataType>Text</fieldDataType>
    <fieldName>Last_Name</fieldName>
    <inputType>text</inputType>
    <label>Last Name</label>
    <mandatory>false</mandatory>
    <maxLength>100</maxLength>
    <selectableValues>
      <string/>
    </selectableValues>
   </DictionaryField>
   <DictionaryField>
    <canSelectMultiple>false</canSelectMultiple>
    <defaultValue>
      <string>admin</string>
    </defaultValue>
    <fieldDataType>Text</fieldDataType>
    <fieldName>Login_ID</fieldName>
    <inputType>text</inputType>
    <label>Login ID</label>
    <mandatory>false</mandatory>
    <maxLength>200</maxLength>
    <selectableValues>
      <string/>
    </selectableValues>
   </DictionaryField>
   <DictionaryField>
    <canSelectMultiple>false</canSelectMultiple>
    <defaultValue>
      <string/>
    </defaultValue>
    <fieldDataType>Text</fieldDataType>
    <fieldName>Personal_Identification</fieldName>
    <inputType>hidden</inputType>
    <label>Personal Identification</label>
    <mandatory>false</mandatory>
    <maxLength>510</maxLength>
    <selectableValues>
      <string/>
    </selectableValues>
   </DictionaryField>
```

```
    <DictionaryField>
     <canSelectMultiple>false</canSelectMultiple>
     <defaultValue>
       <string>ed @cisco.com</string>
     </defaultValue>
     <fieldDataType>Text</fieldDataType>
     <fieldName>Email_Address</fieldName>
     <inputType>text</inputType>
     <label>Email Address</label>
     <mandatory>false</mandatory>
     <maxLength>1024</maxLength>
     <selectableValues>
       <string/>
     </selectableValues>
    </DictionaryField>
    <DictionaryField>
     <canSelectMultiple>false</canSelectMultiple>
     <defaultValue>
       <string>Site Administration</string>
     </defaultValue>
     <fieldDataType>Text</fieldDataType>
     <fieldName>Home_Organizational_Unit</fieldName>
     <inputType>text</inputType>
     <label>Department</label>
     <mandatory>false</mandatory>
     <maxLength>50</maxLength>
     <selectableValues>
       <string/>
     </selectableValues>
    </DictionaryField>
  </fields>
  <name>Initiator_Information</name>
  <readable>false</readable>
  <writable>false</writable>
 </Dictionary>
 <Dictionary>
  <fields>
    <DictionaryField>
     <canSelectMultiple>false</canSelectMultiple>
     <defaultValue>
       <string>Yes</string>
     </defaultValue>
     <fieldDataType>Boolean</fieldDataType>
     <fieldName>PerformWork</fieldName>
     <inputType>radio</inputType>
     <label>Will work be performed at the customer location?</label>
     <mandatory>false</mandatory>
     <maxLength>0</maxLength>
     <selectableValues>
       <string>Yes</string>
       <string>No</string>
     </selectableValues>
    </DictionaryField>
  </fields>
  <name>RC_PerformWork</name>
  <readable>true</readable>
  <writable>true</writable>
 </Dictionary>
 <Dictionary>
  <fields>
    <DictionaryField>
     <canSelectMultiple>false</canSelectMultiple>
     <defaultValue>
       <string/>
```

```
 </defaultValue>
 <fieldDataType>Text</fieldDataType>
 <fieldName>Street1</fieldName>
 <inputType>text</inputType>
 <label>Street</label>
 <mandatory>false</mandatory>
 <maxLength>50</maxLength>
 <selectableValues>
   <string/>
 </selectableValues>
</DictionaryField>
<DictionaryField>
 <canSelectMultiple>false</canSelectMultiple>
 <defaultValue>
   <string/>
 </defaultValue>
 <fieldDataType>Text</fieldDataType>
 <fieldName>Street2</fieldName>
 <inputType>hidden</inputType>
 <label>Street2</label>
 <mandatory>false</mandatory>
 <maxLength>50</maxLength>
 <selectableValues>
   <string/>
 </selectableValues>
</DictionaryField>
<DictionaryField>
 <canSelectMultiple>false</canSelectMultiple>
 <defaultValue>
   <string/>
 </defaultValue>
 <fieldDataType>Text</fieldDataType>
 <fieldName>Floor</fieldName>
 <inputType>hidden</inputType>
 <label>Floor</label>
 <mandatory>false</mandatory>
 <maxLength>50</maxLength>
 <selectableValues>
   <string/>
 </selectableValues>
</DictionaryField>
<DictionaryField>
 <canSelectMultiple>false</canSelectMultiple>
 <defaultValue>
   <string/>
 </defaultValue>
 <fieldDataType>Text</fieldDataType>
 <fieldName>OfficeCubeRoom</fieldName>
 <inputType>text</inputType>
 <label>OfficeCubeRoom</label>
 <mandatory>false</mandatory>
 <maxLength>50</maxLength>
 <selectableValues>
   <string/>
 </selectableValues>
</DictionaryField>
<DictionaryField>
 <canSelectMultiple>false</canSelectMultiple>
 <defaultValue>
   <string/>
 </defaultValue>
 <fieldDataType>Text</fieldDataType>
 <fieldName>Building</fieldName>
 <inputType>hidden</inputType>
```

```
 <label>Building</label>
 <mandatory>false</mandatory>
 <maxLength>50</maxLength>
 <selectableValues>
   <string/>
 </selectableValues>
</DictionaryField>
<DictionaryField>
 <canSelectMultiple>false</canSelectMultiple>
 <defaultValue>
   <string/>
 </defaultValue>
 <fieldDataType>Text</fieldDataType>
 <fieldName>City</fieldName>
 <inputType>text</inputType>
 <label>City</label>
 <mandatory>false</mandatory>
 <maxLength>50</maxLength>
 <selectableValues>
   <string/>
 </selectableValues>
</DictionaryField>
<DictionaryField>
 <canSelectMultiple>false</canSelectMultiple>
 <defaultValue>
   <string/>
 </defaultValue>
 <fieldDataType>Text</fieldDataType>
 <fieldName>State</fieldName>
 <inputType>text</inputType>
 <label>State</label>
 <mandatory>false</mandatory>
 <maxLength>50</maxLength>
 <selectableValues>
   <string/>
 </selectableValues>
</DictionaryField>
<DictionaryField>
 <canSelectMultiple>false</canSelectMultiple>
 <defaultValue>
   <string/>
 </defaultValue>
 <fieldDataType>Text</fieldDataType>
 <fieldName>PostalCode</fieldName>
 <inputType>text</inputType>
 <label>Zip Code</label>
 <mandatory>false</mandatory>
 <maxLength>50</maxLength>
 <selectableValues>
   <string/>
 </selectableValues>
</DictionaryField>
<DictionaryField>
 <canSelectMultiple>false</canSelectMultiple>
 <defaultValue>
   <string/>
 </defaultValue>
 <fieldDataType>Text</fieldDataType>
 <fieldName>Country</fieldName>
 <inputType>hidden</inputType>
 <label>Country</label>
 <mandatory>false</mandatory>
 <maxLength>50</maxLength>
 <selectableValues>
```

```
    <string/>
  </selectableValues>
 </DictionaryField>
 <DictionaryField>
  <canSelectMultiple>false</canSelectMultiple>
  <defaultValue>
    <string/>
  </defaultValue>
  <fieldDataType>Text</fieldDataType>
  <fieldName>MailStop</fieldName>
  <inputType>hidden</inputType>
  <label>MailStop</label>
  <mandatory>false</mandatory>
  <maxLength>50</maxLength>
  <selectableValues>
    <string/>
  </selectableValues>
 </DictionaryField>
 <DictionaryField>
  <canSelectMultiple>false</canSelectMultiple>
  <defaultValue>
    <string/>
  </defaultValue>
  <fieldDataType>Text</fieldDataType>
  <fieldName>Region</fieldName>
  <inputType>hidden</inputType>
  <label>Region</label>
  <mandatory>false</mandatory>
  <maxLength>50</maxLength>
  <selectableValues>
    <string/>
  </selectableValues>
 </DictionaryField>
 <DictionaryField>
  <canSelectMultiple>false</canSelectMultiple>
  <defaultValue>
    <string/>
  </defaultValue>
  <fieldDataType>Text</fieldDataType>
  <fieldName>District</fieldName>
  <inputType>hidden</inputType>
  <label>District</label>
  <mandatory>false</mandatory>
  <maxLength>50</maxLength>
  <selectableValues>
    <string/>
  </selectableValues>
 </DictionaryField>
 <DictionaryField>
  <canSelectMultiple>false</canSelectMultiple>
  <defaultValue>
    <string/>
  </defaultValue>
  <fieldDataType>Text</fieldDataType>
  <fieldName>LocationName</fieldName>
  <inputType>hidden</inputType>
  <label>LocationName</label>
  <mandatory>false</mandatory>
  <maxLength>50</maxLength>
  <selectableValues>
    <string/>
  </selectableValues>
 </DictionaryField>
 <DictionaryField>
```

```
    <canSelectMultiple>false</canSelectMultiple>
    <defaultValue>
      <string/>
    </defaultValue>
    <fieldDataType>Text</fieldDataType>
    <fieldName>LocationCode</fieldName>
    <inputType>hidden</inputType>
    <label>LocationCode</label>
    <mandatory>false</mandatory>
    <maxLength>50</maxLength>
    <selectableValues>
      <string/>
    </selectableValues>
   </DictionaryField>
 </fields>
 <name>RC_RequestorLocation</name>
 <readable>true</readable>
 <writable>true</writable>
</Dictionary>
<Dictionary>
 <fields>
   <DictionaryField>
    <canSelectMultiple>false</canSelectMultiple>
    <defaultValue>
      <string/>
    </defaultValue>
    <fieldDataType>Text</fieldDataType>
    <fieldName>Street</fieldName>
    <inputType>text</inputType>
    <label>Street</label>
    <mandatory>false</mandatory>
    <maxLength>40</maxLength>
    <selectableValues>
      <string/>
    </selectableValues>
   </DictionaryField>
   <DictionaryField>
    <canSelectMultiple>false</canSelectMultiple>
    <defaultValue>
      <string/>
    </defaultValue>
    <fieldDataType>Text</fieldDataType>
    <fieldName>OfficeCubeRoom</fieldName>
    <inputType>text</inputType>
    <label>OfficeCubeRoom</label>
    <mandatory>false</mandatory>
    <maxLength>40</maxLength>
    <selectableValues>
      <string/>
    </selectableValues>
   </DictionaryField>
   <DictionaryField>
    <canSelectMultiple>false</canSelectMultiple>
    <defaultValue>
      <string/>
    </defaultValue>
    <fieldDataType>Text</fieldDataType>
    <fieldName>State</fieldName>
    <inputType>text</inputType>
    <label>State</label>
    <mandatory>false</mandatory>
    <maxLength>40</maxLength>
    <selectableValues>
      <string/>
```

```
    </selectableValues>
   </DictionaryField>
   <DictionaryField>
    <canSelectMultiple>false</canSelectMultiple>
    <defaultValue>
      <string/>
    </defaultValue>
    <fieldDataType>Text</fieldDataType>
    <fieldName>City</fieldName>
    <inputType>text</inputType>
    <label>City</label>
    <mandatory>false</mandatory>
    <maxLength>40</maxLength>
    <selectableValues>
      <string/>
    </selectableValues>
   </DictionaryField>
   <DictionaryField>
    <canSelectMultiple>false</canSelectMultiple>
    <defaultValue>
      <string/>
    </defaultValue>
    <fieldDataType>Text</fieldDataType>
    <fieldName>BuildingName</fieldName>
    <inputType>hidden</inputType>
    <label>BuildingName</label>
    <mandatory>false</mandatory>
    <maxLength>40</maxLength>
    <selectableValues>
      <string/>
    </selectableValues>
   </DictionaryField>
   <DictionaryField>
    <canSelectMultiple>false</canSelectMultiple>
    <defaultValue>
      <string/>
    </defaultValue>
    <fieldDataType>Text</fieldDataType>
    <fieldName>Floor</fieldName>
    <inputType>hidden</inputType>
    <label>Floor</label>
    <mandatory>false</mandatory>
    <maxLength>40</maxLength>
    <selectableValues>
      <string/>
    </selectableValues>
   </DictionaryField>
   <DictionaryField>
    <canSelectMultiple>false</canSelectMultiple>
    <defaultValue>
      <string/>
    </defaultValue>
    <fieldDataType>Text</fieldDataType>
    <fieldName>ZipCode</fieldName>
    <inputType>text</inputType>
    <label>Zip Code</label>
    <mandatory>false</mandatory>
    <maxLength>15</maxLength>
    <selectableValues>
      <string/>
    </selectableValues>
   </DictionaryField>
  </fields>
  <name>RC_ServiceLocation</name>
```

```
            <readable>true</readable>
            <writable>true</writable>
           </Dictionary>
         </dictionaries>
         <estimatedpriceperunit>1500.0</estimatedpriceperunit>
         <name>New Standard Laptop Computer</name>
         <pricingmodel>0</pricingmodel>
         <quantity>0</quantity>
         <serviceId>5</serviceId>
         <version>32</version>
        </ns1:getServiceDefinitionResult>
   </ns1:getServiceDefinitionResponse>
  </soap:Body>
</soap:Envelope>
```

submitRequisition の応答の例

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:req=>
  <soapenv:Header>
   <req:AuthenticationToken>
     <req:Username>admin</req:Username>
     <req:Password>admin</req:Password>
   </req:AuthenticationToken>
  </soapenv:Header>
  <soapenv:Body>
   <req:submitRequisition>
     <req:initiatorLoginName>admin</req:initiatorLoginName>
     <req:customerLoginName>admin</req:customerLoginName>
     <req:serviceRequests>
      <req:ServiceRequest>
        <req:name>New Standard Laptop Computer</req:name>
        <req:quantity>1</req:quantity>
        <req:sections>
         <req:Section>
           <req:fields>
            <req:Field>
              <req:name>ModelNumber</req:name>
              <req:value>
               <req:string>T60</req:string>
              </req:value>
            </req:Field>
            <req:Field>
              <req:name>AssetTag</req:name>
              <req:value>
               <req:string>ABC123</req:string>
              </req:value>
            </req:Field>
           </req:fields>
           <req:name>NewLaptop</req:name>
         </req:Section>
         <req:Section>
           <req:fields>
            <req:Field>
              <req:name>First_Name</req:name>
              <req:value>
               <req:string>admin</req:string>
              </req:value>
            </req:Field>
            <req:Field>
              <req:name>Last_Name</req:name>
              <req:value>
               <req:string>admin</req:string>
              </req:value>
```

```
    </req:Field>
    <req:Field>
      <req:name>Login_ID</req:name>
      <req:value>
       <req:string>admin</req:string>
      </req:value>
    </req:Field>
    <req:Field>
      <req:name>Personal_Identification</req:name>
      <req:value>
       <req:string />
      </req:value>
    </req:Field>
    <req:Field>
      <req:name>Email_Address</req:name>
      <req:value>
       <req:string>training3@cisco.com</req:string>
      </req:value>
    </req:Field>
    <req:Field>
      <req:name>Home_Organizational_Unit</req:name>
      <req:value>
       <req:string>Site Administration</req:string>
      </req:value>
    </req:Field>
    <req:Field>
      <req:name>Company_State</req:name>
      <req:value>
       <req:string />
      </req:value>
    </req:Field>
    <req:Field>
      <req:name>Supervisor</req:name>
      <req:value>
       <req:string />
      </req:value>
    </req:Field>
    <req:Field>
      <req:name>Supervisor_Email</req:name>
      <req:value>
       <req:string />
      </req:value>
    </req:Field>
    <req:Field>
      <req:name>Custom_1</req:name>
      <req:value>
       <req:string />
      </req:value>
    </req:Field>
    <req:Field>
      <req:name>Custom_2</req:name>
      <req:value>
       <req:string />
      </req:value>
    </req:Field>
  </req:fields>
  <req:name>Customer_Information</req:name>
</req:Section>
<req:Section>
  <req:fields>
   <req:Field>
     <req:name>First_Name</req:name>
     <req:value>
      <req:string>admin</req:string>
```

```
            </req:value>
          </req:Field>
          <req:Field>
            <req:name>Last_Name</req:name>
            <req:value>
             <req:string>admin</req:string>
            </req:value>
          </req:Field>
          <req:Field>
            <req:name>Login_ID</req:name>
            <req:value>
             <req:string>admin</req:string>
            </req:value>
          </req:Field>
          <req:Field>
            <req:name>Personal_Identification</req:name>
            <req:value>
             <req:string />
            </req:value>
          </req:Field>
          <req:Field>
            <req:name>Email_Address</req:name>
            <req:value>
             <req:string>training3@cisco.com</req:string>
            </req:value>
          </req:Field>
          <req:Field>
            <req:name>Home_Organizational_Unit</req:name>
            <req:value>
             <req:string>Site Administration</req:string>
            </req:value>
          </req:Field>
         </req:fields>
         <req:name>Initiator_Information</req:name>
       </req:Section>
      </req:sections>
      <req:version>32</req:version>
    </req:ServiceRequest>
   </req:serviceRequests>
  </req:submitRequisition>
 </soapenv:Body>
</soapenv:Envelope>
```

getMyAuthorizations の応答の例

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soap:Body>
   <ns1:getMyAuthorizationsResponse xmlns:ns1="http://smtask.api.newscale.com">
     <ns1:getMyAuthorizationsResult>
      <ns1:Activity>
        <activityFormId xmlns="http://smtask.api.newscale.com">2</activityFormId>
        <activityTypeId xmlns="http://smtask.api.newscale.com">2</activityTypeId>
        <actualDuration xmlns="http://smtask.api.newscale.com">0.0</actualDuration>
        <agentId xmlns="http://smtask.api.newscale.com">0</agentId>
        <clientOrganizationalUnit xmlns="http://smtask.api.newscale.com">
         <authorizationStructure>0</authorizationStructure>
         <billable>true</billable>
         <costCenterCode xsi:nil="true"/>
         <description xsi:nil="true"/>
         <GUID>3C921968-6474-45B2-8D65-A1822E52782F</GUID>
         <id>6</id>
```

```
            <localeId>1</localeId>
            <managerId>0</managerId>
            <managerName xsi:nil="true"/>
            <name>Field Sales</name>
            <organizationalUnitTypeId>2</organizationalUnitTypeId>
            <parentId>0</parentId>
            <parentName xsi:nil="true"/>
            <parentOrganizationalUnitGuid xsi:nil="true"/>
            <placeId>0</placeId>
            <placeName xsi:nil="true"/>
            <recordStateId>1</recordStateId>
            <tenantId>1</tenantId>
          </clientOrganizationalUnit>
          <clientOuId xmlns="http://smtask.api.newscale.com">6</clientOuId>
          <creatorObjectId xmlns="http://smtask.api.newscale.com">57</creatorObjectId>
          <creatorObjectInstId
xmlns="http://smtask.api.newscale.com">9</creatorObjectInstId>
          <customer xsi:nil="true" xmlns="http://smtask.api.newscale.com"/>
          <customerId xmlns="http://smtask.api.newscale.com">12</customerId>
          <customerName xmlns="http://smtask.api.newscale.com">Terry Training</customerName>
          <customerRoleId xmlns="http://smtask.api.newscale.com">0</customerRoleId>
          <customerRoleName xsi:nil="true" xmlns="http://smtask.api.newscale.com"/>
          <defActivityId xmlns="http://smtask.api.newscale.com">0</defActivityId>
          <depth xmlns="http://smtask.api.newscale.com">0</depth>
          <displayOrder xmlns="http://smtask.api.newscale.com">0</displayOrder>
          <dueOn xmlns="http://smtask.api.newscale.com">2009-06-03T23:00:00-07:00</dueOn>
          <dueOnTz xmlns="http://smtask.api.newscale.com">149</dueOnTz>
          <effort xmlns="http://smtask.api.newscale.com">0.5</effort>
          <escalationLevel xmlns="http://smtask.api.newscale.com">0</escalationLevel>
          <expectedDuration xmlns="http://smtask.api.newscale.com">8.0</expectedDuration>
          <flagId xmlns="http://smtask.api.newscale.com">0</flagId>
          <formURL xsi:nil="true" xmlns="http://smtask.api.newscale.com"/>
          <group xmlns="http://smtask.api.newscale.com">0</group>
          <hasChildren xmlns="http://smtask.api.newscale.com">false</hasChildren>
          <icon xsi:nil="true" xmlns="http://smtask.api.newscale.com"/>
          <id xmlns="http://smtask.api.newscale.com">422</id>
          <instructions xmlns="http://smtask.api.newscale.com"/>
          <isBusy xmlns="http://smtask.api.newscale.com">0</isBusy>
          <isLast xmlns="http://smtask.api.newscale.com">false</isLast>
          <lastChannelId xsi:nil="true" xmlns="http://smtask.api.newscale.com"/>
          <listCount xmlns="http://smtask.api.newscale.com">-1</listCount>
          <nextActionId xmlns="http://smtask.api.newscale.com">5</nextActionId>
          <overbookTime xmlns="http://smtask.api.newscale.com">0</overbookTime>
          <parentId xmlns="http://smtask.api.newscale.com">0</parentId>
          <performerActualDuration
xmlns="http://smtask.api.newscale.com">0.0</performerActualDuration>
          <performerId xmlns="http://smtask.api.newscale.com">11</performerId>
          <performerName xmlns="http://smtask.api.newscale.com">Jared
Roberts</performerName>
          <performerOfficeId xmlns="http://smtask.api.newscale.com">0</performerOfficeId>
          <performerRoleId xmlns="http://smtask.api.newscale.com">331</performerRoleId>
          <performerRoleName xsi:nil="true" xmlns="http://smtask.api.newscale.com"/>
          <performerSharable
xmlns="http://smtask.api.newscale.com">false</performerSharable>
          <performerShared xmlns="http://smtask.api.newscale.com">false</performerShared>
          <priority xmlns="http://smtask.api.newscale.com">0</priority>
          <priorityName xsi:nil="true" xmlns="http://smtask.api.newscale.com"/>
          <processId xmlns="http://smtask.api.newscale.com">0</processId>
          <projectActivityId xmlns="http://smtask.api.newscale.com">0</projectActivityId>
          <reqId xmlns="http://smtask.api.newscale.com">170</reqId>
          <retryCount xmlns="http://smtask.api.newscale.com">0</retryCount>
          <scheduledStart
xmlns="http://smtask.api.newscale.com">2009-06-03T15:00:00-07:00</scheduledStart>
```

```
        <startedOn
xmlns="http://smtask.api.newscale.com">2009-06-03T12:09:41.443-07:00</startedOn>
        <startedOnTz xmlns="http://smtask.api.newscale.com">149</startedOnTz>
        <stateId xmlns="http://smtask.api.newscale.com">6</stateId>
        <stateName xsi:nil="true" xmlns="http://smtask.api.newscale.com"/>
        <stepId xmlns="http://smtask.api.newscale.com">4</stepId>
        <stepLogicName xsi:nil="true" xmlns="http://smtask.api.newscale.com"/>
        <subject xmlns="http://smtask.api.newscale.com">Computer Memory - Upgrade -
APPROVAL NEEDED</subject>
        <taskUrl xsi:nil="true" xmlns="http://smtask.api.newscale.com"/>
        <ticketId xmlns="http://smtask.api.newscale.com">175</ticketId>
        <ticketObjectId xmlns="http://smtask.api.newscale.com">37</ticketObjectId>
        <totalCost xmlns="http://smtask.api.newscale.com">0.0</totalCost>
        <waiting xmlns="http://smtask.api.newscale.com">0</waiting>
        <WDDXCheckList xsi:nil="true" xmlns="http://smtask.api.newscale.com"/>
      </ns1:Activity>
    </ns1:getMyAuthorizationsResult>
  </ns1:getMyAuthorizationsResponse>
 </soap:Body>
</soap:Envelope>
```

REST/Web サービスのエラー メッセージ

以下のエラー メッセージにおいて数値を波カッコで囲んだ記号の部分(「{0}」など)は、実際のエラー メッセージではエラーの発生元となったオブジェクトの名前や ID で置き換えられます。

AUTH_0001	このユーザはまだ認証されていないか、セッションがタイムアウトになっています。
AUTH_0002	ユーザ「{0}」の認証に失敗しました。
AUTH_0003	Service Catalog は SSO を使用するよう設定されていますが、設定の問題により SSO が正しく動作していません。
AUTH_0004	パスワードを正しく暗号化する必要があります。
AUTH_0005	ユーザ名のヘッダーが正しくありません。存在していないか、空白です。正確なヘッダーを送信してください。
AUTH_0006	Web サービスへのアクセスがオフになっています。
AUTH_0007	ユーザがこの Web サービスにアクセスできません。
AUTH_0008	SOAP ヘッダーの構造が無効であるか、セッションがタイムアウトしました。
AUTH_0009	リモート ユーザ名が無効です。有効な名前で再試行してください。
AUTH_0010	ユーザに、バージョン情報を表示する権限がありません。
AUTH_0011	アカウントがロックされています。システム管理者にお問い合わせください。
AUTH_0012	パスワードは <date> に期限切れになります。この日付より前にパスワードをリセットしないと、アカウントが停止されます。 つまり、パスワードが期限切れとなり、アカウントが停止します。
INFRA_0001	Service Catalog Business Engine キューが使用できず、非同期送信についての管理者設定がオンになっているため、要求を送信できません。後で再実行するか、管理者にお問い合わせください。
REQ_0001	イニシエータ「{0}」がシステム上で見つかりません。
REQ_0002	カスタマー「{0}」がシステム上で見つかりません。
REQ_0003	サービス「{0}」がシステム上で見つかりません。

REQ_0004	ユーザ「{0}」がシステム上で見つかりません。
REQ_0005	要求 ID「{0}」がシステム上で見つかりません。
REQ_0006	指定した要求 ID「{1}」が一致しないため、要求エントリ「{0}」をキャンセルできません。
REQ_0007	指定したサービス「{1}」が一致しないため、要求エントリ「{0}」をキャンセルできません。
REQ_0008	ユーザにこの要求をキャンセルするための権限がありません。要求をキャンセルできるのは要求のオーナーだけです。
REQ_0009	この要求はすでにキャンセルされています。
REQ_0010	カスタマー「{0}」にはサービス「{1}」をオーダーする権限がありません。
REQ_0011	サービス「{0}」はオーダーできません。
REQ_0012	この要求にコメントを追加するための権限がユーザにありません。
REQ_0013	サービス フォームの必須フィールド「{0}」に値が入力されていません。
REQ_0014	サービス フォーム フィールド「{0}」が、許可されている最大の長さを超えています。
REQ_0015	フィールド「{0}」に有効な数字を入力してください。
REQ_0016	サービス フォーム フィールド「{0}」には、値は 1 つしか指定できません。
REQ_0017	ユーザにこの要求にアクセスするための権限がありません。
REQ_0018	要求で指定されたバージョンが、サービス「{0}」のデータベースにあるバージョンと一致しません。最新のサービス定義を使用してください。
REQ_0019	イニシエータ「{0}」には、このカスタマー「{1}」に対する Order on Behalf 権限がありません。
REQ_0020	認証済みユーザ「{0}」に、Web サービスの RAPI システム アカウント権限がありません。
REQ_0021	このフィールド「{0}」に渡された値が、オプション リストにありません。
REQ_0022	このフィールド「{0}」の値に、設計されている値を超える量のデータが含まれています。
REQ_0023	このフィールド「{0}」に渡された値が、オプション リストにありません。
REQ_0024	サービス フォーム フィールド「{0}」に無効な日付形式が含まれています。
REQ_0025	サービス フォーム フィールド「{0}」に無効な日時形式が含まれています。
REQ_0026	サービス フォーム フィールド「{0}」で、指定されたログイン「{1}」がシステムに存在しません。ログイン名を入力してください。
REQ_0027	この要求 ID「{0}」はキャンセルできません。この要求は閉じられています。
REQ_0028	この要求 ID「{0}」は、変更可能な時点を過ぎています。この要求はキャンセルできません。
REQ_0029	この要求エントリ ID「{0}」はキャンセルできません。この要求エントリは閉じられています。
REQ_0030	サービス「{0}」のステータスが非アクティブです。
REQ_0031	この要求 ID「{0}」はキャンセルできません。この要求エントリはすでにキャンセルされています。
REQ_0032	このフィールド「{0}」に渡された値が、オプション リストにありません。
REQ_0033	このフィールド「{0}」に渡された値が、オプション リストにありません。

REQ_0034	このフィールド「{0}」の値に、設計されている値を超える量のデータが含まれています。
REQ_0035	ディクショナリ名「{0}」がこのサービス用に存在しません。フィールド名を修正してください。
REQ_0036	ディクショナリ フィールド「{0}」がこのディクショナリ「{0}」用に存在しません。フィールド名を修正してください。
REQ_0037	ユーザにこの要求エントリをキャンセルするための権限がありません。
REQ_0038	フィールドの数がこのディクショナリ「{0}」に一致しません。
REQ_0039	ディクショナリの数がこのサービス「{0}」に一致しません。
REQ_0040	ユーザ「{0}」は Service Catalog データベースに存在しません。LDAP からユーザのインポートを試みましたが、LDAP ルックアップが false に設定されています。
REQ_0041	OOB イベントが設定または有効にされていません。このユーザ「{0}」の LDAP 検索は実行されませんでした。
REQ_0042	LDAP からデータをインポート中にエラーが発生しました。
REQ_0043	ユーザ「{0}」が Service Catalog データベースと LDAP システムに存在しません。
REQ_0044	サービス フォームの個人検索イベントが設定されていないか、有効になっていません。このユーザ「{0}」の LDAP 検索が実行されませんでした。
REQ_0045	LDAP から個人をインポートするためのイベント処理が正しく設定されていません。
REQ_0046	このフィールド「{0}」には通貨記号なしの金額のみを入力してください。小数点として「.」を使用してください。
REQ_0047	このフィールド「{0}」には文字と数字だけを入力してください。
REQ_0048	このフィールド「{0}」には文字と数字だけを入力してください。
REQ_0049	ディクショナリ「{0}」が要求に複数回繰り返されています。特定のディクショナリは一度だけ要求に表示できます。
REQ_0050	ディクショナリ フィールド「{0}.{1}」が要求に複数回繰り返されています。特定のディクショナリ フィールドは一度だけ要求に表示できます。
REQ_0051	サービス「{0}」に 1 つ以上のグリッド ディクショナリが含まれています。この API はグリッド ディクショナリ値を持つ要求の送信を受け入れません。
REQ_0052	サービス フォームは次のエラーのために送信できませんでした:{0}
REQ_0053	サービス フォームは次のエラーのために送信できませんでした:必須フィールド {0} がありません
REQ_0054	オーダー モードに基づいた個々の送信だけが可能なサービスが 1 つ以上含まれているため、要求を送信できませんでした。
REQ_0055	プロセスが実行された最後の日付
REQ_0056	ディクショナリ「{0}」に対して許可される最大レコード数「{1}」に達しました。
REQ_0057	リクエスト ID「{0}」はすでに送信済みです。
REQ_0058	未送信の要求が見つかりません。
REQ_0059	オーダー モード オプションがサービス {0} に対して無効になっています。
REQ_0060	要求をキャンセルできません、または削除できません。

REQ_0061	複数の未送信要求が見つかりました。
REQ_0062	要求 {0} のエントリが見つかりません。
REQ_0063	要求 {0} のエントリをキャンセルできません、または削除できません。
REQ_0064	要求操作のためのアクセスが拒否されました。
REQ_0065	この要求の要求エントリが見つかりません。
REQ_0100	実行時の例外が発生しました。これは正規の Business Engine ワークフロー例外によって発生した可能性があります(Service Designer で定義されているためタスクのキャンセルが許可されていないなど)。タスク定義を確認してください。
REQ_0101	サービス フォーム データの読み取り中にランタイム エラーが発生しました。
REQ_0101	サービス フォーム データの読み取り中にランタイム エラーが発生しました。
TASK_0005	タスク「{0}」は拒否できません。
TASK_0008	タスク「{0}」がシステムにありません。
TASK_0001	ユーザ「{0}」には「{1}」を許可する権限がありません。
TASK_0002	ユーザ「{0}」には「{1}」を拒否する権限がありません。
TASK_0003	ユーザ「{0}」には「{1}」を確認する権限がありません。
TASK_0004	タスク「{0}」は許可タスクではありません。
TASK_0005	タスク「{0}」は拒否できません。
TASK_0006	タスク「{0}」は確認タスクではありません。
TASK_0008	タスク「{0}」がシステムにありません。
TASK_0009	指定したタスク ID「{1}」に対する要求エントリ ID「{0}」が一致しません。
TASK_0010	タスク「{0}」には複数の要求エントリを指定できません。
TASK_0011	タスク「{0}」には財務または OU の承認がありません。
TASK_0012	ユーザ「{0}」には、このタスク「{1}」に対する部分的な要求エントリを拒否する権限がありません。
TASK_0013	タスク「{0}」はすでに拒否されています。

CHAPTER 3

Prime Service Catalog RESTful API のリファレンスおよび例

概要

この章では、サービス カタログ（Service Catalog）の Web サービスの使用法について説明します。これには、SOAP ベース バージョンの要求 API（RAPI 2）を実装する Web サービスが含まれます。これは、外部システムが Service Catalog 内でサービス要求を作成および管理できるようにする API です。Web サービスには、サービス要求内の提供タスクや承認タスクの管理を許可し、Service Catalog の内容の確認を行う要求なども含まれます。

REST URL の構文および表記法

- REST の URL は次の規則に従います。

  ```
  http(s)://<serverURL>/RequestCenter/nsapi/<entityGroup>/<entityType>/<filters>?sortBy=<columnName>&sortDir=<sortOrder>?startRow=<x>&recordSize=<y>
  ```

  角カッコ（<>）で囲まれた要素は、適切なパラメータまたはパラメータ値を代入する必要があることを示します。
- フィルタ、ソート、およびページングのコントロールとアクションは、オプション パラメータとして REST URL に渡されます。フィルタには、「サポートされているフィルタ」の項で説明するように、1 つまたは複数の式を含めることができます。フィルタを指定しなかった場合は、エンティティに関して見つかったすべてのインスタンスが返されます。
- 複数のフィルタの組み合わせが可能なこともあります。その場合、2 番め以降のパラメータの指定はオプションです。このようなパラメータについては、構文内でそのパラメータ構文を角カッコ（[]）で囲むことによって示しています。オプションのパラメータは、区切り文字（|）で囲む必要があります。
- 特記のない限り、すべての URL で大文字と小文字が区別されます。
- バージョン管理は nsAPI に組み込まれています。ベース バージョン 1.0 の REST URL にはバージョン番号を表すパラメータがありません。しかし、これは将来のリリースで変更される可能性があります。
- 要求された操作が成功したのか、それとも失敗したのかを示すために、HTTP のステータス コードおよびエラー メッセージが REST 応答で返されます。
- nsAPI Java バインディング パッケージの名前は com.newscale.nsapi.* です。

サポートされているフィルタ

nsAPI は、次のように GET 操作においてエンティティのタイプに基づき多様なフィルタをサポートしています。

表 3-1　サポートされているフィルタ テーブル

エンティティ タイプ	使用可能なフィルタ/構文	REST URL の例
すべてのエンティティ	ID /id/<value>	http://serverURL/RequestCenter/nsapi/definition/servicedefs/id/16
	名前:完全一致 /name/<value>	http://serverURL/RequestCenter/nsapi/definition/servicedefs/name/Create%20Custom%20VM
	名前:ワイルドカード検索 ?name=<value>	http://serverURL/RequestCenter/nsapi/definition/servicedefs?name=Create%20Custom*
標準、サービス項目、カスタムコンテンツ	すべてのテーブルのカラム。次の要素で構成されたフィルタ式を最大 3 つ使用可能 • 比較演算子:=、>、<、>=、<= • 関係演算子:AND、OR(大文字と小文字を区別しない、オーダーの優先順位はサポートされていません) • 区切り文字:| /<columnName1><operator1><value1>[|<AND|OR>|]<columnName2><operator2><value2>][|<AND|OR>|<columnName3><operator3><value3>] *日付フィールドの値は mm-dd-yyyy 形式にする必要があります。*	http://serverURL/RequestCenter/nsapi/standard/StOperatingSystem/Custom1=Linux|AND|Custom2=64 *これは、次のように記述することもできます。* http://serverURL/RequestCenter/nsapi/standard/StOperatingSystem/Custom1=Linux|Custom2=64 http://serverURL/RequestCenter/nsapi/standard/StOperatingSystem/Custom1=Linux|OR|Custom2=64 http://serverURL/RequestCenter/nsapi/serviceitems/serviceitemsubscription/SubmitDate>=03-01-2011 http://serverURL/RequestCenter/nsapi/serviceitems/serviceitemsubscription/AccountName=tenantaccount1 http://serverURL/RequestCenter/nsapi/customcontent/UcAnnouncementObj/Category=Corporate
サービス項目(Service Items)	ビュー名(View Name) ?ViewName=<value> *指定可能なビュー名は次のとおりです。* • *My ServiceItems (My Services に表示される)* • *Manage ServiceItems (Service Item Manager に表示される)*	http://serverURL/RequestCenter/nsapi/serviceitem/SiDesktop?ViewName=My%20ServiceItems http://serverURL/RequestCenter/nsapi/serviceitem/SiDesktop?ViewName=Manage%20ServiceItems

表 3-1　サポートされているフィルタ テーブル(続き)

エンティティ タイプ	使用可能なフィルタ/構文	REST URL の例
組織	OU Type ?type=<all/businessUnit/service Team>	http://serverURL/RequestCenter/nsapi/directory/organizationalunits?type=serviceTeam
個人(Person)	ログイン名(Login Name) ?loginname/<value>	http://serverURL/RequestCenter/nsapi/directory/people?loginname/dsmith
	OU 名前 ?ouname=<value>	http://serverURL/RequestCenter/nsapi/directory/people?ouname=Operations
	グループ名(Group Name) ?groupname=<value>	http://serverURL/RequestCenter/nsapi/directory/people?groupname=Approvers
	役割名 ?rolename=<value> *継承されたロールをフィルタに使用することはできません。*	http://serverURL/RequestCenter/nsapi/directory/people?rolename=Service%20Performer
カテゴリ (Categories)	カテゴリ タイプ(Category Type) ?catalogType=<serviceCatalog/offeringCatalog>	http://serverURL/RequestCenter/nsapi/definition/categories?catalogType=serviceCatalog
	ユーザ コンテキスト ?userContext=<true/false>	http://serverURL/RequestCenter/nsapi/definition/categories?userContext=true
サービス	カテゴリ名(Category Name) ?categoryName=<value>	http://serverURL/RequestCenter/nsapi/definition/servicedefs?categoryName=Manage%20Physical%20Servers
	キーワード ?keywordName=<value>	http://serverURL/RequestCenter/nsapi/definition/servicedefs?keywordName=server
要求および承認	ビュー名、ステータス /ViewName=<value1>[| Status=<value2>] *フィルタに使用可能なビュー名は、My Services の [Requisitions] タブおよび [Authorizations] タブにあるビューのリストに対応しています。* *ステータス フィルタはビュー名フィルタとともに使用する必要があり、単独では使用できません。*	http://serverURL/RequestCenter/nsapi/transaction/requisitions/ViewName=Ordered%20for%20Self http://serverURL/RequestCenter/nsapi/transaction/authorizations/ViewName=Authorizations%20for%20Self http://serverURL/RequestCenter/nsapi/transaction/authorizations/ViewName=Authorizations%20for%20Self| Status=Approved

表 3-1　サポートされているフィルタ テーブル(続き)

エンティティ タイプ	使用可能なフィルタ/構文	REST URL の例
タスク	ビュー名（View Name） ?viewName=<value> *フィルタに使用可能なビュー名は、Service Manager のシステム定義ビューのリストに対応しています。ユーザ定義ビューをフィルタに使用することはできません。*	http://serverURL/RequestCenter/nsapi/transaction/tasks?viewName=AvailableWork
タスク（特定の要求エントリに関するもの）	タスク タイプ（Task Type） ?taskType=<value> 可能なタスク タイプは「すべて」、「提供」、および「承認」です。 タスク ステータス（省略） ?showSkippedTasks=<false|true>	http://<ServerURL>/RequestCenter/nsapi/transaction/tasks/RequisitionEntryNumber=1234?taskType=delivery&showSkippedTasks=true

POST 操作では、次のフィルタがサポートされています。

表 3-2　POST 操作でサポートされているフィルタ テーブル

エンティティ タイプ	使用可能なフィルタ	REST URL の例
個人(Person)	ログイン名または Id *フィルタは操作の中で暗黙的に行われ、個人の識別情報が要求 XML から取得されます。*	http://serverURL/RequestCenter/nsapi/directory/people/update *同じ URL が作成操作と更新操作の両方で使用されます。ユーザの属性を URL に渡す必要はありません。*
タスク	ID /<value>/<done |approve |reject |review> *「|」は、指定可能なオプションを 1 つだけ選択する必要があることを示しています。*	http://serverURL/RequestCenter/nsapi/transaction/tasks/215/approve

ワイルドカード検索エントリ

エンティティ名フィルタでワイルドカード検索を行う場合（?name=<value> など）、検索文字列の先頭のワイルドカード文字（「*」や「%」など）は無視されます。これらの文字が文字列の中または末尾にある場合は、ワイルドカード照合で適用されます。

名前検索で先頭のワイルドカード文字の使用を有効にするには、ContainsQueryInFnS プロパティを newscale.properties ファイル（RequestCenter.war/WEB-INF/classes/config にあります）で見つけ、値を true に設定します。

```
ContainsQueryInFnS=true
```

このプロパティでは、Service Manager および Service Link モジュールでのワイルドカード検索の完全サポートも制御します。

(注)　このような検索操作は、システム パフォーマンスに悪影響を与える可能性があり、実稼働環境では一般的に推奨されません。

関連付けられているエンティティとは、プライマリ エンティティと一緒に取得される、ネストされたエンティティのことです。たとえば、個人がメンバとして属している OU などがあります。関連付けられているエンティティ名のフィルタは、ワイルドカード検索をサポートしません。このようなフィルタによる検索では、完全一致の結果のみが返されます。また、大文字小文字が区別されます。つまり、検索文字列内のワイルドカード文字は、照合の際にリテラルとして扱われます。次に例を示します。

```
/nsapi/directory/people?ouname=star*OU
```

組織内の正確に「star*OU」という名前の個人を返します。

```
/nsapi/directory/people?groupname=starGroup*
```

グループ内の正確に「starGroup*」という名前の個人を返します。

ソート コントロールとページング コントロール

ソート

複数のデータ行を返す操作では、ソート コントロールを使用できます。フィルタの場合と同様、ソート コントロールは REST URL のパラメータとして指定されます。

```
?sortBy=<columnName>&sortDir=<sortOrder>
```

- sortBy – ソートの基準とするフィールド名。
- sortDir – ソートの方向。指定可能な値は **asc**（昇順）と **desc**（降順）です。

どのソート パラメータも REST URL に渡されない場合は、次の表に示すように、デフォルトのソート フィールドとソート順序が使用されます。

表 3-3　　ソート パラメータ

エンティティ タイプ	デフォルトのソート フィールドとソート順序
カテゴリ（Categories）	Name（昇順）
サービス	Name（昇順）
カリキュラム	Name（昇順）
エージェント（Agents）	Name（昇順）
人員（People）	First Name（昇順）
組織	Name（昇順）
グループ（Groups）	Name（昇順）
アカウント	Name（昇順）
すべてのサービス項目	Row Id*（昇順）
サービス項目の詳細	Row Id*（昇順）
標準（Standards）	Row Id*（昇順）

表 3-3　ソート パラメータ(続き)

カスタム コンテンツ(Custom Content)	Row Id*(昇順)
要求	Submit Date(昇順)、Requisition Id(昇順)
要求エントリ	要求エントリ ID(Requisition Entry ID)(昇順)
タスク	タスク名(Task Name)(昇順)、タスク ID(Task Id)(昇順)
承認	要求 ID(Requisition Id)(昇順)、タスク ID(Task Id)(昇順)
契約	Agreement Name(昇順)

Row Id とは、サービス項目、標準、およびカスタム コンテンツ テーブルにレコードが挿入された物理的な順序を表します。これは、それらの各テーブルの PrimaryID というカラムに対応しています。

サービス項目、標準、およびカスタム コンテンツのエンティティはすべてのカラムのソートをサポートしています。他のエンティティは、特定のフィールドによるソートだけがサポートされています。詳細については、API リファレンスおよび例を参照してください。

例

```
/nsapi/directory/people?sortBy=lastName&sortDir=asc
```

個人レコードが、姓の昇順にリストされて返されます。

```
/nsapi/directory/people?sortBy=login&sortDir=desc
```

個人レコードが、ログイン名の降順にリストされて返されます。

ページング

複数のデータ行を返す操作では、ページ コントロールを使用できます。また、REST URL のパラメータとしても指定されます。

```
?startRow=<x>&recordSize=<y>
```

- startRow - レコードの取得を開始する行。デフォルト値は 1 です。
- recordSize – 同時に取得されるレコード数。デフォルト値は、Portal Designer の共通設定で設定されます。許容最大レコード数は 50 です。

上記の各パラメータおよび返されるレコードの総数が、REST XML 応答のルート タグ内に属性として示されます。次に例を示します。

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<categories totalCount="11" recordSize="10" startRow="1">
  <category>
    <categoryId>1</categoryId>
    <categoryName><b>Consumer Services</b></categoryName>
    <description />
    .  .  .
```

取得されるレコード数より大きい startRow を指定すると、HTTP 500 エラーが返されます。取得されるレコード数より大きい recordSize を指定すると、すべての行が返されます。

例

次の例では、Service Designer で 60 個のサービスが定義されており、Portal Designer の nsAPI 設定で指定できるレコードの最大数が 30 であると想定しています。

例 1:

```
/nsapi/definition/servicedefs
```

最初の 30 個のサービスが返されます。このような要求に対する応答の例は次のようになります。

```
<services totalCount="60" recordSize="30" startRow="1">
  .  .  .
</services>
```

例 2:

```
/nsapi/definition/servicedefs?startRow=4
```

4 番めのサービスから始まる 30 個のサービスが返されます。このような要求に対する応答の例は次のようになります。

```
<services totalCount="60" recordSize="30" startRow="4">
  .  .  .
</services>
```

例 3:

```
/nsapi/definition/servicedefs?startRow=4&recordSize=5
```

4 番めのサービスから始まる 5 個のサービスが返されます。このような要求に対する応答の例は次のようになります。

```
<services totalCount="60" recordSize="5" startRow="4">
  .  .  .
</services>
```

例 4:

```
/nsapi/definition/servicedefs?startRow=4&recordSize=35
```

4 番めのサービスから始まる 30 個のサービスが返されます。このような要求に対する応答の例は次のようになります。

```
<services totalCount="60" recordSize="30" startRow="4">
.  .  .
</services>
```

ネストされたエンティティ

ある種のエンティティにはネスト構造が含まれます。第 1 レベルの子だけまたは関連付けられたエンティティの取得は nsAPI によってサポートされます。親エンティティおよびその子エンティティを表にまとめます。

表 3-4　ネストされたエンティティ

親エンティティ	子エンティティ
カテゴリ（Categories）	カテゴリのサブカテゴリ、組み込みサービス、および提供
サービス	サービスに関連付けられたカテゴリ、キーワード、およびバンドル サービス
個人(Person)	個人に関連付けられた OU、グループ、ロール、住所、連絡先、設定、および代理人

結果セットで複数行のデータを返すカテゴリ検索、サービス検索、または人の検索では、関連付けられたエンティティはパフォーマンス上の理由により取得されません。関連付けられたエンティティを利用できるのは、個々のエンティティに対する ID または名前による取得操作の場合だけです。

例

```
/nsapi/directory/people/loginname/<value>
```

個人がメンバーとなっているグループが返されます。

```
<person>
  .  .  .
  <associatedGroups>
    <associatedGroup>
      <id>2</id>
      <name>group2</name>
    </associatedGroup>
    <associatedGroup>
    <id>5</id>
      <name>group5</name>
    </associatedGroup>
  </associatedGroups>
</person>
```

```
/nsapi/directory/people/id/<value>
```

個人が属しているグループが返されます。

```
/nsapi/directory/people
```

個人が属しているグループは返され**ません**。

```
/nsapi/directory/people?ouname=<value>
```

個人が属しているグループは返され**ません**。

JavaScript ポートレットでの nsAPI の使用

nsAPI コールは、Portal Designer モジュールで開発された JavaScript ポートレットで呼び出すことができます。JavaScript、REST URL、AJAX、または Ext JS コンポーネントを使用すると、ポートレットを作成して目的のエンティティのデータを取得し、グリッド形式でレンダリングできます。

Ext JS グリッドでデータをレンダリングする

以下に、Ext JS を使用してカテゴリのリストをレンダリングする例を示します。

```
getUrl = "/RequestCenter/nsapi/definition/categories";

var proxy = new Ext.data.HttpProxy({
    url: getUrl,
    method: 'GET'
});

defaultPageSize = 5;
fieldList = [ "categoryId",
              "categoryName",
              "description",
              "topDescriptionEnabled",
              "topDescription",
              "topDescriptionURL",
              "middleDescriptionEnabled",
              "middleDescription",
              "middleDescriptionURL",
              "bottomDescriptionEnabled",
              "bottomDescription",
              "bottomDescriptionURL",
              "catalogTypeId",
              "catalogType",
              "isRoot",
              "descriptionURL",
              "categoryURL"]

displayList = [
                        {id: 'id', header: 'Id', width: 50,sortable: false,  dataIndex:
'categoryId'},
                        {header: 'Name',sortable: true, dataIndex: 'categoryName'},
                        {header: 'Description', dataIndex: 'description',sortable: false},
                        {header: 'TD Enabled', dataIndex:
'topDescriptionEnabled',hidden:true,sortable: false},
                        {header: 'Top Description', dataIndex:
'topDescription',hidden:true,sortable: false},
                        {header: 'Top Description URL', dataIndex:
'topDescriptionURL',hidden:true,sortable: false},
                        {header: 'Middle DescriptionEnabled', dataIndex:
'middleDescriptionEnabled',hidden:true,sortable: false},
                        {header: 'Middle Description', dataIndex:
'middleDescription',hidden:true,sortable: false},
                        {header: 'Middle Description URL', dataIndex:
'middleDescriptionURL',hidden:true,sortable: false},
                        {header: 'Bottom Description Enabled', dataIndex:
'bottomDescriptionEnabled',hidden:true,sortable: false},
                        {header: 'Bottom Description', dataIndex:
'bottomDescription',hidden:true,sortable: false},
                        {header: 'Bottom Description URL', dataIndex:
'bottomDescriptionURL',hidden:true,sortable: false},
                        {header: 'Catalog Type Id', dataIndex:
'catalogTypeId',hidden:true,sortable: false},
                        {header: 'Catalog Type', dataIndex:
'catalogType',hidden:true,sortable: false},
                        {header: 'isRoot', dataIndex: 'isRoot',hidden:true,sortable:
false},
```

```
                        {header: 'descriptionURL', dataIndex:
'descriptionURL',hidden:true,sortable: false},
                        {header: 'URL', dataIndex: 'categoryURL',sortable: false}
        ];


var store = new Ext.data.XmlStore({
    autoDestroy: true,
    proxy: proxy,
    root : "categories",
    record: 'category',
    idPath: 'rowId',
    totalProperty: '@totalCount',
    autoLoad: true,
    paramNames: {
      start: 'startRow',
      limit: 'recordSize',
      catName : 'categoryName'
    },
    fields: fieldList
});

var sortchange = function(obj,direction) {
    store.setBaseParam("sortBy","categoryName");
  store.setBaseParam("sortDir",direction.direction);
  store.load();
}


var grid =  new Ext.grid.GridPanel({
            renderTo :  '#divName#',
            store : store,
            autoWidth : true,
            height : 300,
            title : 'Category List',
            colModel: new Ext.grid.ColumnModel({
            defaults: {
               width: 120,
               sortable: true
            },
            columns:displayList,
             }),
            bbar : [new Ext.PagingToolbar({
                                store : store,
                                displayInfo : true,
                                pageSize : defaultPageSize,
                                params:{
                                         startRow: 1,
                                         recordSize: defaultPageSize
                                       }
               })]

   });

grid.on("sortchange",sortchange);
store.setBaseParam("recordSize",5);
store.load();
```

ログイン ユーザの取得

現在ログインしているユーザの名前空間変数は JavaScript ポートレットで使用できるようになります。次に例を示します。

```
Ext.onReady(function() {
  /* Demonstrate JavaScript to get Logged-In user details */
  alert('PersonId: ' +nsAPP_CurrentUserId);
  alert('Login name: ' +nsAPP_CurrentUserLoginName);
  alert('First name: ' +nsAPP_CurrentUserFirstName);
  alert('Last name: ' +nsAPP_CurrentUserLastName);
  alert('HomeOUId: ' +nsAPP_CurrentUserHomeOuId);
}
```

JSR ポートレットでの nsAPI の使用

認証

JSR ポートレットを介してアクセスした場合、nsAPI Java クライアントを使用してログイン操作およびログアウト操作を呼び出せます。

```
import com.newscale.nsapiclient.NSApiClientFactory;
import com.newscale.nsapiclient.NSApiClient;
.  .  .
NSApiClient nsApiClient = NSApiClientFactory.getInstance();
nsApiClient.login("http://<serverURL>/RequestCenter","username",   "password" ); // Login
by username, password.
.  .  .
.  .  .
nsApiClient.logout(); // Logout

.  .  .
NSApiClient nsApiClient = NSApiClientFactory.getInstance();
nsApiClient.login("http://<serverURL>/RequestCenter",
sessionId); // Login using current session id.
.  .  .
.  .  .
nsApiClient.logout();
```

ログイン ユーザの取得

次に、ログインしているユーザの詳細情報を取得する、Spring ベースの JSR Portlet Controller の例を示します。

```
import javax.portlet.RenderRequest;
import javax.portlet.RenderResponse;
import org.springframework.ui.Model;
import org.springframework.stereotype.Controller;
import com.newscale.nsapi.directory.person.Person;
…
@Controller
public class MyJSRController {
private NSApiClient nsApiClient = getNSApiClient();
public NSApiClient getNsApiClient() {
return nsApiClient;
  }
public void setNsApiClient(NSApiClient nsApiClient) {
this.nsApiClient = nsApiClient;
```

```
}
@RequestMapping("VIEW")
@RenderMapping("NORMAL")
public String viewNormal(RenderRequest request, RenderResponse response, Model model) {
 nsApiClient.login("http://<AppServer host>:<port>/RequestCenter",
request.getPortletSession().getId());
// Get Currently Logged-in user from nsAPI client
Person persons  = nsApiClient.getDirectory().getCurrentUser();
// Get user info
long personId = persons.getPersonId());
long homeOUId = persons.getHomeOrganizationalUnitId());
String firstName =persons.getFirstName());
String lastName =persons.getLastName());
String username =persons.getLogin());
  }
}
```

Get 操作

ここでは、エンティティの取得方法を示すサンプル コードの一部をいくつか示します。これらの方法の詳細については、個々のエンティティ クラスに関する Javadoc を参照してください。

- Id による個人の取得

```
package com.newscale.nsapiclient.directory;
import com.newscale.nsapi.directory.person.Person;
import com.newscale.nsapi.NSApiConstants;
import com.newscale.nsapiclient.directory.Directory;
import com.newscale.nsapiclient.NSApiClientFactory;
import com.newscale.nsapiclient.NSApiClient;
.  .  .

NSApiClient nsApiClient = NSApiClientFactory.getInstance();
nsApiClient.login("http://<serverURL>/RequestCenter","username","password");

Person person = nsApiClient.getDirectory().getPersonById(123);

/* This is the equivalent of REST URL
http://<serverURL>/RequestCenter/nsapi/directory/people/id/123
*/
.  .  .
nsApiClient.logout();
```

- Name による個人の取得

```
package com.newscale.nsapiclient.directory;
import com.newscale.nsapi.directory.person.Person;
import com.newscale.nsapi.NSApiConstants;
import com.newscale.nsapiclient.directory.Directory;
import com.newscale.nsapiclient.NSApiClientFactory;
import com.newscale.nsapiclient.NSApiClient;
.  .  .
NSApiClient nsApiClient = NSApiClientFactory.getInstance();
nsApiClient.login("http://<serverURL>/RequestCenter","username","password");
Person person = nsApiClient.getDirectory().getPersonByLoginName("jsmith");
/*
This is the equivalent of REST URL
http://<serverURL>/RequestCenter/nsapi/directory/people/loginname/jsmith
*/

.  .  .
nsApiClient.logout();
```

- すべての人の取得

```
package com.newscale.nsapiclient.directory;
import com.newscale.nsapi.directory.person.Person;
import com.newscale.nsapi.NSApiConstants;
import com.newscale.nsapiclient.directory.Directory;
import com.newscale.nsapiclient.NSApiClientFactory;
import com.newscale.nsapiclient.NSApiClient;
import java.util.List;
import org.apache.commons.collections.map.MultiValueMap;
.  .  .
NSApiClient nsApiClient = NSApiClientFactory.getInstance();
nsApiClient.login("http://<serverURL>/RequestCenter","username","password");

PersonList persons = nsApiClient.getDirectory().getPeople(null);

/*
 This is the equivalent of REST URL
    http://<serverURL>/RequestCenter/nsapi/directory/people
*/
.  .  .
nsApiClient.logout();
```

- クエリー フィルタによる個人の取得

```
package com.newscale.nsapiclient.directory;
import com.newscale.nsapi.directory.person.Person;
import com.newscale.nsapi.NSApiConstants;
import com.newscale.nsapiclient.directory.Directory;
import com.newscale.nsapiclient.NSApiClientFactory;
import com.newscale.nsapiclient.NSApiClientConstants;
import com.newscale.nsapiclient.NSApiClient;
import org.apache.commons.collections.map.MultiValueMap;

import java.util.List;

NSApiClient nsApiClient = NSApiClientFactory.getInstance();
nsApiClient.login("http://<serverURL>/RequestCenter","username","password");

MultiValueMap filterMap = new MultiValueMap();//this can be used to specify multiple
filter criteria
filterMap.put(NSApiClientConstants.QUERYPARAM_NAME, "John");
PersonList persons = nsApiClient.getDirectory().getPeople(filterMap);

/*
 This is the equivalent of REST URL
http://<serverURL>/RequestCenter/nsapi/directory/people?name=John
*/
.  .  .
nsApiClient.logout();
```

- 複数のクエリー フィルタによる個人の取得

```
package com.newscale.nsapiclient.directory;
import com.newscale.nsapi.directory.person.Person;
import com.newscale.nsapi.NSApiConstants;
import com.newscale.nsapiclient.directory.Directory;
import com.newscale.nsapiclient.NSApiClientFactory;
import com.newscale.nsapiclient.NSApiClientConstants;
import com.newscale.nsapiclient.NSApiClient;
import org.apache.commons.collections.map.MultiValueMap;

import java.util.List;
.  .  .
NSApiClient nsApiClient = NSApiClientFactory.getInstance();
```

```
nsApiClient.login("http://<serverURL>/RequestCenter","username","password");

MultiValueMap filterMap = new MultiValueMap();
filterMap.put(NSApiClientConstants.QUERYPARAM_NAME, "John");
filterMap.put(NSApiClientConstants.QUERYPARAM_SORTBY, "lastName");
filterMap.put(NSApiClientConstants.QUERYPARAM_SORTDIR, "asc");
PersonList persons = nsApiClient.getDirectory().getPeople(filterMap);

// This is the equivalent of REST URL
//    http://<serverURL>/RequestCenter/nsapi/directory/people?name
//    =John&sortBy=lastName&sortDir=asc
// search for people by the name John,
// sort the result set by Last Name in ascending order
.  .   .
nsApiClient.logout();
```

Post 操作

ここでは、個人の作成および更新方法、ならびにタスクに対するアクションの実行方法を示すサンプル コードの一部をいくつか示します。これらの方法の詳細については、個々のエンティティ クラスに関する Javadoc を参照してください。

- 個人の更新

```
package com.newscale.nsapiclient.directory;
import com.newscale.nsapi.directory.person.Person;
import com.newscale.nsapi.NSApiConstants;
import com.newscale.nsapiclient.directory.Directory;
import com.newscale.nsapiclient.NSApiClientFactory;
import com.newscale.nsapiclient.NSApiClient;
.  .   .
NSApiClient nsApiClient = NSApiClientFactory.getInstance();
nsApiClient.login("http://<serverURL>/RequestCenter","username","password");
Person person = NSApiClient.getDirectory().getPersonById(123);
person.setLastName("Smith");
Person persons = NSApiClient.getDirectory().updatePerson(person);
/*
This is the equivalent of posting XML of person 123 with last name changed to "Smith" to
the REST URL
http://<serverURL>/RequestCenter/nsapi/directory/people/update
*/
.  .   .
nsApiClient.logout();
```

- 配信タスクの完了

```
package com.newscale.nsapiclient.transaction;
import com.newscale.nsapiclient.transaction.task.TaskAction;
import com.newscale.nsapi.NSApiConstants;
import com.newscale.nsapiclient.transaction.Transaction;
import com.newscale.nsapiclient.NSApiClientFactory;
import com.newscale.nsapiclient.NSApiClientConstants;
import com.newscale.nsapiclient.NSApiClient;
import org.apache.commons.collections.map.MultiValueMap;

import java.util.List;
.  .   .
NSApiClient nsApiClient = NSApiClientFactory.getInstance();
nsApiClient.login("http://<serverURL>/RequestCenter","username","password");
```

```
// This is the equivalent of REST URL
//    http://<serverURL>/RequestCenter/nsapi/tasks/10/done

NSApiClient.getTransaction().completeTask(10);
nsApiClient.logout();
```

API リファレンスおよび例

nsAPI の javadoc は、製品インストーラの Image フォルダにあります。REST URL および java クライアントから nsAPI を呼び出す例を、サポートされているエンティティごとに示します。

定義データ

カテゴリ（Categories）

表 3-5　カテゴリ API テーブル

領域	例
Core API	**すべてのカテゴリの取得** デフォルトでは、タイプが「Service Catalog」のカテゴリのみが返されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/categories?catalogType=serviceCatalog Java の例 CategoryList categories = NSApiClient.getDefinition().getCategories(null);
	すべてのコンシューマ サービス カテゴリの取得 「Consumer Services」タイプのすべてのカテゴリが返されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/categories Java の例 `CategoryList categories = NSApiClient.getDefinition().getCategories("serviceCatalog");`
	Name によるコンシューマ サービス カテゴリの取得 指定した名前のサービス カタログ カテゴリが返されます。ネストされたエンティティ（サブカテゴリおよび組み込みサービス）が取得されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/categories/name/<categoryName> Java の例 `Category categories = NSApiClient.getDefinition().getCategoryByName("<categoryName>");`

表 3-5　カテゴリ API テーブル(続き)

領域	例

	Id によるコンシューマ サービス カテゴリの取得
	指定した Id のサービス カタログ カテゴリが返されます。ネストされたエンティティ(サブカテゴリおよび組み込みサービス)が取得されます。
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/definition/categories/id/<categoryId>
	Java の例
	`Category categories = NSApiClient.`
	`getDefinition().getCategoriesById(<categoryId>);`

	Name による提供サービス カテゴリの取得
	ネストされたエンティティ(サブカテゴリおよび組み込み提供)が含まれます。指定した名前のサービス提供カテゴリが返されます。
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/definition/categories/name/<categoryName>?catalogType=offeringCatalog
	Java の例
	`MultiValueMap paramsMap = new MultiValueMap();`
	`paramsMap.put(QUERYPARAM_CATALOG_TYPE, "offeringCatalog");`
	`Category categories =`
	`NSApiClient.getDefinition().getCategoryByName("<categoryName>");`

フィルタ	*カテゴリ名フィルタ*
	検索では大文字と小文字が区別されます。
	StartsWith(newscale.properties の ContainsQueryInFnS=false):先頭のワイルドカードは無視されます。
	Contains(newscale.properties の ContainsQueryInFnS=true):先頭のワイルドカードは適用されます。
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/definition/categories?name=<wildcardValue>&catalogType=serviceCatalog
	`Java Example:`
	`MultiValueMap paramsMap = new MultiValueMap();`
	`paramsMap.put(QUERYPARAM_CATALOG_TYPE, "serviceCatalog");`
	`paramsMap.put(QUERYPARAM_NAME, "<wildcardValue>");`
	`CategoryList categories =`
	`NSApiClient.getDefinition().getCategories(paramsMap);`

表 3-5　カテゴリ API テーブル（続き）

領域	例

	サービス ユーザ アクセス フィルタ
	オーダーする権限を持つログオン中のユーザについて、その直下またはその階層にあるサービスのすべてのカテゴリを返します。
	REST URL:
	http://<ServiceUrl>/RequestCenter/nsapi/definition/categories?userContext=<true or false>
	Java Example:
	`MultiValueMap paramMap = new MultiValueMap();`
	`paramMap.put(QUERYPARAM_USERCONTEXT, "true");`
	`NSApiClient.getDefinition().getCategories(paramMap);`

ソート カラム	CategoryName

応答 XML	`<categories totalCount="x" recordSize="y" startRow="z">`
	`<category>`
	`. . .`
	`<associatedServices>`
	`. . .`
	`<associatedService>`
	`<description> </description>`
	`<id></id>`
	`<imageURL></imageURL>`
	`<name> </name>`
	`<status> </status>`
	`</associatedService>`
	`. . .`
	`</associatedServices>`
	`. . .`
	`</category>`
	`</categories>`

環境

表 3-6　環境 API テーブル

領域	例
コア API	パブリック GUI ID を使用して AMQP 環境の詳細を取得（パブリック GUI ID：[管理（Administration）] > [設定（Settings）] に移動し、パブリック/プライベート キーを指定します）
	すべての AMQP 環境の詳細が返されます。
	POST REST URL：
	http://<ServerURL>/RequestCenter/nsapi/messagebroker/service/<serviceName>?publicKeyGUID=...
	http://<ServerURL>/RequestCenter/nsapi/messagebroker/overview?publicKeyGUID=...
	ステータス コード：200 OK
	ステータス コード：400 Bad request
	パブリック キー GUID を使用した暗号化クレデンシャルの詳細については、「**AMQP との統合**」の項を参照してください。
	プロバイダー タイプ、短縮名によりクラウド接続の詳細を取得
	指定した短縮名について、Base64 符号化形式の SSL 証明書を含む、クラウド タイプ（UCSD/ICFD/PUPPET など）の接続情報が返されます。
	POST REST URL：
	http://<ServerURL>/RequestCenter/nsapi/definition/connections/provider/{providerName}/shortName/{shortName}
	{Provider name} には UCSD/ICFD/PUPPET を指定でき、{psc} は接続に使用される短縮名 ID です。接続インシデントが https の場合、API は接続に使用するルート証明書を返します。接続の SSL 証明書（Base 64 形式）も返されます。
	ステータス コード：200 OK
	ステータス コード：400 Bad request
	サンプル応答：

	<pre>{
	"connections":
	{
	"connection":
	[
	{
	"primaryID": 2,
	"userName": "admin",
	"protocol": "https",
	"hostName": "172.21.38.151",
	"description": "...",
	"url": "https://172.21.38.151:443/app/api/rest",
	"port": 443,</pre>

表 3-6　環境 API テーブル(続き)

領域	例
	プロバイダー タイプ、パブリック GUI ID を使用した短縮名により、クラウドの接続詳細を取得 外部システムのパブリック キー GUID がクエリー パラメータとして渡された場合、応答内のクレデンシャル(パスワードとトークン)は GUID に関連付けられているパブリック キーにより暗号化されます(PO のセキュア ストリング形式を使用)。 POST REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/connections/provider/{providerName}/shortName/{shortName}?publicKeyGUID=.. パブリック キー GUID が渡されないか、間違っている場合、**ステータス コード**: 400 不正な要求が返されます。

表 3-6　環境 API テーブル(続き)

領域	例

	AMQP サーバ詳細の作成
	AMQP サーバとの SSL 接続および非 SSL 接続を作成し、Web インターフェイスで AMQP サーバの詳細を更新します。
	POST REST URL:
	http://<ServerURL>/RequestCenter/nsapi/ucsd/discovery/saveConnection
	SSL 証明書を使用する接続の場合:
	{
	"id": "row-1",
	"shortName": "am2",
	"description": "amqp_server2",
	"protocol": "SSL",
	"hostName": "10.78.0.247",
	"port": "5671",
	"un": "pscuser",
	"up": "!@^_cHNjdXNlcg==_^@!",
	"pollerEnabled": "0",
	"provider": "AMQP",
	"exportUser": "0",
	"cert": "-----BEGIN CERTIFICATE-----\nMIIC5DCCAcygAwIBAgIBATANBgkqhkiG9w0BAQsFADATMREwDwYDVQQDDAhNeVRlc3RDQTAeFw0x\nNjA2MTExMTUwMTJaFw0xNzA2MTExMTUwMTJaMCcxFDASBgNVBAMMCyQoaG9zdG5hbWUpMQ8wDQYD\nVQQKDAZzZXJ2ZXIwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCq8v2yJT7tv+nOwFSo\nEE1c0oVg3skd86JN7jVJaz/mMOyJjDmf1147iUwZPMTTCB34ovYUXFkw+a+0ext2WRHgQLTMPvVO\nnA86jwuPd/bhUxXg8jeEfE4V/1Seci9Xz+5VxqCybCNOzJQ12/vLXvIJK43U/+1GdXnpWxFaF0yd0\nnht3iUy6mfUAHfNMI2SOfJwbbdUaMyD0/Krsiu+X+vFQBUDmM7Y0priItiDVDq7rxug2UOPACzzMG\n5yJ5aJjNLSlJRwKKst/jjvesqHIgWNo0qKvkTET3tIVsKDi1Fn9IdKQuuoI1n225+58cWSANmZ5M\n4BdSIf4z6QRuKliBRi55AgMBAAGjLzAtMAkGA1UdEwQCMAAwCwYDVR0PBAQDAgUgMBMGA1UdJQQM\nMAoGCCsGAQUFBwMBMA0GCSqGSIb3DQEBCwUAA4IBAQC4L9fk4kuu/dpLeFWNUhb5Uyk6AqbTRAyD\nnlq1m11E09EhmTH7cnoFsz0ELBDppASIUADSb9jmUeNKJtjW94gq8luSem1Z8lQzUCOCE6rsaznrw\nnm9jJO7gXA5SSmy7PgdokdhbeTzlSYA3kkkR5ZE8M403Qv5cEPREqnshs6f6bQSm8tzSNETy3OyHL\npUo7YVTBCfJMQ/e2nZxCJSDuEL6QaIL4kmEYeu8j/1RplBAofMkDfDe+yMx2MJYl+MopVggGexpa\nnMqybCchrDTJ/8sI/R18Ld80TQ2Km70sQqvNVenCpGCtgGIcupRWaAOpsQH0PNaUK+lcwYRIvf0VS\n09QE\n-----END CERTIFICATE-----",
	"ovCert": "",
	"skipValidateCertificate": "0",
	"virtualHost": "psc",
	"publicKey": "none",
	"secureStringFormat": "Bytes",
	"serverDownNotification": "none",
	"recoveryInterval": "5",
	"inboundQueue": "psc_inbound_queue",
	"messageType": "XML",
	"basicAuthentication": "0",
	"isTriggeredByPoller": false
	}

表 3-6　環境 API テーブル(続き)

領域	例

	SSL 証明書のない接続の場合:
	{
	"id": "row-1",
	"shortName": "am1",
	"description": "amqp_server1",
	"protocol": "TCP",
	"hostName": "10.78.0.247",
	"port": "5672",
	"un": "admin",
	"up": "!@^_Y2lzY28xMjM=_^@!",
	"pollerEnabled": "0",
	"provider": "AMQP",
	"exportUser": "0",
	"cert": "",
	"ovCert": "",
	"skipValidateCertificate": "0",
	"virtualHost": "default",
	"publicKey": "none",
	"secureStringFormat": "Bytes",
	"serverDownNotification": "none",
	"recoveryInterval": "5",
	"inboundQueue": "psc_inbound_queue",
	"messageType": "XML",
	"basicAuthentication": "0",
	"isTriggeredByPoller": false
	}
	(注)　パスワードの値は、Base64 で暗号化された値として、プレフィックス !@^_ およびサフィックス _^@! が付いた形で渡されます。Base64 で暗号化された値は、JavaScript btoa メソッドを使用して生成できます。
	
	(注)　ShortName の長さは最大 3 文字です。

	AMQP 接続の削除
	POST REST URL:
	http://<Server URL>/RequestCenter/nsapi/ucsd/discovery/deleteConnection
	{
	"shortName": "am3",
	"provider": "AMQP"
	"protocol": "TCP"
	}

ローカリゼーション

表 3-7　ローカリゼーション API テーブル

領域	例
コア API	**すべての言語をロケール詳細とともに取得** すべての言語をロケールの詳細とともに返します。 REST URL: http://<ServerURL>/RequestCenter/nsapi/localization/locales サンプル応答: [{ "active": 1, "defaultCurrency": "REAL", "defaultDateSep": "/", "defaultLongDate": "dd 'de' MMMM 'de' yyyy", "defaultShortDate": "dd/MM/yy", "defaultTime": "HH:mm:ss", "localeCode": "BRPT", "localeID": 9, "name": "Brazilian-Portuguese" }, { "active": 1, "defaultCurrency": "YEN", "defaultDateSep": "/", "defaultLongDate": "MMMM dd, yyyy", "defaultShortDate": "MM/dd/yyyy", "defaultTime": "h:mm aa", "localeCode": "CNZH", "localeID": 6, "name": "Chinese (Simplified)" }, ----- -----]

表 3-7　ローカリゼーション API テーブル(続き)

領域	例
	ロケールに固有な JavaScript 文字列の取得 入力で渡されるロケールに固有な JavaScript 文字列を返します。デフォルトでは、英語ロケールの JavaScript 文字列が返されます。要求で不明なロケールが渡された場合、既知のロケールに関する応答を返し、不明なロケールに関するエラー応答は表示しません。 REST URL: http://<ServerURL>/RequestCenter/nsapi/localization/javascriptstrings?locales=KRKO, ALSQABC, BRBT, CNZH サンプル応答: [{ "StringID": "js1", "US English": "src_eng22", "Chinese (Simplified)": "", "Brazilian-Portuguese": "", "Korean": "update5" }, { "StringID": "js2", "US English": "src_eng2", "Chinese (Simplified)": "", "Brazilian-Portuguese": "", "Korean": "5" }, -------- ---------]

表 3-7　ローカリゼーション API テーブル（続き）

領域	例
	すべての使用可能な言語のすべての使用可能な JavaScript 文字列の取得 すべての使用可能な言語について、すべての使用可能な JavaScript 文字列を返します。つまり、言語の数に制限はありません。 REST URL: http://<ServerURL>/RequestCenter/nsapi/localization/javascriptstrings サンプル応答: [{ "StringID": "js1", "US English": "src_eng22", "German": "", "Spanish": "", "French": "js string french", "Korean": "update5" }, { "StringID": "js2", "US English": "src_eng2", "German": "", "Spanish": "", "French": "french js2", "Korean": "5" }, ---------- ----------]

表 3-7　ローカリゼーション API テーブル(続き)

領域	例
	すべての使用可能な言語のすべての使用可能な JavaScript 文字列をポスト REST URL: http://<ServerURL>/RequestCenter/nsapi/localization/javascriptstrings サンプル要求: [{ "StringID": "js1", "US English": "src_eng22", "German": "", "Spanish": "", "French": "js string french", "Korean": "update5" }, { "StringID": "js2", "US English": "src_eng2", "German": "", "Spanish": "", "French": “french js2", "Korean": "5" }, ---------- ----------] サンプル成功応答: JavaScript 文字列の作成/更新が正常に完了しました。 サンプルエラー応答: JSON コンテンツ内の言語名は、この製品では、スペルが誤っているか、利用できません。「French (Switzerland)」で確認してください。

サービス

表 3-8　サービス API テーブル

領域	例
コア API	**すべてのサービスの取得** すべてのサービスが返されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/servicedefs Java の例 `ServiceList services = NSApiClient.getDefinition().getServices(null);`

	サービス カタログのワイルドカード検索によるすべてのサービスの取得
	サービスの名前、サービスの説明、カテゴリ、またはキーワードが検索文字列と一致しているサービスがすべて返されます。先頭のワイルドカードはサポートされません。
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/definition/servicedefs?search={wildcardValue*}
	Java の例
	`MultiValueMap paramsMap = new MultiValueMap();`
	`paramsMap.put("search", "<wildcardValue*>");`
	`ServiceList services = NSApiClient.getDefinition().getServices(paramsMap);`

	Id によるサービスの取得 ネストされたエンティティ(関連付けられているカテゴリおよびキーワード)は、getById および getByName の場合にのみ取得されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/servicedefs/id/<serviceId> Java の例 `Service services = NSApiClient.getDefinition().getServiceById(<serviceId>);`

	Name によるサービスの取得
	ネストされたエンティティ(関連付けられているカテゴリおよびキーワード)は、getbyId および getByName の場合にのみ取得されます。
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/definition/servicedefs/name/<serviceName>
	Java の例
	`Service services =`
	`NSApiClient.getDefinition().getServiceByName("<serviceName>");`

表 3-8　サービス API テーブル（続き）

領域	例
	カテゴリに含まれるすべてのサービスの取得 そのカテゴリに関連付けられているすべてのサービスが返されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/servicedefs?categoryName=<categoryName> Java の例 `ServiceList services =NSApiClient.getDefinition().getServiceByCategoryName("<categoryName>");`
	キーワードによるすべてのサービスの取得 そのキーワードに関連付けられているすべてのサービスが返されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/servicedefs?keywordName=<keyword> Java の例 `ServiceList services = NSApiClient.getDefinition().getServiceByKeyword("<keyword>");`

フィルタ	**サービス名フィルタ**
	検索では大文字と小文字が区別されます。
	StartsWith（newscale.properties の ContainsQueryInFnS=false）：先頭のワイルドカードは無視されます。
	Contains（newscale.properties の ContainsQueryInFnS=true）：先頭のワイルドカードは、必要に応じて指定してください。
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/definition/servicedefs?name=<wildcardValue>
	Java の例
	`MultiValueMap paramsMap = new MultiValueMap();`
	`paramsMap.put("name","<wildcardValue>");`
	`ServiceList servicesList = NSApiClient.getDefinition().getServices(paramsMap)`

ソート カラム	ServiceName

応答 XML	`<services totalCount="x" recordSize="y" startRow="z">`
	`  <service>`
	`  .  .  .`
	`  <includedServices>`
	`  .  .  .`
	`  <includedService>`
	`  <id></id>`
	`  <name> </name>`
	`  </includedService>`
	`  .  .  .`
	`  </includedServices>`
	`  .  .  .`
	`  </service>`
	`</services>`

表 3-8　サービス API テーブル(続き)

領域	例

CategoriesExtensions およびファセット	カテゴリ、ファセット、および拡張子によりサービスを作成/更新します。
	新しいサービスに POST を使用 - サービスがすでに存在する場合、エラーが発生します。カテゴリ、ファセット、および拡張子を作成/更新し、それをサービスに関連付けます。
	既存のサービスを更新するために PUT を使用 - サービスが存在しない場合、エラーが発生します。カテゴリ、ファセット、および拡張子を作成/更新して、それをサービスに関連付けます。
	REST URL:
	/RequestCenter/nsapi/cloud/marketplace/service
	サンプル ペイロード:
	{
	"***serviceName***": "TestServiceT225", "***serviceId***": "...", "***templateName***": "TestService", "***templateCategory***" : "TemplateCategory", "***description***": "test", "imageData": "", "imageURL": "$RC_IMAGEPATH$/ABCServer.jpeg", "overview": "<p>Product Details<\/strong>
	Version: 1.23.0-0 on Ubuntu 12.04.4
	", "serviceGroupName": "TestGrp",
	"**categories**": [
	{
	"name": "TestABCImageURL2", "description" : "TestDesc", "imageURL": "$RC_IMAGEPATH$/ABCServer.jpeg"}
	],
	"**facetes**": {
	"facetlogicname1" : {"displayName" : "Facet 1", "values" : ["f1val1", "f1val2", "f1val3", "f1val4"]},
	"facetlogicname2" : {"displayName" : "Facet 2", "values" : ["f2val1", "f2val2"]}
	},
	"**extensions**": {
	"extensionlogicname1" : {"displayName" : "Extension 1", "values" : "extval1"},
	"extensionlogicname2" : {"displayName" : "Extension 2", "values" : "extval2"}
	}
	}
	サンプルの応答 XML:
	{ "JSONObject":
	{
	"serviceName": "TestNewJSON1-TestNew",
	"serviceEndpoint":
	"/RequestCenter/nsapi/definition/servicedefs/name/TestNewJSON1-TestNew"
	}
	}

表 3-8　サービス API テーブル（続き）

領域	例

	サービスのカテゴリのリンクを解除する
	POST REST URL:
	http://<Server URL>/RequestCenter/nsapi/definition/servicedefs/id/{serviceId}/delinkCategories
	サンプル ペイロード:
	{"categories" :["catname1","catname2",...]}
	ステータス コード:200 OK
	ステータス コード:400 Bad request
	サンプル応答:
	{
	status-message:
	{
	code: "...."
	value: "....."
	}
	}

表 3-8　サービス API テーブル(続き)

領域	例

	さまざまなシナリオにおける応答:
	1. 渡された serviceId が存在しない場合:
	HTTP コード:404
	HTTP 応答:
	{
	nsapi-error-response:
	{
	errorcode: "SERV_0001"
	errormessage: "ServiceId does not exist."
	}
	}
	2. 渡されるカテゴリのいずれかがサービスと関連付けられていない場合:
	HTTP コード:404
	HTTP 応答:
	{
	nsapi-error-response:
	{
	errorcode: "SERV_0003"
	errormessage: "Categories could not be unlinked from the Service."
	}
	}
	3. すべての汎用例外で、JSON 応答は次のようになります。
	HTTP コード:500
	HTTP 応答:
	{
	nsapi-error-response:
	{
	errorcode: "EXC_0001"
	errormessage: "Some NSAPI Exception Occurred."
	}
	}
	4. ログイン ユーザに serviceId で識別されるサービスに対する RBAC 権限がない場合:
	HTTP コード:401
	HTTP 応答:
	{
	nsapi-error-response:
	{
	errorcode: "NSAPI_ERROR_004"
	errormessage: "The user does not have sufficient permission to perform the operation on this object."
	}
	}

表 3-8　サービス API テーブル（続き）

領域	例

	5. カテゴリのリンク解除が正常に行われた場合:
	HTTP コード:200
	HTTP 応答:
	{
	status-message:
	{
	code: "Success"
	value: "Categories Unlinking is done Successfully for the Service."
	}
	}

表 3-8　サービス API テーブル(続き)

領域	例

	サービスからのファセットのリンク解除
	POST REST URL:
	http://<Server URL>/RequestCenter/nsapi/definition/servicedefs/id/{serviceId}/delinkFacets
	ペイロードはファセット名(<facetName> は表示名ではなく、ファセットの名前)と、次の JSON 配列形式の値をとります。
	{
	"<facets>" :["<facet1>","<facet2>",...]
	}
	さまざまなシナリオにおける応答:
	1. 渡された serviceId が存在しない場合:
	HTTP コード:404
	HTTP 応答:
	{
	nsapi-error-response:
	{
	errorcode: "SERV_0001"
	errormessage: "ServiceId does not exist."
	}
	}
	2. ファセットがサービスに関連付けられていない場合:
	HTTP コード:404
	HTTP 応答:
	{
	nsapi-error-response:
	{
	errorcode: "SERV_0005"
	errormessage: "Facets could not be unlinked from the Service."
	}
	}
	3. ログイン ユーザに serviceId で識別されるサービスに対する RBAC 権限がない場合:
	HTTP コード:401
	HTTP 応答:
	nsapi-error-response:
	{
	errorcode: "NSAPI_ERROR_004"
	errormessage: "The user does not have sufficient permission to perform the operation on this object"
	}
	}

表 3-8　サービス API テーブル（続き）

領域	例

	4. すべての汎用例外で、JSON 応答は次のようになります。
	HTTP コード:500
	HTTP 応答:
	{
	nsapi-error-response:
	{
	errorcode: "EXC_0001"
	errormessage: "Some NSAPI Exception Occurred."
	}
	}
	5. ファセットのリンク解除が正常に行われた場合:
	HTTP コード:200
	HTTP 応答:
	{
	status-message:
	{
	code: "Success"
	value: "Facets Unlinking is done Successfully for the Service."
	}
	}

表 3-8　サービス API テーブル(続き)

領域	例

	複数値のファセットから値をリンク解除します。この API は複数値のファセットに対してのみ適用できます。
	POST REST URL:
	http://<Server URL>/RequestCenter/nsapi/definition/servicedefs/id/{serviceId}/delinkMultiValueFacet
	ペイロードはファセット名(<facetName> は表示名ではなく、ファセットの名前)と、次の JSON 配列形式の値をとります。
	{
	"<facetName>" :["<facetValue1>","<facetValue2>",...]
	}
	さまざまなシナリオにおける応答:
	1. 渡された serviceId が存在しない場合:
	HTTP コード:404
	HTTP 応答:
	{
	nsapi-error-response:
	{
	errorcode: "SERV_0001"
	errormessage: "ServiceId does not exist."
	}
	}
	2. ファセット値がサービスに関連付けられていない場合:
	HTTP コード:404
	HTTP 応答:
	{
	nsapi-error-response:
	{
	errorcode: "SERV_0005"
	errormessage: "Facet values could not be unlinked from the Service."
	}
	}

表 3-8　サービス API テーブル（続き）

領域	例

	3. ログイン ユーザに serviceId で識別されるサービスに対する RBAC 権限がない場合:
	HTTP コード:401
	HTTP 応答:
	nsapi-error-response:
	{
	errorcode: "NSAPI_ERROR_004"
	errormessage: "The user does not have sufficient permission to perform the operation on this object"
	}
	}
	4. すべての汎用例外で、JSON 応答は次のようになります。
	HTTP コード:500
	HTTP 応答:
	{
	nsapi-error-response:
	{
	errorcode: "EXC_0001"
	errormessage: "Some NSAPI Exception Occurred."
	}
	}
	5. ファセット値の解除が正常に行われた場合:
	HTTP コード:200
	HTTP 応答:
	{
	status-message:
	{
	code: "Success"
	value: "Facet Values Unlinking is done Successfully for the Service."
	}
	}

表 3-8　サービス API テーブル(続き)

領域	例

	サービスからサービス拡張子をリンク解除する
	POST REST URL:
	http://<Server URL>/RequestCenter/nsapi/definition/servicedefs/id/{serviceId}/delinkServiceExtensions
	ペイロードは次の JSON 配列形式のサービス拡張子名(<serviceExtensions> は表示名ではなく、論理名)をとります。
	{
	"serviceExtensions" : ["serExt1","serExt2",...]}
	さまざまなシナリオにおける応答:
	1. 渡された serviceId が存在しない場合:
	HTTP コード:404
	HTTP 応答:
	{
	nsapi-error-response:
	{
	errorcode: "SERV_0001"
	errormessage: "ServiceId does not exist."
	}
	}
	2. サービス拡張子がサービスに関連付けられていない場合:
	HTTP コード:404
	HTTP 応答:
	{
	nsapi-error-response:
	{
	errorcode: "SERV_0004"
	errormessage: "Servce Extensions could not be unlinked from the Service."
	}
	}
	3. ログイン ユーザに serviceId で識別されるサービスに対する RBAC 権限がない場合:
	HTTP コード:401
	HTTP 応答:
	nsapi-error-response:
	{
	errorcode: "NSAPI_ERROR_004"
	errormessage: "The user does not have sufficient permission to perform the operation on this object"
	}
	}

表 3-8　サービス API テーブル（続き）

領域	例

	4. すべての汎用例外で、JSON 応答は次のようになります。
	HTTP コード:500
	HTTP 応答:
	{
	nsapi-error-response:
	{
	errorcode: "EXC_0001"
	errormessage: "Some NSAPI Exception Occurred."
	}
	}
	5. サービス拡張子のリンク解除が正常に行われた場合:
	HTTP コード:200
	HTTP 応答:
	{
	status-message:
	{
	code: "Success"
	value: "Service Extensions Unlinking is done Successfully for the Service."
	}
	}

表 3-8　サービス API テーブル(続き)

領域	例

	サービスの削除
	この API は次の URL で削除するサービスの ID をとります。
	http://<Server URL>/RequestCenter/nsapi/definition/servicedefs/id/{id}
	HTTP メソッド:DELETE
	1. サービスの削除が正常に終了した場合:
	HTTP コード:200
	HTTP 応答:
	{
	status message:
	{
	code: “Success”
	"value": "Service with ServiceID {id} deleted successfully.”
	}
	}
	2. 渡された serviceId が存在しない場合:
	HTTP コード:404
	HTTP 応答:
	{
	nsapi-error-response:
	{
	errorcode: "SERV_0001"
	errormessage: "ServiceId does not exist."
	}
	}
	3. サービスに要求があり、削除できない場合
	HTTP コード:403
	HTTP 応答:
	{
	nsapi-error-response:
	{
	errorcode: "SERV_0002"
	errormessage: "Service Cannot be deleted since it already exists amongst requisitions"
	}
	}

表 3-8　サービス API テーブル（続き）

領域	例

	4. 一般的な例外
	HTTP コード:500
	HTTP 応答:
	{
	nsapi-error-response:
	{
	errorcode: "EXC_0001"
	errormessage: "Delete Service Failed"
	}
	}
	5. ログイン ユーザに serviceId で識別されるサービスに対する RBAC 権限がない場合:
	HTTP コード:401
	HTTP 応答:
	nsapi-error-response:
	{
	errorcode: "NSAPI_ERROR_004"
	errormessage: "The user does not have sufficient permission to perform the operation on this object"
	}
	}

表 3-8　サービス API テーブル(続き)

領域	例

REST API	**サービスのコピー**
	既存のアクティブ フォーム コンポーネント(AFC)とディクショナリを再利用して、またはサービスを AFC のディクショナリ、電子メール テンプレート、およびスクリプトとともに複製することによって、サービス定義をコピーします。
	POST REST URL:
	http://<ServerURL>/RequestCenter/nsapi/definition/v1/service
	サービスのコピー:
	既存のアクティブ フォーム コンポーネント(AFC)およびディクショナリを再利用することで、サービス定義をコピー。
	入力例:
	`{`
	`"Service ID": "88",`
	`"New Service Name": "service",`
	`"Deep Copy": "false",`
	`}`
	サービスの複製:
	サービス内のプレフィックスを持つフォームおよびディクショナリをすべてコピー。
	入力例:
	`{`
	`"Service ID": "88",`
	`"New Service Name": "service",`
	`"Deep Copy": "true",`
	`"Form Name Prefix": "NewForm",`
	`"Dictionary Name Prefix": "NewDictionary"`
	`}`
	サービス内の必要なフォームおよび AFC のみをコピー。
	入力例:
	`{`
	`"Service ID": "88",`
	`"New Service Name": "service",`
	`"Deep Copy": "true",`
	`"Forms":{`
	`"Old form name":{`
	`"New Form Name": "Deep Copy Form",`
	`"New Form Group Name": "Deep copy form group",`
	`"Dictionaries":{`
	`"Old dictionary name":{`
	`"New Dictionary Name": "Deep_Copy_Dictionary",`
	`"New Dictionary Group Name": "new group"`
	`}`
	`}`
	`}`
	`},`
	`}`

エージェント（Agents）

表 3-9　エージェント API テーブル

領域	例
コア API	エージェントの停止 アクティブなサービス リンク エージェントを停止します。 POST REST URL： http://<ServerURL>/RequestCenter/nsapi/definition/agent/<agent name>/stop
	エージェントの起動 非アクティブなサービス リンク・エージェントを起動します。 POST REST URL： http://<ServerURL>/RequestCenter/nsapi/definition/agent/<agent name>/start
応答 XML	`Agent<agent name> <started/stopped> successfully`

表 3-10　エージェント API テーブル

領域	例
コア API	**すべてのエージェントの取得** すべてのエージェントが返されます。 GET REST URL： http://<ServerURL>/RequestCenter/nsapi/definition/agents Java の例 `AgentList agents = NSApiClient.getDefinition().getAgents(null);`
	Id によるエージェントの取得 発信および着信のプロパティは、getById および getByName の場合にのみ取得されます。 GET REST URL： http://<ServerURL>/RequestCenter/nsapi/definition/agents/id/<agentId> Java の例 `Agent agents = NSApiClient.getDefinition().getAgentById(<agentId>);`
	Name によるエージェントの取得 発信および着信のプロパティは、getById および getByName の場合にのみ取得されます。 GET REST URL： http://<ServerURL>/RequestCenter/nsapi/definition/agents/name/<agentName> Java の例 `Agent Agents = NSApiClient.getDefinition().getAgentByName("<agentName>");`

表 3-10　エージェント API テーブル（続き）

領域	例
	エージェントのステータスの取得 エージェントのステータスを取得します。 GET REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/agent/<agentName>/status
フィルタ	**エージェント名フィルタ** 検索では大文字と小文字が区別されます。 StartsWith（newscale.properties の ContainsQueryInFnS=false）：先頭のワイルドカードは無視されます。 Contains（newscale.properties の ContainsQueryInFnS=true）：先頭のワイルドカードは、必要に応じて指定してください。 REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/agents?name=<wild cardValue> Java の例 `AgentList agents = NSApiClient.getDefinition().getAgentsByFilter("<wildcardValue>");`
ソート カラム	AgentName
応答 XML	`<agents totalCount="x" recordSize="y" startRow="z"> <agent> . . . </agent> </agents>`

契約

表 3-11　契約 API テーブル

領域	例
APIs の取得	**Id による契約の取得** REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/agreements/id/<agreementId>
	Name による契約の取得 REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/agreements/name/<agreementName>
	すべてのマスター契約の取得 すべてのマスター契約が返されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/agreements/master

表 3-11　契約 API テーブル(続き)

領域	例
	すべてのプロジェクト契約の取得 すべてのプロジェクト契約が返されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/agreements/project
	アカウント名のすべての契約を取得 そのアカウント名のすべての契約が返されます(完全一致、大文字小文字は区別されません)。 REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/agreements/accountname/{accountName}
	アカウント名の契約の取得 そのアカウント名の契約が返されます(アカウント名はこの領域で必須です)。 REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/agreements/name/{agreementname}?&accountName={accountName}
ソート カラム(s)	AgreementName
応答 XML	<agreements totalCount="x" recordSize="y" startRow="z"> <agreement> . . . </agreement> </agreements>

表 3-12　使用シナリオ テーブル

領域	例
他の使用シナリオ	アカウント下のすべての OU の取得 このアカウントで関連付けられているすべての OU が返されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/agreements/agreementorgunitlist?&accountName={accountName}
	契約の作成 REST URL (HTTP POST): http://<ServerURL>/RequestCenter/nsapi/definition/agreements/create <agreement agreementId="123" name="Sample Agreement Name"> parentAgreementName="Sample Parent Agreement Name" > <description>Sample Agreement Description</description> <agreementTemplateName>Sample Agreement Template Name</agreementTemplateName> </agreement>

表 3-12　使用シナリオ テーブル(続き)

領域	例
	契約の削除 REST URL (HTTP DELETE): http://<ServerURL>/RequestCenter/nsapi/definition/agreements/delete <agreement agreementId="123" name="Sample Agreement Name"> </agreement>
	契約の更新 REST URL (HTTP PUT): http://<ServerURL>/RequestCenter/nsapi/definition/agreements/update <agreement agreementId="123" name="Sample Agreement Name" accountName="Sample Account Name" parentAgreementName="Sample Parent Agreement Name"> <description>Sample Agreement Description</description> <agreementAttributes> <agreementAttribute serviceItemTypeName="Sample Service Item Type Name 1" aggregateFunction="Count" serviceItemAttribute="" quota="123" /> <agreementAttribute serviceItemTypeName="Sample Service Item Type Name 2" aggregateFunction="Count" serviceItemAttribute="" quota="456" /> <agreementAttribute serviceItemTypeName="Sample Service Item Type Name 2" aggregateFunction="Sum" serviceItemAttribute="Sample Attribute Name" quota="789" /> </agreementAttributes> <organizationalUnits> <organizationalUnit> <organizationalUnitName>Sample OU Name</organizationalUnitName> <organizationalUnitType>Business Unit</organizationalUnitType> </organizationalUnit> </organizationalUnits> </agreement>

課金レート

課金レート グループ

表 3-13　課金レート グループ API テーブル

領域	例
Core API	すべての課金レート グループの取得 すべての課金レート グループが返されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/billing/rategroups

表 3-13　課金レート グループ API テーブル（続き）

領域	例
	ID によるレート グループの取得 ID により課金レート グループが返されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/billing/rategroups/id/{id}
	名前によるレート グループの取得 名前により課金レート グループが返されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/billing/rategroups/name/{name}
	ワイルド カード名前検索によるレート グループの取得 ワイルド カード名前検索により課金レート グループが返されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/billing/rategroups?name={wildcard name}
	レート グループの作成（1 レート グループ オブジェクトのみ） REST URL (HTTP POST): http:// <ServerURL>/RequestCenter/nsapi/definition/billing/rategroups/create POST DATA: <rateGroup name=”Sample Rate Group” description=”Sample rate group description”> <rateTables> <rateTable id=”1” name=”First”/> <rateTable id=”2” name=”Second”/> <rateTable id=”3” name=”Third”/> </rateTables> </rateGroup>

表 3-13　課金レート グループ API テーブル(続き)

領域	例
	レート グループの更新(1 レート グループ オブジェクトのみ) REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/billing/rategroups/update 本文 <rateGroup id=”123” name=”Sample Rate Group” description=”Sample rate group description”> <rateTables> <rateTable id=”1” name=”First”/> <rateTable id=”2” name=”Second”/> <rateTable id=”3” name=”Third”/> </rateTables> </rateGroup>
	レート グループの削除(1 レート グループ オブジェクトのみ) REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/billing/rategroups/delete BODY: <rateGroup id=”123” name=”Sample Rate Group Name” />

課金レート定義

表 3-14　課金レート定義 API テーブル

領域	例
	グループ名によるすべての課金レート テーブルの取得 ID、名前、説明、サービス項目タイプの ID と名前を持つレート テーブルの軽量オブジェクトが返されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/billing/ratetables/rategroupname/{groupName}
	グループ ID によるすべての課金レート テーブルの取得 ID、名前、説明、サービス項目タイプの ID と名前を持つレート テーブルの軽量オブジェクトが返されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/billing/ratetables/rategroupid/{groupID}

表 3-14　課金レート定義 API テーブル（続き）

領域	例
	名前によるレート テーブル定義またはスキーマ詳細の取得 名前によるレート テーブル定義またはスキーマ詳細が返されます。属性と操作情報が付いた完全なレート テーブル定義が返されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/billing/ratetables/schema/name/{name}
	ID によるレート テーブル定義またはスキーマ詳細の取得 属性と操作情報が付いた完全なレート テーブル定義が返されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/billing/ratetables/schema/id/{id}
	ワイルドカード名前検索によるレート テーブルの取得 ID、名前、説明、グループ ID と名前、サービス項目タイプ ID と名前を持つレート テーブルの軽量オブジェクトが返されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/billing/ratetables/schema?name={wild card name}
	レート テーブル スキーマの作成（1 レート テーブル オブジェクトのみ） REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/billing/ratetables/schema/create <rateTable id=”" name=”Sample Rate Table” logicName=”SampleRateTable” description=”Sample Rate Table Description”> <rateGroup id=”123” name=”Sample Rate Group Name”/> <serviceItemType id=”123” name=”Sample Service Item Type Name”/> <billingAttributes> <attribute name="Attribute Name 1" billingField=”true” memoField=”false”/> <attribute name="Attribute Name 2" billingField =”false” memoField =”false”/> <attribute name="Attribute Name 3" billingField =”true” memoField =”false”/> </billingAttributes> <billingOperations> <operation name="Operation1" applicable=”false”/> <operation name="Operation2" applicable =”true”/> <operation name="Operation3" applicable =”true”/> </billingOperations> </rateTable>

表 3-14　課金レート定義 API テーブル(続き)

領域	例
	レート テーブル スキーマの更新(1 レート テーブル オブジェクトのみ) 課金レート テーブル スキーマを更新します(1 レート テーブル オブジェクトのみ)。 REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/billing/ratetables/schema/update <rateTable id=”123" name=”Sample Rate Table” logicName=”BiSampleRateTable” description=”Sample Rate Table Description”> <rateGroup id=”123” name=”Sample Rate Group Name”/> <serviceItemType id=”123” name=”Sample Service Item Type Name”/> <billingAttributes> <attribute name="Attribute Name 1" billingField=”true” memoField=”false”/> <attribute name="Attribute Name 2" billingField =”false” memoField =”false”/> <attribute name="Attribute Name 3" billingField =”true” memoField =”false”/> </billingAttributes> <billingOperations> <operation name="Operation1" applicable=”false”/> <operation name="Operation2" applicable =”true”/> <operation name="Operation3" applicable =”true”/> </billingOperations> </rateTable
	レート テーブル スキーマの削除(1 レート テーブル オブジェクトのみ) REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/billing/ratetables/schema/delete <rateTable id=”123" name=”Sample Rate Table” logicName="BiSampleRateTable" />

課金レート データ

表 3-15　課金レート データ API テーブル

領域	例
	レート テーブル名によるレート テーブル データの取得 レート テーブル 名により課金レート テーブルのデータが返されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/billing/ratetables/data/tablename/{tableName}

表 3-15　課金レート データ API テーブル(続き)

領域	例
	レート テーブル ID によるレート テーブルの取得 レート テーブル ID により課金レート テーブルのデータが返されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/billing/ratetables/data/tableid/{tableID}
	レート テーブル データの更新(1 レート テーブル オブジェクトのみ) レート テーブル データの既存のエントリが消去され、特定の入力データを挿入します(「置換」などの作業)。 REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/billing/ratetables/data/update <rateTableData id=”123” name=”Sample Rate Table” logicName=”BiSampleRateTable”> <rateTableRecord rate=”123.45” rateCode=”RateCode1” unitOfMeasure=”Per Month”> <attributes> <attribute name="Attribute Name 1">Value1</attribute> <attribute name="Attribute Name 2">Value2</attribute> <attribute name="Attribute Name 3">Value3</attribute> </attributes> </rateTableRecord> <rateTableRecord rate=”456.78” rateCode=”RateCode2” unitOfMeasure=”Per Week”> <attributes> <attribute name="Attribute Name 11">Value11</attribute> <attribute name="Attribute Name 12">Value12</attribute> <attribute name="Attribute Name 13">Value13</attribute> </attributes> </rateTableRecord> </rateTableData

表 3-15　課金レート データ API テーブル(続き)

領域	例
フィルタ	• URL で許可されるパラメータ:URL で許可されるパラメータは startRow、recordSize、sortBy、sortDir、および responseType（xml または json）です。 • 「RateGroup」および「RateTable」での検索 URL:ソートを実行するカラムとソート順を指定するには、「sortBy」および「sortDir」クエリー パラメータを使用できます。「sortBy」が空または無効なカラム名である場合、デフォルトのソートは「ID」に対する「昇順」です。 • 「RateTableData」での検索 URL:デフォルトのソートは「ID」に対する「降順」です。「sortBy」および「sortDir」クエリー パラメータはサポートされません。 • ワイルド カード検索では「starts-with」が使用でき、パフォーマンス上の理由から contains クエリーおよび先頭のワイルド カード検索はできません。 • レート テーブル スキーマの作成/更新では、レート グループおよびサービス項目タイプは名前に基づきます。ID がある場合、ID と名前の組み合わせのクロスチェックを試みた後に、更新または削除を実行します。 • レート グループまたはレート テーブルの更新/削除では、ID があれば ID に基づきます。ID がない場合は名前に基づきます。 • レート テーブル スキーマ検索では、属性は「課金」または「メモ」フィールドとして選択されている場合にのみ取得されます。また、操作は適用可能と選択された場合にのみ取得されます。 • レート テーブル スキーマの作成では、レート グループ名およびサービス項目タイプ名は必須であり、ID は任意選択です。ID がある場合は、レート テーブルの作成前に ID と名前の組み合わせがクロスチェックされます。 • レート テーブル スキーマの更新では、属性または操作は、それらが入力オブジェクトに指定された場合にのみ更新されます（選択または選択解除）。そうでない場合は、既存の属性や操作を変更しないでそのままにしておきます。 • レート テーブル データの更新では、レート テーブル データの既存のレコードがすべて削除され、入力 XML/JSON のレコードがレート テーブル データとして保存されます。 • 課金履歴ポートレット UI のすべての検証が、すべての作成、更新、削除に対して実行されます。
ソート カラム	ソートは URL で許可される 2 つのカラム（「ID」または「名前」）で可能です。

表 3-15　課金レート データ API テーブル(続き)

領域	例

応答 XML	軽量レート グループ リストの XML:
	`<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>`
	<rateGroups recordSize="2" startRow="1" totalCount="2">
	`<rateGroup id="11" name="Sample Rate Group 1" description="First group" />`
	`<rateGroup id="22" name="Sample Rate Group 2" description="Second group" />`
	`</rateGroups>`
	
	単一 レート グループ取得の XML
	レート グループの更新要求のペイロードと同じ
	グループの ID または名前により取得される軽量レート テーブル リストの XML:
	`<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>`
	`<rateTables recordSize="2" startRow="1" totalCount="2">`
	`<rateTable id="12" name="Small scale Rate Table"`
	`logicName="BiSmallscaleRateTable"  description="Rate table for small scale businesses">`
	`<serviceItemType id="6" name="Laptop"/>`
	`</rateTable>`
	`<rateTable id="22" name="Large scale Rate Table"`
	`logicName="BiLargescaleRateTable"  description="Rate table for large scale businesses">`
	`<serviceItemType id="4" name="Desktop"/>`
	`</rateTable>`
	`</rateTables>`
	
	単一レート スキーマ取得の XML
	レート グループの更新要求のペイロードと同じ

	レート テーブル データ取得の XML: レート グループの更新要求のペイロードと同じ

オーダー管理

表 3-16　オーダー管理 API テーブル

領域	例

GET API	要求の情報の取得
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/definition/v1/ordermgmt/reqinfo/reqid/{reqId}
	サンプル出力:
	[
	{
	"Status":"Cancelled",
	"OrgUnit":"<s ID=\"847\"/>",
	"CustomerPhone":null,
	"Customer":"Customer1",
	"Initiator":"Initiator1",
	"CustomerEmail":null,
	"CreatedDate":"5/26/16 7:46 AM",
	"ClosedDate":"05/26/2016",
	"ReqNo":"1",
	"SubmittedDate":"05/26/2016"
	}
	]

	要求のためのサービスの情報を取得
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/definition/v1/ordermgmt/servicenfo/reqid/{reqId}
	サンプル出力:
	[
	{
	"Status":"<s ID=\"68\" />",
	"StdDuration":null,
	"UnitCost":"0.0",
	"Qty":"1",
	"Subtotal":"0.0",
	"Name":"TestsService",
	"ServiceLevelDesc":null
	},
	{
	"TotalCost":"0.0"
	}
	]

表 3-16　オーダー管理 API テーブル(続き)

領域	例

	要求に関するすべてのシステム コメントおよびユーザ コメントを取得
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/definition/v1/ordermgmt/comments/reqid/{reqId}
	[
	[
	{
	"CommentText":"Comment",
	"Name":"admin admin",
	"On":"Mon Jun 27 03:29:00 PDT 2016"
	}
	],
	[
	{
	"CommentText":"admin admin cancelled the requisition 20.",
	"Name":"admin admin",
	"On":"Tue Jun 14 09:04:49 PDT 2016"
	}
	]
	]

	要求のすべての添付ファイルの詳細を取得
	REST URL:
	URL:
	http://<ServerURL>/RequestCenter/nsapi/definition/v1/ordermgmt/attachmentslist/reqid/{reqId}
	サンプル出力:
	[
	{
	"UploadDate":"05/26/2016",
	"DocSize":"881304",
	"DocumentId":"1",
	"DocumentName":"file.txt"
	}
	]

表 3-16　オーダー管理 API テーブル（続き）

領域	例
	要求のドキュメント ID を使用して添付ファイルをダウンロード REST URL: URL: http://<ServerURL>/RequestCenter/nsapi/definition/v1/ordermgmt/attachment/viewdocid/{document Id} サンプル出力: `downloads the file`

	ログイン中のユーザがオーダーできるすべてのサービス ID を取得
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/definition/v1/orderableserviceids
	サンプル出力:
	{
	"Clone Container": 1015,
	"Cassandra Cluster App (v2.1.1)": 1086,
	"Add Cloud Center User": 1081,
	"API_Service": 1091
	}

	ログイン中のユーザがログイン ID がわかっている顧客に代わってオーダーできるすべてのサービスを取得
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/definition/v1/orderableserviceids/customerlogin/{customerlogin}
	出力例::
	{
	"1Service:: 2
	}

ポリシー

領域	例
コア API	サービス項目タイプによるポリシーの取得 サービス項目タイプの名前によりポリシーが返されます。 REST URL: /RequestCenter/nsapi/serviceitem/definition/{ serviceItemTypeName }/policies XML 応答: <policies totalCount="2" startRow="1" recordSize="2"> <policy policyType="quota" accountName="bangalore" policyEnabled="true" executionOrder="2" dataTypeLogicName="SiLaptop" policyTemplateLogicName="quotacheck" policyId="34" name="Laptop Quota policy"> <description> </description> <policyParamValues> <paramValue policyParamLogicName="quotacheck-serviceitemattribute" paramValue="Quota"/> <paramValue policyParamLogicName="quotacheck-quotathreshold" paramValue="90"/> </policyParamValues> <policyActions> <policyAction policyActionTemplateName="orderservice" executionOrder="1"> <actionParamValues> <actionParamValue policyActionTemplateParamLogicName="orderservice-servicename" actionParamValue="Apple iPhone"/> <actionParamValue policyActionTemplateParamLogicName="orderservice-initiatorname" actionParamValue="admin"/> <actionParamValue policyActionTemplateParamLogicName="orderservice-customername" actionParamValue="admin"/> <actionParamValue policyActionTemplateParamLogicName="orderservice-billtoOU" actionParamValue="Site Administration"/> <actionParamValue

領域	例
	policyActionTemplateParamLogicName="orderservice-sections"> <dictionary name="TabletComputerDict"> <fields/> </dictionary> </actionParamValue> </actionParamValues> </policyAction> </policyActions> </policy> <policy policyType="update" accountName="bangalore" policyEnabled="true" executionOrder="1" dataTypeLogicName="SiLaptop" policyTemplateLogicName="updatecheckchange" policyId="33" name="Laptop Policy"> <description> </description> <policyParamValues> <paramValue policyParamLogicName="updatecheckchange-serviceitemattribute" paramValue="Price"/> </policyParamValues> <policyActions> <policyAction policyActionTemplateName="policyalert" executionOrder="1"> <actionParamValues> <actionParamValue policyActionTemplateParamLogicName="errormessage" actionParamValue="Policy alert when Price is updated for a Laptop"/> </actionParamValues> </policyAction> </policyActions> </policy> </policies>
	ServiceitemtypeName および PolicyID によるポリシーの取得 ServiceitemtypeName およびポリシー ID によりポリシーが返されます。 REST URL: /RequestCenter/nsapi/serviceitem/definition/**{serviceItemTypeName }**/policies/id/{policyId}

領域	例
	ServiceItemTypeName、PolicyName、PolciyType によるポリシーの取得 **serviceItemTypeName**、ポリシー名、ポリシー タイプによりポリシーが返されます。 **REST URL:** /RequestCenter/nsapi/serviceitem/definition/**{ serviceItemTypeName}**/policies/name/{name}?policyType={ capacity, quota, time} **Example:** http://localhost:8088/RequestCenter/nsapi/serviceitem/definition/Laptop/policies/name/Laptop Policy?policyType=update 注:「policyType」クエリ パラメータに許可される入力値は次のとおりです。 capacity/quota/time/update
XML 応答:	<policy policyType="update" accountName="bangalore" policyEnabled="true" executionOrder="1" dataTypeLogicName="SiLaptop" policyTemplateLogicName="updatecheckchange" policyId="33" name="Laptop Policy"> <description> </description> <policyParamValues> <paramValue policyParamLogicName="updatecheckchange-serviceitemattribute" paramValue="Price"/> </policyParamValues> <policyActions> <policyAction policyActionTemplateName="policyalert" executionOrder="1"> <actionParamValues> <actionParamValue policyActionTemplateParamLogicName="errormessage" actionParamValue="Policy alert when Price is updated for a Laptop"/> </actionParamValues> </policyAction> </policyActions> </policy>
エラー応答	<nsapi-error-response> No policy found for this name (Laptop ABCD Policy) and for policyType (update). </nsapi-error-response>

領域	例
POST	特定の ServiceItemTypename のポリシーの作成 POST URL: /RequestCenter/nsapi/serviceitem/definition/**{serviceItemTypeName}**/policies/create 入力 XML: <policy policyType="update" accountName="bangalore" policyEnabled="true" executionOrder="1" dataTypeLogicName="SiLaptop" policyTemplateLogicName="updatecheckchange" policyId="33" name="Laptop Policy"> <description> </description> <policyParamValues> <paramValue policyParamLogicName="updatecheckchange-serviceitemattribute" paramValue="Price"/> </policyParamValues> <policyActions> <policyAction policyActionTemplateName="policyalert" executionOrder="1"> <actionParamValues> <actionParamValue policyActionTemplateParamLogicName="errormessage" actionParamValue="Policy alert when Price is updated for a Laptop"/> </actionParamValues> </policyAction> </policyActions> </policy> 応答 XML: <nsapi-response> <status-message code="POLICY_SUCCESS_003"> <value>Policy created successfully.</value> </status-message> </nsapi-response>

領域	例
PUT	**特定の ServiceItemTypename のポリシーの更新** **PUT URL:** /RequestCenter/nsapi/serviceitem/definition/**{serviceItemTypeName}**/policies/update 入力 XML: <policy policyType="update" accountName="bangalore" policyEnabled="true" executionOrder="1" dataTypeLogicName="SiLaptop" policyTemplateLogicName="updatecheckchange" policyId="33" name="Laptop Policy"> <description> </description> <policyParamValues> <paramValue policyParamLogicName="updatecheckchange-serviceitemattribute" paramValue="Price"/> </policyParamValues> <policyActions> <policyAction policyActionTemplateName="policyalert" executionOrder="1"> <actionParamValues> <actionParamValue policyActionTemplateParamLogicName="errormessage" actionParamValue="Policy alert when Price is updated for a Laptop"/> </actionParamValues> </policyAction> </policyActions> </policy> 応答 XML: <nsapi-response> <status-message code="POLICY_SUCCESS_003"> <value>Policy updated successfully.</value> </status-message> </nsapi-response>

領域	例
DELETE	**PolicyID または PolicyName による特定の ServiceItemTypename のポリシーの削除** **DELETE URL:** /RequestCenter/nsapi/serviceitem/definition/**{serviceItemTypeName}**/policies/delete **入力 XML:** ポリシー名により削除するには、次のように入力します。 <policy policyType="quota" name="Laptop Quota policy Fourth"> </policy> ポリシー ID により削除するには、次のように入力します。 <policy policyId="38"> </policy> **応答:** <nsapi-response> <status-message code="POLICY_SUCCESS_002"> <value>Policy deleted successfully.</value> </status-message> </nsapi-response>

ディレクトリ データ

個人（Person）

表 3-17　個人 API テーブル

領域	例
コア API	**すべての人の取得** すべての人が返されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/people Java の例 PersonList person = NSApiClient.getDirectory().getPeople(null);
	Id による個人の取得 指定した Person Id を持つ個人が返されます。 ネストされたエンティティ（OU、グループ、ロール、住所、連絡先、および設定）が取得されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/people/id/<personId> Java の例 `Person persons = NSApiClient.getDirectory().getPersonById(<personId>);`

表 3-17　個人 API テーブル(続き)

領域	例
	LoginName による個人の取得 指定した LoginName を持つ個人が返されます。 ネストされたエンティティ(OU、グループ、ロール、住所、連絡先、および設定)が取得されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/people/loginname/<loginName> Java の例 `Person persons = NSApiClient.getDirectory().getPersonByLoginName("<loginName>");`
	ログイン ユーザの取得 現在ログインしている個人が返されます。 REST URL: http://<serverURL>/RequestCenter/nsapi/directory/people/currentuser Java の例 `Person person  = NSApiClient.getDirectory().getCurrentUser();`

	ログイン名がわかっている場合に、個人にロールを割り当てる、または割り当てを解除する
	REST URL: /nsapi/directory/people/{loginname}/roles
	Method: POST
	サンプル ペイロード/本文:
	{
	"rolemapping":
	{
	"operation":"assign",
	"roles":["My Role","Tenant User"],
	}
	}
	operation に指定可能な値:「assign」または「unassign」
	成功応答:200
	エラー応答:400 bad request

表 3-17　個人 API テーブル(続き)

領域	例

特殊な条件	**個人の作成/更新**
	個人を取得(GET)する前述の REST URL のいずれかの応答から、個人 XML を取得します。
	個人を更新する REST URL に対して個人 XML を POST することによって、個人の作成または変更を行います。
	(XML のログインまたは personId で識別される)個人が存在する場合は更新操作が実行され、そうでない場合は新規個人が作成されます。
	作成操作には、次の 5 つの要素が必要です。
	• firstName
	• lastName
	• homeOrganizationalUnitName
	• 電子メール(e-mail)
	• ログイン
	作成操作では、個人のパスワードにログイン名が設定されます。
	更新操作でホーム OU に対する変更を行うと、個人のホーム OU が置換されます。関連付けられている他の OU、グループ、およびロールは、作成操作でも更新操作でも無視されます。
	ホーム OU、タイムゾーン、ロケール、スーパーバイザ、代理承認者、ログイン モジュール、連絡先、および住所の各属性には、次のルールが適用されます。
	• id 要素が XML で送信されたがデータベースには見つからなかった場合、例外がスローされ、該当するメッセージが示されます。
	• id 要素が見つからなかった場合は、XML で送信された Name 要素が代わりに使用されます。その名前がデータベースに見つからなかった場合は、例外がスローされます。
	• id または name 要素のいずれも XML 内に存在せず、属性がオプションである場合、その属性は作成/更新操作で無視されます(HomeOU は必須です)。
	• XML で送信されたデータに誤りや欠落があったために作成または更新操作が失敗した場合、HTTP ステータス コード「422 Unprocessable Entity」が返されます。
	POST REST URL:
	http://<ServerURL>/RequestCenter/nsapi/directory/people/update
	Java の例
	`Person person = NSApiClient.getDirectory().getPersonById(<personId>);`
	`person.setLastName("<lastName>");`
	`Person persons = NSApiClient.getDirectory().updatePerson(person);`

表 3-17　個人 API テーブル(続き)

領域	例

フィルタ	**OU 名フィルタ**
	検索では大文字と小文字が区別され、完全一致が使用されます。
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/directory/people?ouname=<ouName>
	Java の例
	`MultiValueMap paramsMap = new MultiValueMap();`
	`paramsMap.put("ouname", "<ouName>");`
	`PersonList person = NSApiClient.getDirectory().getPeople(paramsMap);`

	グループ名フィルタ
	検索では大文字と小文字が区別され、完全一致が使用されます。
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/directory/people?groupname=<groupName>
	Java の例
	`MultiValueMap paramsMap = new MultiValueMap();s`
	`paramsMap.put("groupname", "<groupName>");`
	`PersonList person = NSApiClient.getDirectory().getPeople(paramsMap);`

	ロール名フィルタ
	検索では大文字と小文字が区別され、完全一致が使用されます。
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/directory/people?rolename=<roleName>
	Java の例
	`MultiValueMap paramsMap = new MultiValueMap();`
	`paramsMap.put("rolename", "<roleName>");`
	`PersonList person = NSApiClient.getDirectory().getPeople(paramsMap);`

ソート カラム	[姓(First Name)]、[名(Last Name)]、[ログイン名(Login Name)]

XML 応答	`<people totalCount="x" recordSize="y" startRow="z">`
	`  <person>`
	`  . . .`
	`  <contacts>`
	`  . . .`
	`  <contact>`
	`  <contactId></contactId>`
	`  <contactType></contactType>`
	`  <contactTypeId></contactTypeId>`
	`  <value></value>`
	`  </contact>`
	`  . . .`
	`  </contacts>`
	`  . . .`
	`  </person>`
	</people>

組織

表 3-18　組織単位 API テーブル

領域	例
コア API	**すべての組織単位の取得** すべての組織単位が返されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/organizationalunits Java の例 `OrganizationalUnitList Ou = NSApiClient.getDirectory().getOrganizationalUnits(null);`
	Id による組織単位の取得 ネストされたエンティティ（下位の組織単位）は、getById および getByName の場合にのみ取得されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/organizationalunits/id/<ouId> Java の例 `OrganizationalUnit Ou = NSApiClient.getDirectory().getOrganizationalUnitById(ouId);`
	Name による組織単位の取得 ネストされたエンティティ（下位の組織単位）は、getById および getByName の場合にのみ取得されます。 同じ名前でタイプが異なる OU が 2 つ見つかった場合は、サービス チーム OU が返されます。 オプションのパラメータ ?type=<BusinessUnit/serviceTeam> も指定できます（「all」はタイプ パラメータ値では許可されません）。 REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/organizationalunits/name/<ouName> Java の例 `OrganizationalUnit Ou = NSApiClient.getDirectory().getOrganizationalUnitByName("<ouName>");`
	Type による OU の取得 指定した OU タイプの組織単位がすべて返されます。 指定可能な値:「all」、「*BusinessUnit*」、「*ServiceTeam*」。 REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/organizationalunits?type=<ouType> Java の例 `OrganizationalUnitList Ou = NSApiClient.getDirectory().getOrganizationalUnitByType("<ouType>");`

表 3-18　組織単位 API テーブル（続き）

領域	例

	親組織には割り当てずに、タイプが事業単位（BusinessUnit）の組織単位を作成する
	REST URL: nsapi/directory/organizationalunits
	Method: PUT
	サンプル ペイロード/本文:
	{
	"organizationalunit": {
	"description": "Details about Tail-f",
	"isBillable": "true",
	"organizationalUnitName": "Tail-f",
	"organizationalUnitType": "BusinessUnit",
	"status": "Active"
	}
	}
	成功応答：201 created
	エラー応答：400 bad request

	親組織に割り当てずに、タイプがサービス チーム（ServiceTeam）の組織単位を作成する
	REST URL: nsapi/directory/organizationalunits
	Method: PUT
	サンプル ペイロード:
	{
	"organizationalunit": {
	"description": "Automobile company,
	"isBillable": "true",
	"organizationalUnitName": "Tail-f",
	"organizationalUnitType": "ServiceTeam",
	"status": "Active"
	}
	}
	成功応答：201 created
	エラー応答：400 bad request

表 3-18　組織単位 API テーブル(続き)

領域	例

	ステータスが非アクティブでタイプが事業単位(BusinessUnit)の組織単位を作成する
	REST URL: nsapi/directory/organizationalunits
	Method: PUT
	サンプル ペイロード:
	{
	"organizationalunit": {
	"description": "Automobile company",
	"isBillable": "true",
	"organizationalUnitName": "Ferrari",
	"organizationalUnitType": "BusinessUnit",
	"status": "Inactive"
	}
	}
	成功応答:201 created
	エラー応答:400 bad request

	ステータスがアクティブで、タイプが事業単位(BusinessUnit)の組織単位を作成し、親組織単位に関連付ける
	REST URL: nsapi/directory/organizationalunits
	Method: PUT
	サンプル ペイロード:
	{
	"organizationalunit": {
	"description": "Automobile company",
	"isBillable": "true",
	"organizationalUnitName": "Hero Honda",
	"organizationalUnitType": "BusinessUnit",
	"status": "Active",
	"parentId": "8",
	"parentName": "Honda"
	}
	}
	成功応答:201 created
	エラー応答:400 bad request

表 3-18　組織単位 API テーブル（続き）

領域	例

	組織単位を更新するが、親組織単位には関連付けない
	更新操作により、説明、OUName、OUType、およびステータスを更新できます。
	REST URL: nsapi/directory/organizationalunits
	Method: POST
	サンプル ペイロード:
	{
	"organizationalunit": {
	"description": "Automobile company",
	"organizationalUnitId": "19",
	"isBillable": "true",
	"organizationalUnitName": "f-tail",
	"organizationalUnitType": "ServiceTeam",
	"organizationalUnitTypeId": "1",
	"status": "Active"
	}
	}
	成功応答：200 created
	エラー応答：400 bad request

	親組織単位に関連付けられている組織単位を更新する
	更新操作により、説明、OUName、OUType、ステータス、および親組織を更新できます。
	REST URL: nsapi/directory/organizationalunits
	Method: POST
	サンプル ペイロード:
	{
	"organizationalunit": {
	"description": "Automobile company",
	"organizationalUnitId": "5",
	"isBillable": "false",
	"organizationalUnitName": "Aveeno",
	"organizationalUnitType": "BusinessUnit",
	"organizationalUnitTypeId": "2",
	"status": "Active",
	"parentId": "16",
	"parentName": "Mazda23"
	}
	}
	成功応答：200 created
	エラー応答：400 bad request

表 3-18　組織単位 API テーブル(続き)

領域	例

	OU 名がわかっている場合に、組織単位にロールを割り当てる、または割り当てを解除する
	REST URL: nsapi/directory/organizationalunits/name/{OUname}/roles
	Method: POST
	サンプル ペイロード:
	{
	"rolemapping":
	{
	"operation":"assign",
	"roles":["My Role",”Tenant User”],
	"type":"ServiceTeam"
	}
	}
	タイプに指定可能な値:「BusinessUnit」または「ServiceTeam」
	operation に指定可能な値:「assign」または「unassign」
	成功応答:200 created
	エラー応答:400 bad request

	OU ID がわかっている場合に、組織単位にロールを割り当てる、または割り当てを解除する
	REST URL: nsapi/directory/organizationalunits/name/{OUID}/roles
	Method: POST
	サンプル ペイロード:
	{
	"rolemapping":
	{
	"operation":"assign",
	"roles":["My Role",”Tenant User”],
	"type":"ServiceTeam"
	}
	}
	type に指定可能な値:「BusinessUnit」または「ServiceTeam」
	operation に指定可能な値:「assign」または「unassign」
	成功応答:200 created
	エラー応答:400 bad request

表 3-18　組織単位 API テーブル（続き）

領域	例
フィルタ	**組織単位名フィルタ** 検索では大文字と小文字が区別されます。 StartsWith（newscale.properties の ContainsQueryInFnS=false）：先頭のワイルドカードは無視されます。 Contains（newscale.properties の ContainsQueryInFnS=true）：先頭のワイルドカードは、必要に応じて指定してください。 ワイルドカードの名前検索では、オプションのパラメータ ?type=<businessUnit/serviceTeam> を指定することもできます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/organizationalunits?name=<wildcard Value> Java の例 `OrganizationalUnitList Ou = NSApiClient.getDirectory().getOrganizationalUnitsByFilter("<wildcardValue>");`
ソート カラム	Organizational Unit Name
応答 XML	`<organizationalunits totalCount="x" recordSize="y" startRow="z"> <organizationalunit> . . . </organizationalunit> </organizationalunits>`

グループ（Groups）

表 3-19　グループ API テーブル

領域	例
コア API	**すべてのグループの取得** すべてのグループが返されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/groups Java の例 `GroupList groups = NSApiClient.getDirectory().getGroups(null);`
	Id によるグループの取得 ネストされたエンティティ（サブグループ）は、getById および getByName の場合にのみ取得されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/groups/id/<groupId> Java の例 `Group groups = NSApiClient.getDirectory().getGroupsById(<groupId>);`

表 3-19　グループ API テーブル(続き)

領域	例
	Name によるグループの取得 ネストされたエンティティ(サブグループ)は、getById および getByName の場合にのみ取得されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/groups/name/<groupName> Java の例 `Group groups = NSApiClient.getDirectory().getGroupsByName("<groupName>");`

	グループの作成
	REST URL: nsapi/directory/groups
	Method: PUT
	サンプル ペイロード/本文:
	{
	“group”:
	{
	“groupName”: “Test”, “description”, “Test Group”, “status”: “Active”
	}
	}
	成功応答:201 created
	エラー応答:400 bad request

	グループを作成し、親 ID に関連付ける
	REST URL: nsapi/directory/groups
	Method: PUT
	サンプル ペイロード:
	{
	“group”:
	{
	“groupId”: “1”, “groupName”: “TestP”, “parentId”: “1”, “parentName”: “ABCGroup”
	}
	}
	成功応答:201 created
	エラー応答:400 bad request

表 3-19　グループ API テーブル（続き）

領域	例

	グループ ID を使用してグループを更新する - 名前、説明、ステータスを更新する
	REST URL: nsapi/directory/groups
	Method: POST
	サンプル ペイロード:
	{
	“group”:
	{
	“groupId”: “14”, “groupName”: “DRMGroup”, “status”: “Active”
	}
	}
	成功応答：200 OK
	エラー応答：400 bad request

	グループ ID を使用してグループを更新し、新しい親グループに関連付ける - 名前、説明、ステータスを更新し、親グループと関連付ける
	REST URL: nsapi/directory/groups
	Method: POST
	サンプル ペイロード:
	{
	“group”:
	{
	“groupId”: “14”, “groupName”: “NimbusGroup”, “status”: “Active”, “parentId”: “22, “parentName”: “XYZGroup”
	}
	}
	成功応答：200 OK
	エラー応答：400 bad request

表 3-19　グループ API テーブル(続き)

領域	例

	グループにロールを割り当てる、または割り当てを解除する
	REST URL: nsapi/directory/groups/name/{groupname}/roles
	Method: POST
	サンプル ペイロード:
	{
	"rolemapping":
	{
	"operation":"assign",
	"roles":["My Role"],
	}
	}
	operation に指定可能な値:「assign」または「unassign」
	成功応答:200 created
	エラー応答:400 bad request

フィルタ	***グループ名フィルタ*** 検索では大文字と小文字が区別されます。 StartsWith(newscale.properties の ContainsQueryInFnS=false):先頭のワイルドカードは無視されます。 Contains(newscale.properties の ContainsQueryInFnS=true):先頭のワイルドカードは、必要に応じて指定してください。 REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/groups?name=<wild cardValue> Java の例 `GroupList groups = NSApiClient.getDirectory().getGroupsByFilter("<wildcardValue>");`
ソート カラム	グループ名

応答 XML	`<groups totalCount="x" startRow="y" recordSize="z">`
	`  <group>`
	`  . . .`
	`  </group>`
	`<groups>`

アカウント

表 3-20　アカウント API テーブル

領域	例
Core API	すべてのアカウントの取得 すべてのアカウントが返されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/accounts
	ID によるアカウントの取得 ID によりアカウントが返されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/accounts/id/{id}
	名前によるアカウントの取得 名前によりアカウントが返されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/accounts/name/{name}
	ワイルド カードを使用した名前によるアカウントの取得 検索対象のアカウントが返されます。 REST URL (HTTP POST): http://<ServerURL>/RequestCenter/nsapi/directory/accounts?name={wild card name}

表 3-20　アカウント API テーブル(続き)

領域	例
他の使用シナリオ	アカウントを作成または更新します(1 アカウント オブジェクトのみ)。 REST URL(HTTP POST): http://<ServerURL>/RequestCenter/nsapi/directory/accounts/update <account> <name>Sample Account Name</name> <accountId>123</accountId> <accountType>Project Account</accountType> <description>Sample Account Description</description> <billingRateGroup>Sample Rate Group Name</billingRateGroup> <organizationalUnits> <organizationalUnit> <organizationalUnitName>OU Name 1</organizationalUnitName> <organizationalUnitType>Service Team</organizationalUnitType> </organizationalUnit> <organizationalUnit> <organizationalUnitName>OU Name 2</organizationalUnitName> <organizationalUnitType>Service Team</organizationalUnitType> </organizationalUnit> </organizationalUnits> <functionalPositions> <functionalPosition> <functionalPositionName>Func Pos 1</functionalPositionName> <ownerLoginName>assigneeLoginName</ownerLoginName> </functionalPosition> <functionalPosition> <functionalPositionName>Func Pos 2</functionalPositionName> <ownerLoginName /> </functionalPosition> </functionalPositions> <customAttribute name="IntegerValue" /> <customAttribute name="LongString">This is Long String</customAttribute> <customAttribute name="MediumString" /> <customAttribute name="ShortString">This is short value</customAttribute> <customAttribute name="MoneyValue">1234567.89</customAttribute> <customAttribute name="DoubleValue">12345678</customAttribute> <customAttribute name="DateValue">31-Jan-2013</customAttribute> </account>

表 3-20　アカウント API テーブル(続き)

領域	例
	アカウントを削除します(1 アカウント オブジェクトのみ) REST URL (HTTP DELETE): http://<ServerURL>/RequestCenter/nsapi/directory/accounts/delete <account> <name>Sample Account Name</name> <accountId>123</accountId> </account>
	OU 名と OU のタイプの取得 テナント アカウントに関連付けられていない OU 名、OU のタイプが返されます(ログイン ユーザが OU へのアクセス権限を持つテナント アカウント。RBAC チェックが実施されます)。 REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/accounts/tenantorgunitlist
	アカウントの役職の取得 アカウントの役職が返されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/accounts/functionalpositionlist
フィルタ	フィルタはサポートされません。
ソート カラム	ソートは許可されません。
応答 XML	<accounts totalCount="x" startRow="y" recordSize="z"> <account> . . .(same as the request payload for update) </account> </accounts>

トランザクション データ

要求

表 3-21　要求 API テーブル

領域	例
コア API	**現行ユーザに対する要求の取得** 次に示すデフォルトのフィルタを使用して要求を取得します。 ViewName = Ordered for Self Status = Ongoing REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/requisitions Java の例 `RequisitionList requisitions = NSApiClient.getTransaction().getRequisitions(null);`
	Id による要求の取得 ログインしているユーザに対して RBAC チェックが適用されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/requisitions/id/<requisitionId> Java の例 `Requisition requisitions = NSApiClient.getTransaction().getRequisitionsById(<requisitionId>);`

表 3-21　要求 API テーブル(続き)

領域	例
	サービス フォーム データの取得 指定された要求 ID のすべての要求エントリのサービス フォーム データを取得します。応答は、各要求のサービス名、フォーム フィールドのデータ、および requisitionEntryId を含むリストです。応答は、XML 形式および JSON 形式で取得できます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/requisitions/id/<requisitioin_id>/requisitiondata Java の例 `RequisitionEntriesList requisitionEntries = NSApiClient.getTransaction().getRequisitionData(id)`
	サービスを追加せずに、未送信の既存の要求/カートを送信する Method: POST REST URL: /RequestCenter/nsapi/transaction/requisitions 要求ペイロード/本文は不要です。 成功コード:201 応答エラー コード:表 REST/Web サービスのエラー メッセージを参照

表 3-21　要求 API テーブル(続き)

領域	例

	サービスを追加して、新しい要求/カートを送信します。
	Method: POST
	REST URL:
	/RequestCenter/nsapi/transaction/requisitions
	ペイロード:
	{
	"**requisition**": {
	"**customerLoginName**": "admin",
	"**billToOU**": "H_OU",
	"**services**":[{
	"name": "TestServiceRest"
	"quantity": "1",
	"version": "0",
	"dictionaries": [{
	"**name**": "TestNonGrid",
	"**data**": {
	"FullName": "AAB",
	"HireDate": "78/02/20",
	"MultiSelect":["MS3","MS4"],
	}
	},{
	"**name**": "TestG",
	"**data**":
	[
	{"city":"Bangalore", "country": "India"},
	{"city":"Mysore", "country":"India"}
	]
	}]
	}]
	}
	}
	成功コード:201
	応答エラー コード:表 REST/Web サービスのエラー メッセージを参照

表 3-21　要求 API テーブル(続き)

領域	例

	既存の未送信の要求/カートにオーダーを追加する
	Method: PUT
	REST URL:
	/RequestCenter/nsapi/transaction/requisitions
	ペイロード:
	{
	"**requisition**": {
	"**customerLoginName**": "admin",
	"**billToOU**": "H_OU",
	"**services**":[{
	"name": "TestServiceRest"
	"quantity": "1",
	"version": "0",
	"dictionaries": [{
	"**name**": "TestNonGrid",
	"**data**": {
	"FullName": "AAB",
	"HireDate": "78/02/20",
	"MultiSelect":["MS3","MS4"],
	}
	},{
	"**name**": "TestG",
	"**data**":
	[
	{"city":"Bangalore", "country": "India"},
	{"city":"Mysore", "country":"India"}
	]
	}]
	}]
	}
	}
	成功コード:200
	応答エラー コード:表 REST/Web サービスのエラー メッセージを参照

表 3-21　要求 API テーブル(続き)

領域	例
	未送信の要求/カートをキャンセルする Method: DELETE REST URL: /RequestCenter/nsapi/transaction/requisitions ペイロードは不要です。 成功コード:200 応答エラー コード:表 REST/Web サービスのエラー メッセージを参照
	要求/カートから特定のサービスを削除/キャンセルする Method: DELETE REST URL: /RequestCenter/nsapi/transaction/requisitionentries/{reqentry_id} ペイロードは不要です。 成功コード:200 応答エラー コード:表 REST/Web サービスのエラー メッセージを参照
	オープン オーダーのモニタ プランのキャンセル 要求キャンセル nsAPI の cancelMonitorPlan オプションは、オープン オーダーのモニタ プランをキャンセルします。要求のステータスは Delivery Canceled に設定されます。 Method: Delete REST URL: /RequestCenter/nsapi/transaction/requisition/{requisitionID}?cancelMonitorPlan=true ペイロードは不要です。 成功コード:200 応答エラー コード:表 REST/Web サービスのエラー メッセージを参照 (注)　cancelMonitorPlan オプションを実行する権限があるのは、サイト管理者ロールを持つユーザのみです。

表 3-21　要求 API テーブル(続き)

領域	例
	サービス要求フォームを更新する nsXML で指定された入力に基づいてフォーム データを更新する POST REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/updateServiceRequest サンプル ペイロード: <?xml version="1.0" encoding="UTF-8"?> <message channel-id="" is-autocomplete="false" is-published="true"> <task-started task-type="task"> <requisition> <requisition-entry> <data-values> <data-value multi-valued="false"> <name>GridUpdate-1.Name</name> <value>ordernamechanged4</value> </data-value> <data-value multi-valued="false"> <name>GridUpdate-1.Roll</name> <value>orderrollchanged4</value> </data-value> </data-values> <requisition-entry-id>55</requisition-entry-id> </requisition-entry> </requisition> </task-started> </message> サンプル応答: 更新サービス要求が正常に開始されます。

表 3-21　要求 API テーブル(続き)

領域	例

フィルタ	**ビューおよびステータス フィルタ**
	フィルタに使用可能なビューおよびステータスは、My Services の [Requisitions] タブにあるものに対応しています。
	Status が指定されなかった場合は、「Ongoing」が使用されます。
	ViewName が指定されなかった場合は、「Ordered for Self」が使用されます。
	ViewName に指定可能な値:
	• Ordered for Self:現行ユーザに対する要求。
	• Ordered for Others:現行ユーザが(「Order on Behalf」を使用して)他人の代わりに送信した要求。
	• Ordered for my unit:現行ユーザが属している OU 内の人に関する要求(ユーザが「See Requisitions for My Business Units」機能を持っている場合に限り、このビューは他の人に関するデータを返します)。
	Status に指定可能な値:Ongoing、Preparation、Ordered、Closed、Canceled、Rejected、All。
	誤った値が指定された場合、デフォルトの値(「Ongoing」および「Ordered for Self」)が使用されます。
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/transaction/requisitions/ViewName=<viewName>[|AND|Status=<status>]
	Java の例
	`String filterString = "ViewName=<viewName>|AND|Status=<status>";`
	`RequisitionList requisitons = NSApiClient.getTransaction().getRequisitionsByFilter(filterString);`
	(注)　PUT および POST メソッドではフィルタはサポートされません。

ソート カラム	Customer Name、Owner(Initiator)Name、Requisition ID、Service Name、Started Date、Status、Submit Date **(注)**　PUT および POST メソッドではソートはサポートされません。

表 3-21　要求 API テーブル(続き)

領域	例
GET の応答 XML	<requisitions totalCount="x" recordSize="y" startRow="z"> <requisition> . . . </requisition> </requisitions>

PUT および POST の応答 XML	{ "RequisitionSubmit": {
	"requisitionId": 536,
	"customer": "admin admin",
	"initiator": "admin admin",
	"startedDate": "…",
	"startedDateRaw": …,
	"dueDate": "…"
	"dueDateRaw": …,
	"status": "Ordered",
	"links" : [
	{ "name" : "RequisitionStatus",
	"href" : "http://…"
	},
	{ "name" : "ServiceItems",
	"href" : "http://…"
	}
	]
	}
	}
	(注)　dueDate および dueDateRaw 属性は、非同期伝送設定が [管理(Administration)] > [設定(Settings)] モジュールで無効の場合だけ使用できます。

表 3-21　要求 API テーブル(続き)

領域	例

REST API	サービス フォーム詳細の取得
	Method: GET
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/transaction/v1/orderform?serviceids=<list of service ids, comma seperated>
	出力例: :
	{
	"API_Service": {
	"serviceID": "12",
	"quantity": 1,
	"API_AFC-API_Dictionary": {
	"Name": {
	"value": ""
	},
	"City": {
	"value": [],
	"isMultiValued": "true",
	"selectableValues": [
	"Bangalore",
	"Mysore"
	]
	},
	"Range": {
	"value": " ",
	"selectableValues": [
	"100-1000",
	"1000-2000"
	]
	},
	"Sal": {
	"value": [],
	"isMultiValued": "true",
	"selectableValues": [
	"10k",
	"20k",
	"30k",
	"40k"
	]
	}
	},
	"Form-Dictionary": {
	"Select_Person": {
	"value": ""
	},
	"Login_ID": {
	"value": ""
	}
	}
	},

表 3-21　要求 API テーブル(続き)

領域	例

	<pre>"0Service": {
	"serviceID": "1",
	"quantity": 1,
	"maxQuantity": 2,
	"Form-Dictionary": {
	"Select_Person": {
	"value": ""
	},
	"Login_ID": {
	"value": ""
	}
	}
	}
	}</pre> フォームにグリッドディクショナリが含まれている場合の出力例: <pre>{
	"Grid_Service": {
	"serviceID": "114",
	"quantity": 1,
	"grid_form-Grid_dict": {
	"isGrid": true,
	"row-1": {
	"Name": {
	"value": ""
	},
	"Description": {
	"value": ""
	}
	}
	}
	}
	}</pre> **(注)**　フィールドに複数の値がある場合、そのフィールドの値はリストとみなされ、そのフィールドには複数値フラグが表示されます。
	
	(注)　フィールドが必須の場合は、そのフィールドに必須フラグが表示されます。

表 3-21　要求 API テーブル(続き)

領域	例

	フォームにグリッド ディクショナリが含まれている場合の入力例:
	`{`
	`"Grid_Service": {`
	`"serviceID": "114",`
	`"quantity": 1,`
	`"grid_form-Grid_dict": {`
	`"row-1": {`
	`"Name": {`
	`"value": "Name1"`
	`},`
	`"Description": {`
	`"value": "desc1"`
	`}`
	`},`
	`"row-2": {`
	`"Name": {`
	`"value": "Name2"`
	`},`
	`"Description": {`
	`"value": "desc2”`
	`}`
	`}`
	`}`
	`}`
	`}`
	出力例::
	`{`
	`"Requisition ID":6,`
	`"InitiatorLogin":"admin admin",`
	`"customerLogin":"user1 user1",`
	`"startedDate":"Wed Oct 26 12:31:13 IST 2016",`
	`"status":"Ordered"`
	`}`
	(注)　要求送信 API のペイロードは、サービス フォーム詳細 API から取得できます。
	(注)　グリッド ディクショナリが個人ベースの場合、名前の代わりにログイン ID を送信する必要があります。
	(注)　サーバ側イベント(PostSubmit)があるフォーム ルールのみが実行されます。

要求エントリ

表 3-22　要求エントリ API テーブル

領域	例
コア API	ID による要求エントリの取得 現行ユーザに対して RBAC チェックが適用されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/requisitionentries/id/<reqEntryId> Java の例 `RequisitionEntry requisitonEntry = NSApiClient.getTransaction().getRequisitionEntryById(<reqEntryId>);`

	要求 ID による要求エントリの取得
	現行ユーザに対して RBAC チェックが適用されます。
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/transaction/requisitionentries/RequisitionNumber=<reqId>
	Java の例
	`MultiValueMap paramsMap = new MultiValueMap();`
	`paramsMap.put("startRow", "1"); //optional`
	`paramsMap.put("recordSize","10"); //optional`
	`String filterString = "RequisitionNumber=" + <reqId>;`
	`RequisitionEntryList requisitonEntries = NSApiClient.getTransaction().getRequisitionEntries(paramsMap, filterString);`

フィルタ	**ServiceName** Displays all ordered services that contains the filter values. Service Name filter value is test. Lists all ordered services where the service name contains "test". REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/requisitionentries/ViewName=<viewName>|AND|ServiceName=<wildcard>|AND|Status=<status> Java の例 **isNgcRequest** Displays service request process complete percentage if the isNgcRequest filter value is false or not mentioned. If the isNgcRequest filter value is true, the service process request percentage is not displayed on the Order form page. REST URL http://localhost:8088/RequestCenter/nsapi/transaction/requisitionentries/ViewName=Ordered%20for%20Myself|AND|ServiceName=test|AND|Status=AllOpenExceptPreparation?responseType=json&isNgcRequest=true&recCount=20&sortBy=submitDate&sortDir=desc
ソート カラム	Due On、Requisition Entry ID

応答 XML	`<requisitionEntries totalCount="x" recordSize="y" startRow="z">`
	`  <requisitionEntry>`
	`  . . .`
	`  </requisitionEntry>`
	`</requisitionEntries>`

表 3-23　DELETE 要求エントリ API テーブル

領域	例
コア API	要求またはカートのサービスを取り消す • 要求またはカートのステータスが未送信の場合に、サービスを削除します。 • 要求またはカートのステータスが送信済みの場合は、サービスを取り消します。 DELETE REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/requisitionentries/<reqEntryId> (注)　<reqEntryId> は要求 ID に置き換えます。http://<ServerURL>/RequestCenter/nsapi/transaction/requisitionentries/175 を考慮し、応答は次のようになります。
フィルタ	N/A
ソート カラム	N/A

応答 XML	{
	"requisition id": 175
	}

承認

表 3-24　承認 API テーブル

領域	例
コア API	**現行ユーザに対する承認の取得** 次に示すデフォルトのフィルタを使用して承認を取得します。 ViewName = Authorizations for Self Status = Ongoing REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/authorizations Java の例 `AuthorizationList authorizations = NSApiClient.getTransaction().getAuthorizations(null);`
	Id による承認の取得 現行ユーザに対して RBAC チェックが適用されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/authorizations/id/<taskId> Java の例 `Authorization authorizations = NSApiClient.getTransaction().getAuthorizationsById(<taskId>);`

表 3-24　承認 API テーブル(続き)

領域	例

	ID による関連承認タスク(承認チェーン)の取得
	現行ユーザに対して RBAC チェックが適用されます。
	渡された ID が、要求レベルの承認タスク(部門承認、部門確認、財務承認)の ID であった場合、その特定の要求に属するすべての要求レベルの承認タスクが返されます。
	渡された ID が、要求エントリ レベルの承認タスク(サービス グループ承認、サービス グループ確認)の ID であった場合、その特定の要求エントリに属するすべての要求エントリ レベルの承認タスクが返されます。
	デフォルトのソートは、Due On の降順で行われます。
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/transaction/authorizations?taskId=<taskId>
	Java の例
	`MultiValueMap paramsMap = new MultiValueMap();`
	`paramsMap.put("startRow", "" + 1);  //optional`
	`paramsMap.put("recordSize", "" +10);  //optional`
	`paramsMap.put("sortBy", "dueOn");  //optional`
	`paramsMap.put("sortDir", "asc"); //optional`
	`paramsMap.put("taskId", "<taskId>");`
	`Authorization authorizations =`
	`NSApiClient.getTransaction().getAuthorizations (paramsMap);`

フィルタ	**ビューおよびステータス フィルタ**
	フィルタに使用可能なビューおよびステータスは、My Services の [Authorizations] タブにあるものに対応しています。
	Status が指定されなかった場合は、「Ongoing」に設定されます。
	ViewName が指定されなかった場合は、「Authorizations for Self」に設定されます。
	ViewName に指定可能な値:
	• Authorizations for Self:現行ユーザに割り当てられている承認
	• Assigned and Unassigned Authorizations for Self:現行ユーザ、または現行ユーザがアクセスできるキューに割り当てられている承認
	• Authorizations for Others:現行ユーザが属している OU 内の人に割り当てられている承認(ユーザが「See Authorizations for My Business Units」機能を持っている場合に限り、このビューはデータを返します)
	Status に指定可能な値:Ongoing、Approved、Rejected、Canceled、Reviewed、All。
	誤った値が指定された場合、デフォルトの値(「Ongoing」および「Authorizations for Self」)が使用されます。
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/transaction/authorizations/ViewName=<viewName>[|AND|Status=<status>]
	Java の例
	`String filterString = "ViewName=<viewName>|AND|Status=<status>";`
	`AuthorizationList authorizations =`
	`NSApiClient.getTransaction().getAuthorizationsWithFilters(filterString);`

表 3-24　承認 API テーブル(続き)

領域	例
ソート カラム	Customer Name、Due On、Performer Name、Priority、Requisition ID
応答 XML	<authorizations totalCount="x" recordSize="y" startRow="z" /> <authorization> . . . </authorization> </authorizations>

タスク

表 3-25　タスク API テーブル

領域	例
コア API	**現行ユーザに対するタスクの取得** 次に示すデフォルトのフィルタを使用してタスクを取得します。 ViewName = AvailableWork REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/tasks Java の例 TaskList tasks = NSApiClient.getTransaction().getDeliveryTasks(null);
	ID によるタスクの取得 現行ユーザに対して RBAC チェックが適用されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/tasks/id/<taskId> Java の例 TaskFull tasks = NSApiClient.getTransaction().getDeliveryTaskById(<taskId>);

表 3-25　タスク API テーブル(続き)

領域	例

	要求エントリ ID によるタスクの取得
	現行ユーザに対して RBAC チェックが適用されます。
	デフォルトのソートは、Activity ID の昇順で行われます。
	パラメータおよび指定可能な値:
	• showSkippedTasks:false(デフォルト)、true
	• taskType:all(デフォルト)、delivery、authorization
	• showNestedTasks:false、true(デフォルト)
	このパラメータが true に設定されている場合、親と子のネスト階層が XML 構造内に保持された状態で配信タスクが返されます。
	• showChildDeliveryPlan:false(デフォルト)、true
	これは、バンドル サービスにのみ適用されます。このパラメータが true に設定されている場合は、すべての組み込みサービスの配信タスクも返されます。
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/transaction/tasks/RequisitionEntryNumber=<reqEntryId>?taskType=<taskType>&showSkippedTasks=<falseltrue>&showNestedTasks=<falseltrue>&showChildDeliveryPlan=<falseltrue>
	Java の例
	`MultiValueMap paramsMap = new MultiValueMap();`
	`paramsMap.put("taskType", "delivery"); //optional`
	`paramsMap.put("showSkippedTasks", "true"); //optional`
	`paramsMap.put("showNestedTasks", "true"); //optional`
	`paramsMap.put("sortBy", "dueOn"); //optional`
	`paramsMap.put("sortDir", "desc"); //optional`
	`String filterString = "RequisitionEntryNumber=" + <reqEntryId>;`
	`TaskList tasks = NSApiClient.getTransaction().getAuthAndDeliveryTasks (paramsMap, filterString);`

	要求 ID によるマイルストーンの取得
	現行ユーザに対して RBAC チェックが適用されます。
	配信プロセスのマイルストーン(確認、承認、配信プロジェクト)が、時系列順に返されます。ソートおよびページングはサポートされていません。
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/transaction/tasks/RequisitionNumber=<reqId>
	Java の例
	`String filterString = "RequisitionNumber=" + <reqId>;`
	`MilestoneList milestones = NSApiClient.getTransaction().getDeliveryProcessForMilestone (filterString);`

表 3-25　タスク API テーブル(続き)

領域	例
特殊な条件	**タスクの許可** REST URL にアクションおよびタスク ID を指定して HTTP POST を実行します。 POST REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/tasks /<taskId>/approve Java の例 `TaskAction approve = NSApiClient.getTransaction().approveTask(<taskId>);`
	タスクの拒否 REST URL にアクションおよびタスク ID を指定して HTTP POST を実行します。 POST REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/tasks/<taskId>/reject Java の例 `TaskAction Reject = NSApiClient.getTransaction().rejectTask(<taskId>);`
	タスクの完了 REST URL にアクションおよびタスク ID を指定して HTTP POST を実行します。 POST REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/tasks/<taskId>/done Java の例 `TaskAction Complete = NSApiClient.getTransaction().completeTask(<taskId>);`
	タスクの確認 REST URL にアクションおよびタスク ID を指定して HTTP POST を実行します。 POST REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/tasks/<taskId>/review Java の例 `TaskAction Review = NSApiClient.getTransaction().reviewTask(<taskId>);`

表 3-25　タスク API テーブル(続き)

領域	例

フィルタ	**ビュー フィルタ**
	フィルタに使用可能なビューは、Service Manager モジュールのビューに対応しています。ユーザ定義ビューは使用できません。
	ViewName フィルタが指定されなかった場合、「AvailableWork」に設定されます。
	ViewName に指定可能な値: AvailableWork、MyWork、MyLateWork、WorkForeCast
	誤った値が指定された場合は、デフォルト値が使用されます。
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/transaction/tasks?viewName=<viewName>
	Java の例
	`MultiValueMap paramsMap = new MultiValueMap();`
	`paramsMap.put("ViewName", "<viewName>");`
	`TaskList tasks = NSApiClient.getTransaction().getDeliveryTasks(paramsMap);`
	• ***要求エントリ レベル***のすべての承認タスク(将来の承認タスクを含む履歴)を表示できます。
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/transaction/tasks/RequisitionEntryNumber=111?taskType=authorization&Status=All
	• ***要求レベル***のすべての承認タスク(将来の承認タスクを含む履歴)を表示できます。
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/transaction/tasks/RequisitionNumber=111?taskType=authorization&Status=All

ソート カラム	Activity ID、Completed On、Customer Name、Customer OU Name、Due On、Effort、Initiator Name、Performer Name、Priority、Requisition ID、Scheduled Start Date、Service Name、Task Name、Task Type

応答 XML	`<tasks totalCount="x" recordSize="y" startRow="z">`
	`  <task>`
	`  . . .`
	`  </task>`
	`</tasks>`

ポリシーアラート

表 3-26　ポリシー アラート API テーブル

領域	例
Core API	CreatedDate によるポリシー アラートの取得 作成日によりポリシー アラートが返されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/policyalert/{columnName}{Operator}{Value}
	重大度のタイプと CreatedDate によるポリシー アラートの取得 重大度のタイプおよび作成日により、ポリシー アラートが返されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/policyalert/{columnName1}{Operator}{Value1}|{Join}|{columnName2}{Operator}{Value2}
	サービス項目タイプの名前と CreatedDate によるポリシー アラートの取得 サービス項目タイプの名前と作成日により、ポリシー アラートが返されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/policyalert/{columnName1}{Operator}{Value1}|{Join}|{columnName2}{Operator}{Value2}
	アカウント名と CreatedDate によるポリシー アラートの取得 アカウント名と作成日により、ポリシー アラートが返されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/policyalert/{columnName1}{Operator}{Value1}|{Join}|{columnName2}{Operator}{Value2}
	ポリシー アラートの削除 ポリシー アラートの削除はポリシー アラート エントリの UniqueID で実行されます。この固有 ID はポリシー アラート エントリごとにシステムにより生成される一意の ID です。 REST URL (HTTP DELETE): http://<ServerURL>/RequestCenter/nsapi/transaction/policyalert/delete/uniqueid/{uniqueIDs}

表 3-26　ポリシー アラート API テーブル(続き)

領域	例
フィルタ	最大 4 つのフィルタをサポートします。 比較演算子: 文字列カラム:equals、starts-with 演算子(contains および like 演算子は許可されません)。 関係演算子:AND(OR は許可されません) 区切り文字 = | 使用可能なカラム:ServiceItemTypeName、AccountName、SeverityType、および CreatedDate URL で許可されるパラメータ:startRow、recordSize、sortBy、sortDir、および responseType(xml または json)
ソート カラム	ソートは許可されません。
応答 XML	<policyAlert recordSize="2" startRow="1" totalCount="1"> <policyAlertRecord id="123" severityLevel="Info" alertContext="Policy Alert Action" requisitionID="12" requisitionEntryID="32"> <uniqueID>e96c062f-fe50-4b77-981a-dda1ab078808</uniqueID> <policy id="8" name="Quota policy 8" /> <serviceItemType id="82" name= "Laptop with Quota"/> <serviceItem id="12" name="Service Item one" /> <account id="1" name="Custom Account 1" /> <agreement id="6" name="Agreement Name"/> <message>Message Message Message Message</message> <createdDate local="04/09/2013 12:44 PM" raw="2013-04-09T19:44:28.720-07:00" /> </policyAlertRecord> </policyAlert>

課金履歴

表 3-27　課金履歴 API テーブル

領域	例
Core API	**取引日による課金履歴の取得** 取引日により、課金履歴が返されます。 REST URL: http://<ServerURL>:8088/RequestCenter/nsapi/transaction/billinghistory/{columnName}{Operator}{Value}

表 3-27　課金履歴 API テーブル(続き)

領域	例
	サービス項目名、サービス項目タイプの名前、および取引日による課金履歴の取得 サービス項目名、サービス項目タイプの名前、および取引日により、課金履歴が返されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/billinghistory/{columnName1}{Operator}{Value1}|{Join}|{columnName2}{Operator}{Value2}
	サービス項目タイプの名前と取引日による課金履歴の取得 サービス項目タイプと取引日により、課金履歴が返されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/billinghistory/{columnName1}{Operator}{Value1}|{Join}|{columnName2}{Operator}{Value2}
	アカウント ID と取引日による課金履歴の取得 アカウント ID と取引日により、課金履歴が返されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/billinghistory/{columnName1}{Operator}{Value1}|{Join}|{columnName2}{Operator}{Value2}
	アカウント名と取引日による課金履歴の取得 アカウント名と取引日により、課金履歴が返されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/billinghistory/{columnName1}{Operator}{Value1}|{Join}|{columnName2}{Operator}{Value2}

表 3-27　課金履歴 API テーブル（続き）

領域	例
フィルタ	最大 4 つのフィルタをサポートします。 REST URL: 比較演算子: 数値/日付カラム:=、>、<、>=、<= 文字列カラム: equals、starts-with 演算子（contains および like 演算子は許可されません） 関係演算子:AND（OR は許可されません） 区切り文字 = | 使用可能なカラム:ServiceItemName、ServiceItemTypeName、AccountID、AccountName、および TransactionDate フィルタの組み合わせ: 許可されるフィルタの組み合わせは次のとおりです。 • ServiceItemTypeName フィルタ、ServiceItemName フィルタ、および TransactionDate フィルタ • TransactionDate フィルタ（1 回以上） • TransactionDate フィルタおよび ServiceItemTypeName フィルタ • TransactionDate フィルタおよび AccountID フィルタ • TransactionDate フィルタおよび AccountName フィルタ http://<ServerURL>/RequestCenter/nsapi/serviceitem/<serviceItemName>/<columnName1><operator1><value1>[|<AND>|<columnName2><operator2><value2>][|<AND>|<columnName3><operator3><value3>] Java の例 `String filter = "<columnName1><operator1><value1>|<and>|<columnName2><operator2><value2>|<and>|<columnName3><operator3><value3>"; ServiceItemDTO serviceItems =`

表 3-27　課金履歴 API テーブル(続き)

領域	例
ソート カラム	ソートは許可されません。
応答 XML	<billingHistory recordSize="2" startRow="1" totalCount="1"> <billingRecord id="123"> <requisitionID>45</requisitionID> <requisitionEntryID>56</requisitionEntryID> <transactionDate local="01/03/2013 6:31 PM" raw="2013-01-04T02:31:43.400-08:00"/> <serviceItemType id="5" name="Laptop"/> <serviceItem id="4" name="DELL"/> <organizationalUnit id="5" name="My OU"/> <operation>Assemble Laptop</operation> <account id="5" name="Account Name"/> <agreement id="6" name="Agreement Name"/> <customer id="2" name="John John"/> <rateRecord rate="123.45" rateCode="RateCode1" unitOfMeasure="Seconds"/> <billingAttributes> <attribute name="Attr1">Value1</attribute> <attribute name="Attr2">Value2</attribute> <attribute name="Attr3">Value3</attribute> </billingAttributes> </billingRecord> </billingHistory>

サービス項目データ

サービス項目の詳細

表 3-28　サービス項目の詳細 API テーブル

領域	例
コア API	**Name による取得** 指定したサービス項目名に関して現行ユーザに割り当てられている、サービス項目インスタンスの詳細情報およびサブスクリプション データが返されます。 また、ユーザが「View Service Items for My Business Units」機能を持っている場合は、そのユーザが属している全 OU に含まれている人に関するサービス項目が返されます。 サービス項目名パラメータは、[Design Service Item] ページの [Name] フィールドに示されるサービス項目の内部テーブル名(SiVirtualMachine など)を受け入れます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/serviceitem/<serviceItemName> Java の例 ServiceItemDTO serviceItems = NSApiClient.getServiceItem().getServiceItemData("<serviceItemName>", null);

表 3-28　サービス項目の詳細 API テーブル(続き)

領域	例

特殊な条件	フィルタ時の特殊文字のサポート
	特殊文字(「=」、「|」、または「/」など)を使用する必要がある場合は、要求のペイロード(本文)で、フィルタとともに POST メソッドを使用します。
	Method: POST
	REST URL:
	/RequestCenter/nsapi/serviceitem/{serviceItemName}
	サンプル ペイロード:
	{
	"filterString":"<columnName1><operator1><value1>|<and|or>|<columnName2><operator2><value2>|<and|or>|<columnName3><operator3><value3>"
	}

	ビュー フィルタ フィルタに使用可能なビューは、My Services および Service Item Manager モジュールの表示内容に対応しています。 ViewName に指定可能な値: • My ServiceItems:現行ユーザが所有しているサービス項目のすべてのインスタンス「View Service Items for My Business Units」機能を持っているユーザは、そのユーザが属している OU 内の他人が所有している項目も表示できます。 • Manage ServiceItems:サービス項目のすべてのインスタンス(ユーザが「Manage Service Item Instances」機能を持っている場合に限り、このビューはデータを返します)。 ソート順やページ サイズなど他の引数も指定する場合は、ViewName 引数を最後に配置する必要があります。 REST URL: http://<ServerURL>/RequestCenter/nsapi/serviceitem/<serviceItemName>?ViewName=<viewName> Java の例 `ServiceItemDTO serviceItems = NSApiClient.getServiceItem().getServiceItemDataWithFilters("<serviceItemName>", "<viewName>");`

表 3-28　サービス項目の詳細 API テーブル(続き)

領域	例
フィルタ	**サービス項目属性フィルタ** 最大 3 つのフィルタがサポートされます。 サービス項目およびサブスクリプションのすべてのカラムがサポートされます。 サービス項目分類名(グループ)またはサービス項目タイプ(シスコが予約しているもの、またはユーザ定義のもの)によるフィルタはサポートされていません。 REST URL: 比較演算子: 数値/日付カラム:=、>、<、>=、<= 文字列カラム:=(大文字小文字の区別あり。like、contains、および starts-with 演算子については後述) 関係演算子:AND、OR(大文字と小文字を区別しない、オーダーの優先順位はサポートされていません) 区切り文字 = | 使用可能なカラム:サービス項目およびサブスクリプションの全カラム http://<ServerURL>/RequestCenter/nsapi/serviceitem/<serviceItemName>/<columnName1><operator1><value1>[|<AND|OR>|<columnName2><operator2><value2>][|<AND|OR>|<columnName3><operator3><value3>] Java の例 `String filter = "<columnName1><operator1><value1>|<and|or>|<columnName2><operator2><value2>|<and|or>|<columnName3><operator3><value3>"; ServiceItemDTO serviceItems = NSApiClient.getServiceItem().getServiceItemDataWithFilters("<serviceItemName>", filter);`
	文字列カラムのフィルタ %(starts with)演算子用のフィルタ。 *(contains)演算子用のフィルタ。 (ends with)演算子はサポートされていません。 例: Name=service* Name=*g* Name=*g:許可されません。 REST URL: http://<ServerURL>/RequestCenter/nsapi/serviceitem/<serviceItemName>/<columnName>=<wildcardValue> Java の例 `ServiceItemDTO serviceItems = NSApiClient.getServiceItem().getServiceItemDataWithFilters("<serviceItemName>",`

表 3-28　サービス項目の詳細 API テーブル(続き)

領域	例
	日付カラムの属性 日付フィールドの値は mm-dd-yyyy 形式にする必要があります。 比較演算子:=、>、<、<=、>= 例: SubmittedDate=12-10-2010 REST URL: http://<ServerURL>/RequestCenter/nsapi/serviceitem/<serviceItemName>/<columnName><operator><mm-dd-yyyy> Java の例 `ServiceItemDTO serviceItems = NSApiClient.getServiceItem().getServiceItemDataWithFilters("<serviceItemName>", "<columnName><operator><mm-dd-yyyy>");`
ソート カラム	サービス項目の属性:すべてのテーブル カラム。 サブスクリプション:Assigned Date、Display Name、ID(内部 ID)、Requisition ID、Submit Date。

応答 XML	`<serviceitem totalCount="x" recordSize="3" startRow="1" id="62">`
	`<logicName></logicName>`
	`<name></name>`
	`<subscription>`
	`. . .`
	`<assignedDate> </assignedDate>`
	`<assignedDateRaw></assignedDateRaw>`
	`<customerID></customerID>`
	`<customerName> </customerName>`
	`<displayName> </displayName>`
	`<id></id>`
	`<organizationalUnitID></organizationalUnitID>`
	`<organizationalUnitName> </organizationalUnitName>`
	`<requisitionEntryID></requisitionEntryID>`
	`<requisitionID></requisitionID>`
	`<serviceItemClassificationID></serviceItemClassificationID>`
	`<serviceItemTypeID></serviceItemTypeID>`
	`<serviceItemTypeName> </serviceItemTypeName>`
	`<submittedDate> </submittedDate>`
	`<submittedDateRaw></submittedDateRaw>`
	`. . .`
	`</subscription>`
	`</serviceitem>`

表 3-28　サービス項目の詳細 API テーブル（続き）

領域	例

v2 CRUD サービス項目の NSAPI	version2（v2）nsAPI のサービス項目は、階層的なサービス項目をサポートするために実行されます。すべてのv2 API は JSON 形式のみをサポートします（入力および応答）。
	読み取り、作成、更新、削除について v2 サービス項目 NSAPI でサポートされる URL は次のとおりです。
	GET API-1:
	http://<Server URL>/RequestCenter/nsapi/v2/serviceitem/{serviceItemLogicName}
	特定のサービス項目タイプの論理名のすべてのサービス項目レコードを、フィルタなしで取得します。
	URL 例：http://<Server URL>/RequestCenter/nsapi/v2/serviceitem/SiDesktop
	出力例：：
	HTTP ステータス コード：200 OK
	{
	"serviceitem": {
	"id": 109,
	"name": "Desktop",
	"logicName": "SiDesktop",
	"startRow": 1,
	"recordSize": 2,
	"totalCount": 2,
	"serviceItemData": [
	{
	"items": [
	{
	"Name": "1 desktop",
	"SingleStr": "Single 1",
	"MultiStr": [
	"1-First",
	"1-2nd",
	"1-3rd"
	]
	}
	],

表 3-28　サービス項目の詳細 API テーブル(続き)

領域	例

	"subscription": {
	"accountID": 0,
	"agreementID": 0,
	"customerID": 1,
	"id": 1,
	"organizationalUnitID": 1,
	"requisitionEntryID": 0,
	"requisitionID": 0,
	"submittedDateRaw": 1412223018210
	}
	
	]
	}
	}
	}

表 3-28　サービス項目の詳細 API テーブル（続き）

領域	例

	GET API-2:
	http://<ServerURL>/RequestCenter/nsapi/v2/serviceitem/{serviceItemLogicName}/{filters}
	特定のサービス項目タイプの論理名の、フィルタ基準に一致するすべてのサービス項目のレコードを取得します。
	URL 例:
	http:<Server URL>/RequestCenter/nsapi/v2/serviceitem/SiDesktop/Name=1desktop
	出力例: :
	HTTP ステータス コード:200 OK
	{
	"serviceitem": {
	"id": 109,
	"name": "Desktop",
	"logicName": "SiDesktop",
	"startRow": 1,
	"recordSize": 2,
	"totalCount": 2,
	"serviceItemData": [
	{
	"items": [
	{
	"Name": "1 desktop",
	"SingleStr": "Single 1",
	"MultiStr": [
	"1-First",
	"1-2nd",
	"1-3rd"
	]
	}
	],

表 3-28　サービス項目の詳細 API テーブル(続き)

領域	例

	"subscription": {
	"accountID": 0,
	"agreementID": 0,
	"customerID": 1,
	"id": 1,
	"organizationalUnitID": 1,
	"requisitionEntryID": 0,
	"requisitionID": 0,
	"submittedDateRaw": 1412223018210
	}
	
	]
	}
	}
	}

	POST API
	http://<ServerURL>/RequestCenter/nsapi/v2/serviceitem/{serviceItemLogicName}
	URL 例:http://<Server URL>/RequestCenter/nsapi/v2/serviceitem/SiDesktop
	入力例:(「serviceItemData」のオブジェクトの構造は、GET 呼び出し応答および POST/PUT 呼び出しの入力と同じです)。
	{
	“serviceitem": {
	"serviceItemData": {
	"Name": "2 desktop",
	"SingleStr": "Single 2",
	"MultiStr": ["2-1st",
	"2-2nd",
	"2-3rd"],
	"Harddisk": {
	"Name": "2 Disk",
	"Size": "200"
	},

表 3-28　サービス項目の詳細 API テーブル（続き）

領域	例

	"Harddisk": {
	"Name": "2 Disk",
	"Size": "200"
	},
	"MultiHardDisk": [{
	"Name": "2-1 MDisk",
	"MSize": "21"
	},
	{
	"Name": "2-2 MDisk",
	"MSize": "22"
	}],
	"CPU": {
	"Name": "HP2",
	"Speed": "20"
	},
	"MultiCPU": [{
	"Name": "MHP1",
	"MSpeed": "100"
	},
	{
	"Name": "MHP2",
	"MSpeed": "200"
	}]
	},
	"subscription": {
	"loginID": "admin",
	"ouname": "Site Administration"}
	}
	}

表 3-28　サービス項目の詳細 API テーブル(続き)

領域	例

	出力例:
	応答 HTTP ステータス コード:200 OK
	出力: {
	"nsapi-response": {
	"status-messages": [
	{
	"code": "SI_SUCCESS_001",
	"value": "Service item successfully updated."
	}
	]
	}
	}

	DELETE API
	http://<ServerURL>/RequestCenter/nsapi/v2/serviceitem/{serviceItemLogicName}
	特定の入力のサービス項目を削除します。
	URL 例:http://<Server URL>/RequestCenter/nsapi/v2/serviceitem/SiDesktop
	入力例:
	{
	"serviceitem":{
	"serviceItemData": {
	"Name": "14 desktop"
	}
	}
	}
	出力例::
	応答 HTTP ステータス コード:204

注意事項と検証

1. これらの API をすべて使用するには、サービス項目の定義が存在している必要があります。
2. URL の「serviceItemLogicName」は有効な名前である必要があります。
3. 作成/更新/削除の入力には、次の検証が必要です。
   a. サービス項目名は必須です。
   b. 「作成」では、「更新」および「削除」とは異なり、同じサービス項目が以前にあってはなりません。
   c. 入力 JSON の属性が SI 定義に存在する必要があります。

d. 親レコードでレコードを参照するためには、「作成」および「更新」API のいずれにも、その参照レコードが存在している必要があります。

e. 包含レコードは親レコードの更新シナリオに存在する必要はありません。

f. 親レコードの更新時に、「名前」を除く、それに含まれる子レコード データを更新できます。このためには所有者情報が一致する必要があります。

g. 子レコードにこれまで一度も所有者 SI がなかった場合は、ユーザが親レコードの更新を使用して、既存の SI レコードを包含レコードとして別の既存の SI（親）に追加する必要があります。
例：デスクトップ（親 SI）と CPU（子 SI）、およびこれら両方に別々に作成されるインスタンス（Desktop1 および CPU1）。ここで、Desktop1 を更新し、CPU1 がこれまで所有者 SI を持ったことがない場合は、CPU1 をこれに接続できます。この呼び出しによって、CPU1 は Desktop1 の包含レコードになります。

h. 更新時に、入力のすべての SI レコード（親レコードまたはそれに含まれる子レコード）はマージされ、保存される必要があります。既存の属性データは、それらの属性が更新 API の入力で指定されていない場合は、変更しないでください。

i. 参照レコードが親レコード更新で更新されることはありません。これらは、参照されるだけです。

j. 親レコードに 1 つの包含オブジェクトがあり、その包含レコードを親に含めない場合、含まれるのが単一値であれば入力として空のオブジェクト {} を渡し、含まれるのが複数値であれば空の配列 [] を渡す必要があります。
例：Harddisk - 単一値が含まれるタイプの属性 MultiHardDisk - 複数値が含まれるタイプの属性。

```
{
"serviceitem": {
"serviceItemData": {
"Name": "1 desktop",
"Harddisk": {
},
"MultiHardDisk": [
]
}
}
```

k. 既存の参照レコードを削除するために親を更新するには、入力構造が前述と同じである必要があります。

l. 「Name」は包含/参照の子レコードに必須です。

m. 更新 API の入力で属性が指定されない場合、その属性データは更新の実行によって変更されません。

4. 削除 API の場合、親レコードとそれに関連していたすべての子レコードが削除されますが、親レコードと関連していない参照レコードは削除されません。

5. 必要に応じ、**newscale.properties** ファイルで **serviceitem.nsapi.rbac.check** プロパティを *false* に設定することで、nsAPI に基づいてサービス項目に対する RBAC チェックをバイパスすることができます。

すべてのサービス項目

表 3-29　すべてのサービス項目 API テーブル

領域	例
コア API	**すべての項目の取得** すべてのサブスクリプションおよびサービス項目データを表示します。 デフォルトでは、現行ユーザに割り当てられているサービス項目のみがフィルタされます。また、ユーザが「View Service Items for My Business Units」機能を持っている場合は、そのユーザが属している全 OU に含まれている他人のサービス項目も返されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/serviceitems/serviceitemsubscription Java の例 `ServiceItemSubscriptionList AllserviceItems = NSApiClient.getServiceItemSubscription().getServiceItemSubscriptionData(null) ;`

特殊な条件	フィルタ時の特殊文字のサポート
	特殊文字(「=」、「|」、または「/」など)を使用する必要がある場合は、要求のペイロード(本文)で、フィルタとともに POST メソッドを使用します。
	Method: POST
	REST URL:
	/RequestCenter/nsapi/serviceitems/serviceitemsubscription
	サンプル ペイロード:
	{
	"filterString":"<columnName1><operator1><value1>|<and|or>|<columnName2><operator2><value2>|<and|or>|<columnName3><operator3><value3>"
	}

表 3-29　すべてのサービス項目 API テーブル(続き)

領域	例
フィルタ	**ビュー フィルタ** フィルタに使用可能なビューは、My Services および Service Item Manager モジュールのそれぞれに表示される内容に対応しています。 ViewName に指定可能な値: • My ServiceItems:現行ユーザが所有しているすべてのサービス項目「View Service Items for My Business Units」を持っているユーザの場合は、そのユーザが属している OU 内の他人が所有している項目も取得します。 • Manage ServiceItems:すべてのサービス項目(ユーザが「サービス項目インスタンスの管理(Manage Service Item Instances)」権限を持っている場合に限り、このビューはデータを返します)。 ソート順やページ サイズなど他の引数も指定する場合は、ViewName 引数を最後に配置する必要があります。 REST URL: http://<ServerURL>/RequestCenter/nsapi/serviceitems/serviceitemsubscription?ViewName=<viewName> Java の例 `ServiceItemSubscriptionList AllserviceItems = NSApiClient.getServiceItemSubscription().getServiceItemSubscriptionFilterData ("<viewName>");`
	サブスクリプション フィルタ Service Item Subscription テーブルの全カラム用のフィルタ。 最大 3 つのフィルタがサポートされます。 REST URL: 比較演算子: • 数値/日付カラム:=、>、<、>=、<= • 文字列カラム:=(大文字小文字の区別あり。like、contains、および starts-with 演算子については後述) 関係演算子:AND、OR(大文字と小文字を区別しない、オーダーの優先順位はサポートされていません) フィルタ区切り文字:| サポートされるカラム:すべてのカラム http://<ServerURL>/RequestCenter/nsapi/serviceitems/serviceitemsubscription/<columnName1><operator1><value1>[|<AND|OR>|<columnName2><operator2><value2>][|<AND|OR>|<columnName3><operator3><value3>] Java の例 `String filter = "<columnName1><operator1><value1>|<and|or>|<columnName2><operator2><value2>|<and|or>|<columnName3><operator3><value3>"; ServiceItemSubscriptionList AllserviceItems = NSApiClient.getServiceItemSubscription().getServiceItemSubscriptionFilterData (filter);`

表 3-29　すべてのサービス項目 API テーブル(続き)

領域	例

	文字列カラムのフィルタ
	%(starts with)演算子がサポートされています。
	*(contains)演算子がサポートされています。
	(ends with)演算子はサポートされていません。
	例:
	`Name=service*`
	`Name=*g*`
	`Name=*g -- not allowed`
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/serviceitems/serviceitemsubscription/<columnName>=<wildcardValue>
	Java の例
	`String filter = "<columnName>=<wildcardValue>";`
	`ServiceItemSubscriptionList AllserviceItems = NSApiClient.getServiceItemSubscription().getServiceItemSubscriptionFilterData (filter);`

	日付カラムのフィルタ
	日付フィールドの値は mm-dd-yyyy 形式にする必要があります。
	比較演算子:=、>、<、<=、>=
	例:
	`SubmittedDate=12-10-2010`
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/serviceitems/serviceitemsubscription/<columnName><operator><mm-dd-yyyy>
	Java の例
	`String filter = "<columnName><operator><mm-dd-yyyy>";`
	`ServiceItemSubscriptionList AllserviceItems = NSApiClient.getServiceItemSubscription().getServiceItemSubscriptionFilterData (filter);`

表 3-29　すべてのサービス項目 API テーブル(続き)

領域	例
ソート カラム	Assigned Date、Customer ID、Display Name、Organizational Unit ID、Requisition ID、Requisition Entry ID、Service Item Classification ID、Service Item ID、Service Item Type ID、Service Item Type Name、Submitted Date
応答 XML	<AllServiceItems totalCount="x" recordSize="y" startRow="z"> <serviceitemsubscription displayName=" " id=""> . . . <serviceItemTypeName> </serviceItemTypeName> <organizationalUnitID></organizationalUnitID> <assignedDate> </assignedDate> <requisitionID></requisitionID> <submittedDate> </submittedDate> <submittedDateRaw></submittedDateRaw> <assignedDateRaw></assignedDateRaw> <customerID></customerID> <requisitionEntryID></requisitionEntryID> <serviceItemID></serviceItemID> <serviceItemTypeID></serviceItemTypeID> <organizationalUnitName> </organizationalUnitName> <customerName> </customerName> <serviceitem id=""> <logicName> </logicName> <name> </name> <serviceItemData rowId=""> <serviceItemAttribute name=" "> </serviceItemAttribute> . . . </serviceitemsubscription> </AllServiceItems>

表 3-29　すべてのサービス項目 API テーブル(続き)

領域	例

	すべてのサービス項目を取得する API
	この API は、すべてのサービス項目を取得するのに使用します。
	次のオプションを使用できます。
	• SearchFilter : name:equals:laptop
	• startRow および recordsPerPage : 改ページ調整
	• SortBy : カラム名
	• sortDir : 並べ替えの方向
	(注)　sortBy と sortDir は、同時に使用する必要があります。
	REST URL:
	http://<Server URL>/ RequestCenter/nsapi/serviceitem/v1/mdrdatatable/serviceitemlist?
	サンプル出力 :
	{
	"Result":{
	"recordsReturned":10,
	"totalRecords":1,
	"startIndex":1,
	"records":[
	{
	"serviceItemId":"1",
	"assignedDate":"25/05/2016 7:49 AM",
	"owner":"admin admin",
	"serviceItemTypeId":"107",
	"accountName":"",
	"classification":"UCS Director",
	"submittedDate":"25/05/2016 7:49 AM",
	"organizationUnitName":"Site Administration",
	"requisitionId":"",
	"serviceItemTypeName":"APIC Container",
	"encServiceItemId":"6M+JgN1m4ZpqG79GquDakQ==",
	"agreementName":"",
	"encServiceItemTypeId":"LAZcCouqBeYcZYsLd0hpRQ==",
	"name":"Customer",
	"id":"1"
	}
	]
	}
	}

表 3-29　すべてのサービス項目 API テーブル(続き)

領域	例

	すべてのサービス項目の CSV をダウンロード
	すべてのサービス項目の CSV をダウンロード
	次のオプションを使用できます。
	• SearchFilter：name:equals:laptop
	• startRow および recordsPerPage：改ページ調整
	• SortBy：カラム名
	• sortDir：並べ替えの方向
	(注)　sortBy と sortDir は、同時に使用する必要があります。
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/serviceitem/v1/mdrdatatable/serviceitemlist/CSV
	サンプル出力:
	`Service Item Group, Service Item Name, Service Item Type, Requisition ID, Submitted Date, Assigned Date, Organizational Unit, Account, Agreement, Customer`
	`"UCS Director", SuFen5, Container, 8, 04/19/2016 7:02 AM, 04/19/2016 7:02 AM, UCSD::fen::Default Group, , vu1, vu1`

表 3-29　すべてのサービス項目 API テーブル(続き)

領域	例

	履歴情報の取得
	サービス項目の履歴を取得
	ペイロード コンテンツ タイプ:application/x-www-form-urlencoded(フォーム データ)。
	フォーム データは次のように送信されます。serviceItemTypeId=149&serviceItemId=1
	• serviceItemTypeId:149
	• serviceItemId:1
	REST URL:
	http://<Server URL> /RequestCenter/nsapi/serviceitem/v1/myservices/assetlist/history?
	出力例
	`[`
	`  {`
	`  "name": "",`
	`  "operation": "Update",`
	`  "requisitionId": "",`
	`  "serviceId": "",`
	`  "submitDate": "06/06/2016"`
	`  },`
	`  {`
	`  "name": "",`
	`  "operation": "Create",`
	`  "requisitionId": "",`
	`  "serviceId": "",`
	`  "submitDate": "06/06/2016"`
	`  }`
	`]`

表 3-29　すべてのサービス項目 API テーブル(続き)

領域	例

	分類ツリーの取得
	サービス項目の分類ツリーを取得
	REST URL:
	http://<ServerURL>
	/RequestCenter/nsapi/serviceitem/v1/sim/serviceitemtypestree/classificationdatatypetree?
	出力例
	{
	"data":[
	{
	"data":[
	{
	"hasChildren":false,
	"iconCls":"sitinstance",
	"id":"107",
	"leaf":true,
	"module":"itemType",
	"moduleId":"107S",
	"text":"APIC Container",
	"vsocTenantNodeType":""
	}
	],
	"hasChildren":true,
	"iconCls":"sitfolder",
	"id":"12",
	"leaf":false,
	"module":"classification",
	"moduleId":"12C",
	"text":"Cloud Infrastructure",
	"vsocTenantNodeType":""
	}
	]
	}

サービス項目に対する作成/更新/削除 API

次の表では、サービス項目に対して作成/更新/削除操作を行うための nsAPI URL について説明します。要求のコンテンツ タイプは、application/xml または application/json です。デフォルトの応答のコンテンツ タイプは要求のコンテンツ タイプと同じになります。つまり入力データが xml 形式であれば応答データも xml 形式になります。これらは、追加のクエリー パラメータ「responseType」を指定することにより、変更できます。パラメータで指定できる値は「xml」または「json」です。たとえば、xml で入力した場合、URL で responseType-json を指定すると、応答は json 形式で返されます。

表 3-30　サービス項目に対する作成/更新/削除 API

領域	例

Core API	サービス項目の作成
	POST URL:
	http://<ServerURL>/RequestCenter/nsapi/serviceitem/process
	xml:
	<serviceitem>
	<name>custom_sitype</name>
	<serviceItemData>
	<serviceItemAttribute name="Name">sit1</serviceItemAttribute>
	<serviceItemAttribute name="Field1">a</serviceItemAttribute>
	<serviceItemAttribute name="Field2">b</serviceItemAttribute>
	</serviceItemData>
	</serviceitem>
	json:
	{
	"serviceitem" : {
	"name" : "custom_sitype",
	"serviceItemData" : {
	"serviceItemAttribute" : [{
	"name" : "Name",
	"value" : "sit1"
	}, {
	"name" : "Field1",
	"value" : "a"
	}, {
	"name" : "Field2",
	"value" : "b"
	}]
	}
	}
	}

表 3-30　サービス項目に対する作成/更新/削除 API（続き）

領域	例
	サービス項目の更新 PUT URL: http://<ServerURL>/RequestCenter/nsapi/serviceitem/process xml: <serviceitem> <name>custom_sitype</name> <serviceItemData> <serviceItemAttribute name="Name">sit1</serviceItemAttribute> <serviceItemAttribute name="Field1">a</serviceItemAttribute> <serviceItemAttribute name="Field2">b</serviceItemAttribute> </serviceItemData> </serviceitem>
	サービス項目の削除 DELETE URL: http://<ServerURL>/RequestCenter/nsapi/serviceitem/process xml: <serviceitem> <name>custom_sitype</name> <serviceItemData> <serviceItemAttribute name="Name">sit1</serviceItemAttribute> </serviceItemData> </serviceitem>
	サービス項目のカスタム操作 サービス項目に対してカスタム操作を実行します。 PUT URL: http://<ServerURL>/RequestCenter/nsapi/serviceitem/process?operation=test xml: <serviceitem> <name>custom_sitype</name> <serviceItemData> <serviceItemAttribute name="Name">sit1</serviceItemAttribute> <serviceItemAttribute name="Field1">a</serviceItemAttribute> <serviceItemAttribute name="Field2">b</serviceItemAttribute> </serviceItemData> </serviceitem>

表 3-30　サービス項目に対する作成/更新/削除 API（続き）

領域	例

	サービス項目を作成し、特定の OU に関連付ける
	サービス項目を作成し、要求データでサービス項目のサブスクリプション詳細を指定することにより、サービス項目を特定の OU や個人などと関連付けます。
	POST URL:
	http://<ServerURL>/RequestCenter/nsapi/serviceitem/process
	xml:
	<pre><serviceitem>
	    <name>custom_sitype</name>
	    <serviceItemData>
	      <serviceItemAttribute name="Name">sit1</serviceItemAttribute>
	      <serviceItemAttribute name="Field1">a</serviceItemAttribute>
	      <serviceItemAttribute name="Field2">b</serviceItemAttribute>
	      <subscription>
	        <loginID>admin</loginID>
	        <ouname>testOU1</ouname>
	        <accountName>account1</accountName>
	        <requisitionEntryID>6</requisitionEntryID>
	      </subscription>
	    </serviceItemData>
	  </serviceitem></pre>
	json:
	<pre>{
	  "serviceitem" : {
	    "name" : "custom_sitype",
	    "serviceItemData" : {
	      "serviceItemAttribute" : [{
	        "name" : "Name",
	        "value" : "sit1"
	      }, {
	        "name" : "Field1",
	        "value" : "a"
	      }, {
	        "name" : "Field2",
	        "value" : "b"
	      }],
	      "subscription" : {
	        "loginID" : "admin",
	        "ouname" : "testOU1",
	        "accountName" : "account1",
	        "requisitionEntryID" : ""
	      }
	    }
	  }
	}</pre>

フィルタ	フィルタはサポートされません。

表 3-30　サービス項目に対する作成/更新/削除 API（続き）

領域	例
ソート カラム	ソートは許可されません。

応答 XML	xml:
	<nsapi-response>
	<status-message>
	<value>Service item [sit1] successfully created.</value>
	</status-message>
	</nsapi-response>
	
	<nsapi-response>
	<status-message>
	<value>Operation 'Test' of service item [sit1] successfully performed.</value>
	</status-message>
	</nsapi-response>
	
	json:
	{
	"nsapi-response" : {
	"status-messages" : [{
	"value" : "Service item [sit140211] successfully created."
	}]
	}
	}

権限の付与または取り消し

nsAPI は、サービス項目および標準の権限の割り当てと取り消しに対してのみサポートされます。SI インスタンス データに対するレコード レベルの権限の定義は、SIM モジュールから、または nsAPI を使ってのみ可能です（Org Designer からはできません）。

表 3-31　権限の付与および取り消しAPI テーブル

領域	例
コア API	権限の付与 サービス項目および標準に対する権限を付与します。 REST URL: http://<ServerURL>/RequestCenter/nsapi/rbac/extensibleschema/grantpermission SIM RBAC に関連するすべての権限は、この nsAPI によって実行できます。 • 特定の SI グループ/すべての SI グループ/標準グループの定義の権限をロールに付与します。 • 特定のサービス項目/すべてのサービス項目/標準の定義の権限をロールに付与します。 • 特定のサービス項目/すべてのサービス項目/標準のインスタンス データの権限をロールに付与します。 • 特定の SI インスタンス/レコードに対する SI インスタンス レベル/レコード レベルの権限を、個人/OU/グループ/ロール/アカウントに付与します。 • すべてのオブジェクトに対する「Maintain Billing Rates（課金レートの維持）」権限を付与します。
	権限の取り消し サービス項目および標準に対する権限を取り消します。 REST URL: http://<ServerURL>/RequestCenter/nsapi/rbac/extensibleschema/revokepermission SIM RBAC に関連するすべての権限は、この nsAPI によって実行できます。 • ロールに付与されている、特定の SI グループ/すべての SI グループ/標準グループの定義の権限を取り消します。 • ロールに付与されている、特定のサービス項目/すべてのサービス項目/標準の定義の権限を取り消します。 • ロールに付与されている、特定のサービス項目/すべてのサービス項目/標準のインスタンス データの権限を取り消します。 • 個人/OU/グループ/ロール/アカウントに付与されている、特定の SI インスタンス/レコードに対する SI インスタンス レベル/レコード レベルの権限を取り消します。 • すべてのオブジェクトに対する「Maintain Billing Rates（課金レートの維持）」権限を取り消します。
フィルタ	フィルタはサポートされません。

表 3-31　権限の付与および取り消し API テーブル（続き）

領域	例
ソート カラム	ソートは許可されません。

要求 XML	権限の付与:
	ヘッダー:<Username> <password> および <Content-Type=application/xml>
	Method=POST
	`<?xml version="1.0" encoding="UTF-8"?>`
	`<grantpermission scope="SpecificObject/AllObjects"`
	`object="Instance/Definition/Group"`
	`recipient="Role/OrgUnit/Person/Group/ProjectAccount/TenantAccount"`
	`recipientName=<Name of the Recipient> recipientType="ServiceTeam/BuisnessUnit`
	`- Applicable only if the recipient is set to OrgUnit"`
	`domain="ServiceItem/Standard" permission=<Logical Names> instanceName=<Name`
	`of the Instance  if object="Instance"> objectName=<Name of the attribute`
	`object> />`
	例:
	`<?xml version="1.0" encoding="UTF-8"?>`
	`<grantpermission scope="SpecificObject" object="Instance" recipient="OrgUnit"`
	`recipientName="test5466" recipientType="ServiceTeam" domain="ServiceItem"`
	`permission="mdr_record_read" instanceName="Cap" objectName="SravItem"/>`
	権限の取り消し:
	ヘッダー:<Username> <password> および <Content-Type=application/xml>
	Method=POST
	`<?xml version="1.0" encoding="UTF-8"?>`
	`<revokepermission scope="SpecificObject/AllObjects"`
	`object="Instance/Definition/Group"`
	`recipient="Role/OrgUnit/Person/Group/ProjectAccount/TenantAccount"`
	`recipientName=<Name of the Recipient> recipientType="ServiceTeam/BuisnessUnit`
	`- Applicable only of the recipient is set to OrgUnit"`
	`domain="ServiceItem/Standard" permission=<Logical Names> instanceName=<Name`
	`of the Instance  if object="Instance"> objectName=<Name of the attribute`
	`object> />`
	例:
	`<?xml version="1.0" encoding="UTF-8"?>`
	`<revokepermission scope="SpecificObject" object="Instance"`
	`recipient="OrgUnit" recipientName="test5466" recipientType="ServiceTeam"`
	`domain="ServiceItem" permission="mdr_record_read" instanceName="Cap"`
	`objectName="SravItem"/>`
	特定のオブジェクトにユーザが付与/取り消しできる権限の詳細については、特定のオブジェクトにユーザが付与/取り消しできる権限の表を参照してください。

表 3-32　特定のオブジェクトにユーザが付与/取り消しできる権限

範囲	UI ラベル	権限	object= “Definition“, domain= “ServiceItem “	object= “Group“, domain= “ServiceItem “	object= “Instance“, domain= “ServiceItem “	object= “Definition“ , domain= “Standard“	object= “Group“, domain= “Standard“	object= “Instance“, domain= “Standard“
AllObjects/ SpecificObject	[読み取り (Read)]	mdr_record_read	N/A	N/A	AllObjects には適用されない。インスタンス名が NULL ではない場合は、特定のオブジェクトに適用可能	N/A	N/A	N/A
	書き込み (Write)	mdr_record_write	N/A	N/A	AllObjects には適用されない。インスタンス名が NULL ではない場合は、特定のオブジェクトに適用可能	N/A	N/A	N/A
	読み取り定義	mdr_datatype_read	適用可能	N/A	N/A	適用可能	N/A	N/A
	ReadWrite 定義	mdr_datatype_write	適用可能	N/A	N/A	適用可能	N/A	N/A
	すべてのインスタンス データの読み取り	content_def_read_allcontentdata	N/A	N/A	適用可能	N/A	N/A	適用可能
	すべてのインスタンス データの読み取り/書き込み	content_def_write_allcontentdata	N/A	N/A	適用可能	N/A	N/A	適用可能
	個人の BU のすべてのインスタンス データの読み取り	allinstanceddata_my_bu_read	N/A	N/A	適用可能	N/A	N/A	N/A
	個人の BU のすべてのインスタンス データの読み取り/書き込み	allinstanceddata_my_bu_write	N/A	N/A	適用可能	N/A	N/A	N/A

表 3-32　特定のオブジェクトにユーザが付与/取り消しできる権限(続き)

範囲	UI ラベル	権限	object= “Definition“, domain= “ServiceItem “	object= “Group“, domain= “ServiceItem “	object= “Instance“, domain= “ServiceItem “	object= “Definition“ , domain= “Standard“	object= “Group“, domain= “Standard“	object= “Instance“, domain= “Standard“
	個人の BU およびその下位単位のすべてのインスタンス データの読み取り	allinstanceddata_my_bu_hierarachy_read	N/A	N/A	適用可能	N/A	N/A	N/A
	個人の BU およびその下位単位のすべてのインスタンス データの読み取り/書き込み	allinstanceddata_my_bu_hierarachy_write	N/A	N/A	適用可能	N/A	N/A	N/A
	個人のテナント アカウントのすべてのインスタンス データの読み取り	allinstanceddata_my_tenantaccount_read	N/A	N/A	適用可能	N/A	N/A	N/A
	個人のテナント アカウントのすべてのインスタンス データの読み取り/書き込み	allinstanceddata_my_tenantaccount_write	N/A	N/A	適用可能	N/A	N/A	N/A
	個人のプロジェクト アカウントのすべてのインスタンス データの読み取り	allinstanceddata_my_projectaccount_read	N/A	N/A	適用可能	N/A	N/A	N/A
	個人のプロジェクト アカウントのすべてのインスタンス データの読み取り/書き込み	allinstanceddata_my_projectaccount_write	N/A	N/A	適用可能	N/A	N/A	N/A
	自己がサブスクライブするすべてのインスタンス データの読み取り	allinstancedata_subscribed_self_read	N/A	N/A	N/A	N/A	N/A	N/A

表 3-32　特定のオブジェクトにユーザが付与/取り消しできる権限（続き）

範囲	UI ラベル	権限	object= “Definition“, domain= “ServiceItem “	object= “Group“, domain= “ServiceItem “	object= “Instance“, domain= “ServiceItem “	object= “Definition“, domain= “Standard“	object= “Group“, domain= “Standard“	object= “Instance“, domain= “Standard“
	自己がサブスクライブするすべてのインスタンス データの読み取り/書き込み	allinstanceddata_subscribed_self_write	N/A	N/A	N/A	N/A	N/A	N/A
	新しいインスタンス データの作成	create_new_service_item	N/A	N/A	適用可能	N/A	N/A	N/A
	新しい標準のインスタンスデータの作成	create_new_standard_instance_data	N/A	N/A	N/A	N/A	N/A	適用可能
	このグループのすべての定義の読み取り	mdr_class_read	N/A	適用可能	N/A	N/A	適用可能	N/A
	このグループのすべての定義の読み取り/書き込み	mdr_class_write	N/A	適用可能	N/A	N/A	適用可能	N/A
AllObjects	MaintainBillingRates	billingrates_maintain	object=“Definition“	N/A	domain=“Billing“	object=“Definition“	N/A	N/A

標準（Standards）

表 3-33　標準 API テーブル

領域	例
コア API	**Name による取得** デフォルトでは、指定した標準の最初の 50 個のエントリが返されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/standard/<standardName> Java の例 `StandardDTO standards = NSApiClient.getStandard().getStandardData("<standardName>", null);`

表 3-33　標準API テーブル(続き)

領域	例

フィルタ	最大 3 つのフィルタがサポートされます。
	REST URL:
	比較演算子:
	• 数値/日付カラム:=、>、<、>=、<=(大文字小文字の区別あり)
	• 文字列カラム:=(like、contains、および starts-with 演算子については後述)
	関係演算子:AND、OR(大文字と小文字を区別しない、オーダーの優先順位はサポートされていません)
	フィルタ区切り文字:|
	サポートされるカラム:すべてのカラム
	http://<ServerURL>/RequestCenter/nsapi/standard/<standardName>/<columnName1><operator1><value1>[|<AND|OR>|<columnName2><operator2><value2>][|<AND|OR>|<columnName3><operator3><value3>]
	Java の例
	`String filter:`
	`"<columnName1><operator1><value1>|<and|or>|<columnName2><operator2><value2>|<and|or>|<columnName3><operator3><value3>";`
	`StandardDTO standards =`
	`NSApiClient.getStandard().getStandardDataWithFilters("<standardName>", filter);`

	文字列カラムのフィルタ
	%(starts with)演算子がサポートされています。
	*(contains)演算子がサポートされています。
	(ends with)演算子はサポートされていません。
	例:
	`Name=service*`
	`Name=*g*`
	`Name=*g -- not allowed`
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/standard/<standardName>/<columnName>=<wildcardName>
	Java の例
	`StandardDTO standards =`
	`NSApiClient.getStandard().getStandardDataWithFilters("<standardName>", "<wildcardName>");`

表 3-33　標準 API テーブル(続き)

領域	例

	日付カラムのフィルタ
	日付フィールドの値は mm-dd-yyyy 形式にする必要があります。
	比較演算子:=、>、<、<=、>=
	例:
	`SubmittedDate=12-10-2010`
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/standard/<standardName>/<columnName><operator><mm-dd-yyyy>
	Java の例
	`StandardDTO standards = NSApiClient.getStandard().getStandardDataWithFilters("<standardName>", "<columnName><operator><mm-dd-yyyy>");`

ソート カラム	すべてのテーブル カラム。

応答 XML	`<standard totalCount="x" startRow="y" recordSize="z" id="a">`
	`<loginName></loginName>`
	`<name></name>`
	`<standardData rowId="">`
	`. . .`
	`<standardAttribute name="id" />`
	`. . .`
	`<standardURL><a ></a></standardURL>`
	`<standardURLOnly> </standardURLOnly>`
	`. . .`
	`</standardData>`
	`</standard>`

サービス カタログ(Service Catalog)のデータ

カスタム コンテンツ(Custom Content)

カスタム コンテンツは、サービス ポータルのコンテンツのソースとして機能する、ユーザ定義のテーブルにより構成されます。これらのテーブルは、標準と同様にポートレットのデータ ソースとして参照されます。

表 3-34　カスタム コンテンツ API テーブル

領域	例
コア API	**Name による取得** デフォルトでは、指定したテーブルの最初の 50 個のエントリが返されます。 REST URL: http://<ServerURL>/RequestCenter/nsapi/customcontent/<customContentName> Java の例 `CustomContentDTO customs = NSApiClient.getCustomContent().getcontentData("<customContentName>", null);`

フィルタ	すべてのカスタム コンテンツ カラムがサポートされています。
	%（starts with）演算子がサポートされています。
	（ends with）演算子はサポートされていません。
	例:
	`Name=custom*`
	`Name=*g -- not allowed`
	REST URL:
	比較演算子:=、>、<、>=、<=（大文字小文字の区別あり）
	関係演算子:AND、OR（大文字小文字の区別あり）
	フィルタ区切り文字:|
	サポートされるカラム:すべてのカラム
	http://<ServerURL>/RequestCenter/nsapi/customcontent/<customContentName>/<columnName1><operator1><value1>[|<AND|OR>|<columnName2><operator2><value2>][|<AND|OR>|<columnName3><operator3><value3>]
	http://<ServerURL>/RequestCenter/nsapi/customcontent/<customContentName>/<columnName>=<wildcardName>
	http://<ServerURL>/RequestCenter/nsapi/customcontent/<customContentName>/<columnName><operator><mm-dd-yyyy>
	Java の例
	`String filter = "<columnName><operator><value>";`
	`CustomContentDTO customs = NSApiClient.getCustomContent().getcustomContentDataWithFilters("CoPortalContent", filter);`

ソート カラム	すべてのテーブル カラム

応答 XML	`<customContent totalCount="x" startRow="y" recordSize="z" id="a">`
	`  <logicName></logicName>`
	`  <name></name>`
	`  . . .`
	`  <customContentData rowId="">`
	`  <customContentAttribute name=" "></customContentAttribute>`
	`  . . .`
	`  </customContentData>`
	`</customContent>`

全カラム名のリスト、および各エンティティの説明については、『*Cisco Prime Service Catalog Designer Guide*』の「Designing Portals」の章にある「Reference Data」の項を参照してください。

テナント管理

表 3-35　テナント管理 API テーブル

領域	例

Get API	**ユーザ ロールを取得する API**
	Method: GET
	REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/tenant/v2/userroles
	サンプル応答:
	{
	"roles": {
	"startRow": 0,
	"totalCount": 0,
	"recordSize": 0,
	"role": [
	{
	"roleId": 73,
	"roleName": "Service Administrator",
	"statusId": 1,
	"roleTypeId": 2,
	"logicName": "service-administrator",
	"inherited": false,
	"parentId": 0
	},
	{
	"roleId": 1336,
	"roleName": "Team Administrator",
	"statusId": 1,
	"roleTypeId": 2,
	"logicName": "team-administrator",
	"inherited": false,
	"parentId": 0
	}
	]
	}
	}

Get API	**テナント管理設定を取得する API** Method: GET REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/tenant/v2/tenantconfiguration

表 3-35　テナント管理 API テーブル(続き)

領域	例

	サンプル応答:
	<pre>{
	    "ArrayList": [
	        {
	            "name": "TenantType",
	            "value": "Multi"
	        },
	        {
	            "name": "TenantID",
	            "value": "System generated UUID"
	        },
	        {
	            "name": "TenantName",
	            "value": "User ID"
	        },
	        {
	            "name": "Organization",
	            "value": "CSV list imported"
	        },
	        {
	            "name": "Billing",
	            "value": "Active"
	        },
	        {
	            "name": "BillingType",
	            "value": "Credit Card"
	        },
	        {
	            "name": "BillingDept",
	            "value": "Dynamic join to people data in LDAP"
	        },
	        {
	            "name": "FinApp",
	            "value": "Yes"
	        },
	        {
	            "name": "FinAppType",
	            "value": "Add approvers using CSV"
	        },
	        {
	            "name": "EntApp",
	            "value": "Yes"
	        },
	        {
	            "name": "TenantApp",
	            "value": "Yes"
	        },
	        {
	            "name": "TenantAppType",
	            "value": "Parent Tenant Admin (Multi Only)"
	        }
	    ]
	}</pre>**(注)**　設定を行っていない場合、応答は **ArrayList: [ ]** となります。

表 3-35　テナント管理 API テーブル(続き)

領域	例

Get API	**管理者メンバー情報を含むカードおよびリストを表示するため、階層的チームとチーム詳細を取得する API**
	Method: GET
	REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/tenant/v2/id/{teamId}
	サンプル要求:
	http://<ServerURL>/RequestCenter/nsapi/directory/tenant/v2/id/1040
	サンプル応答:
	<pre>{
	"TeamList": {
	"startRow": 0,
	"totalCount": 2,
	"recordSize": 0,
	"teams": [
	{
	"teamName": "Root Team",
	"teamId": 39,
	"statusId": 0,
	"status": "Active",
	"parentId": 0,
	"parentName": "",
	"accountId": 22,
	"hasChild": false,
	"memberCount": 2,
	"createdOn": 1477035094013,
	"createdOnString": "10/21/2016 12:31 AM",
	"adminMembers": [
	{
	"memberName": "root cisco",
	"memberId": 25,
	"status": "Active",
	"email": "root@cisco.com",
	"type": "Team Admin"
	}
	],
	"rootTenant": "T122",
	"parentTenant": "",
	"rootTenantDisplayName": "Root Team",
	"currentTenantAdmin": false,
	"currentParentTenantAdmin": true
	},
	{
	"teamName": "Tenant A",
	"teamId": 1040,
	"statusId": 0,
	"status": "Active",</pre>

表 3-35　テナント管理 API テーブル(続き)

領域	例


```
                "parentId": 0,
                "parentName": "",
                "accountId": 23,
                "hasChild": true,
                "memberCount": 2,
                "createdOn": 1478049861410,
                "createdOnString": "11/01/2016 6:24 PM",
                "childTeams": [
                    {
                        "teamName": "Tenant B",
                        "teamId": 1041,
                        "statusId": 0,
                        "status": "Active",
                        "parentId": 1040,
                        "parentName": "Tenant A",
                        "accountId": 23,
                        "hasChild": true,
                        "memberCount": 2,
                        "createdOn": 1478050009523,
                        "createdOnString": "11/01/2016 6:26 PM",
                        "adminMembers": [
                            {
                                "memberName": "adminB cisco",
                                "memberId": 35,
                                "status": "Active",
                                "email": "adminB_tl@cisco.com",
                                "type": "Team Admin"
                            }
                        ],
                        "rootTenant": "T123",
                        "parentTenant": "T123",
                        "rootTenantDisplayName": "Tenant A",
                        "currentTenantAdmin": false,
                        "currentParentTenantAdmin": false
                    }
                ],
                "adminMembers": [
                    {
                        "memberName": "UserA UserA",
                        "memberId": 14,
                        "status": "Active",
                        "email": "usera_tl@cisco.com",
                        "type": "Team Admin"
                    }
                ],
                "rootTenant": "T123",
                "parentTenant": "",
                "rootTenantDisplayName": "Tenant A",
                "currentTenantAdmin": false,
                "currentParentTenantAdmin": false
            }
        ]
    }
}
For the first time loading data, you have to pass teamid=0.
For example: http://<ServerURL>/RequestCenter/nsapi/directory/tenant/v2/id/0
```


(注)　関連付けられたテナント アカウントを持っていない場合、**{}** という空の応答が受信されます。

表 3-35　テナント管理 API テーブル(続き)

領域	例

Get API	**管理者メンバー情報を含むリストを表示するため、チーム リストを取得する API**
	searchPattern は *name* という形式になります。
	SortBy：Name、Status、CreatedOn
	SortDir：Asc、Desc
	Method: GET
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/directory/tenant/v2/teamlist?startRow={startRow}&recordSize={recordSize}&sortBy={sortBy}&sortDir={sortDir}&name={searchName}
	サンプル要求：
	http://<ServerURL>/RequestCenter/nsapi/directory/tenant/v2/teamlist?startRow=1&recordSize=20&sortBy=Nams&sortDir=Asc&name=*
	サンプル応答：
	<pre>{
	"TeamList": {
	"startRow": 1,
	"totalCount": 4,
	"recordSize": 20,
	"teams": [
	{
	"teamName": "Root Team",
	"teamId": 39,
	"statusId": 0,
	"status": "Active",
	"parentId": 0,
	"parentName": "",
	"accountId": 22,
	"hasChild": false,
	"memberCount": 2,
	"createdOn": 1477035094013,
	"createdOnString": "10/21/2016 12:31 AM",
	"adminMembers": [
	{
	"memberName": "root cisco",
	"memberId": 25,
	"status": "Active",
	"email": "root@cisco.com",
	"type": "Team Admin"
	}
	],
	"rootTenant": "T122",
	"parentTenant": "",
	"rootTenantDisplayName": "Root Team",
	"currentTenantAdmin": false,
	"currentParentTenantAdmin": false
	},</pre>

表 3-35　テナント管理 API テーブル(続き)

領域	例

	{
	"teamName": "Tenant A",
	"teamId": 1040,
	"statusId": 0,
	"status": "Active",
	"parentId": 0,
	"parentName": "",
	"accountId": 23,
	"hasChild": true,
	"memberCount": 2,
	"createdOn": 1478049861410,
	"createdOnString": "11/01/2016 6:24 PM",
	"adminMembers": [
	{
	"memberName": "UserA UserA",
	"memberId": 14,
	"status": "Active",
	"email": "usera_tl@cisco.com",
	"type": "Team Admin"
	}
	],
	"rootTenant": "T123",
	"parentTenant": "",
	"rootTenantDisplayName": "Tenant A",
	"currentTenantAdmin": false,
	"currentParentTenantAdmin": false
	},
	{
	"teamName": "Tenant B",
	"teamId": 1041,
	"statusId": 0,
	"status": "Active",
	"parentId": 1040,
	"parentName": "Tenant A",
	"accountId": 23,
	"hasChild": true,
	"memberCount": 2,
	"createdOn": 1478050009523,
	"createdOnString": "11/01/2016 6:26 PM",
	"adminMembers": [
	{
	"memberName": "adminB cisco",
	"memberId": 35,
	"status": "Active",
	"email": "adminB_tl@cisco.com",
	"type": "Team Admin"
	}
	],
	"rootTenant": "T123",
	"parentTenant": "T123",
	"rootTenantDisplayName": "Tenant A",
	"currentTenantAdmin": false,
	"currentParentTenantAdmin": false
	},

表 3-35　テナント管理 API テーブル(続き)

領域	例

```
                    {
                "teamName": "Tenant C",
                "teamId": 1042,
                "statusId": 0,
                "status": "Active",
                "parentId": 1041,
                "parentName": "Tenant B",
                "accountId": 23,
                "hasChild": false,
                "memberCount": 1,
                "createdOn": 1478050137437,
                "createdOnString": "11/01/2016 6:28 PM",
                "adminMembers": [
                    {
                        "memberName": "adminC cisco",
                        "memberId": 36,
                        "status": "Active",
                        "email": "adminC_tl@cisco.com",
                        "type": "Team Admin"
                    }
                ],
                "rootTenant": "T123",
                "parentTenant": "T124",
                "rootTenantDisplayName": "Tenant A",
                "currentTenantAdmin": false,
                "currentParentTenantAdmin": false
            }
        ]
    }
}
```

表 3-35　テナント管理 API テーブル(続き)

領域	例

Get API	**親チーム ID でサブチームを取得する API**
	searchPattern は *name* という形式になります。
	SortBy：Name、Status、CreatedOn、または MemberCount
	SortDir：Asc、Desc
	Method: GET
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/directory/tenant/v2/id/{teamid}/teamlist?startRow={startRow}&recordSize={recordSize}&sortBy={sortBy}& sortDir={sortDir}&name={searchName}
	サンプル要求:
	http://<ServerURL>/RequestCenter/nsapi/directory/tenant/v2/id/3/teamlist?startRow=1&recordSize=10&sortBy=Name& sortDir=Asc&name=*
	サンプル応答:
	<pre>{
	"TeamList": {
	"startRow": 1,
	"totalCount": 2,
	"recordSize": 10,
	"teams": [
	{
	"teamName": "Program Management",
	"teamId": 4,
	"statusId": 1,
	"status": "Active",
	"parentId": 3,
	"parentName": "Cisco Team",
	"accountId": 0,
	"hasChild": false,
	"memberCount": 3,
	"createdOn": 1475721937253,
	"createdOnString": "10/05/2016 7:45 PM"
	},
	{
	"teamName": "UX Team",
	"teamId": 5,
	"statusId": 2,
	"status": "Inactive",
	"parentId": 3,
	"parentName": "Cisco Team",
	"accountId": 0,
	"hasChild": false,
	"memberCount": 0,
	"createdOn": 1475721947930,
	"createdOnString": "10/05/2016 7:45 PM"
	}
	]
	}
	}</pre>

表 3-35　テナント管理 API テーブル(続き)

領域	例

Get API	**チーム メンバー リストを取得する API**
	SortBy：Name、Email、または Status
	SortDir：Asc、Desc
	SearchName：*name*
	Method: GET
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/directory/tenant/v2/id/{teamId}/memberlist?startRow={startRow}&recordSize={recordSize}&sortBy={sortBy}& sortDir={sortDir}&name={searchName}
	サンプル要求:
	http://<ServerURL>/RequestCenter/nsapi/directory/tenant/v2/id/3/memberlist?startRow=1&recordSize=10&sortBy=Name&sortDir=ASc&name=*
	サンプル応答:
	`{`
	`  "TeamMemberList": {`
	`  "startRow": 1,`
	`  "totalCount": 3,`
	`  "recordSize": 10,`
	`  "teamMembers": [`
	`  {`
	`  "memberName": "admin  admin",`
	`  "memberId": 1,`
	`  "status": "Active",`
	`  "email": "thanes_demo@cisco.com",`
	`  "type": "Member"`
	`  },`
	`  {`
	`  "memberName": "Ryan  Marfone",`
	`  "memberId": 5,`
	`  "status": "Active",`
	`  "email": "Ryan@cisco.com",`
	`  "type": "Team Admin"`
	`  },`
	`  {`
	`  "memberName": "test  test",`
	`  "memberId": 3,`
	`  "status": "Active",`
	`  "email": "test@test.com",`
	`  "type": "Team Admin"`
	`  }`
	`  ]`
	`  }`
	`}`

表 3-35　テナント管理 API テーブル(続き)

領域	例

Post API	**メンバーを AS チーム管理者に昇格させる API**
	この API は、次のことを行います。
	• ユーザへの team-admin ロールの割り当て
	• 既存チームの管理者ユーザから team-admin ロールの割り当てを解除
	• 既存チームの管理者ユーザへの it-user ロールの割り当て
	Method: POST
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/directory/tenant/v2/id/{teamId}/people/id/{personId}/promoteTeamAdmin
	サンプル要求:
	http://<ServerURL>/RequestCenter/nsapi/directory/tenant/v2/id/3/people/id/10/promoteTeamAdmin
	サンプル応答:
	`{ "status-message": { "code": "TA_054", "value": "User is promoted to Team Admin successfully" } }`
	エラー応答:
	`{ "status-message": { "code": " TA_030", "value": "Cannot promote user to a team admin, user is already a team admin" } }`

表 3-35　テナント管理 API テーブル(続き)

領域	例

Post API	**Cloud Center ユーザの API キーの更新**
	Method: POST
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/directory/people/id/{personId}/updateapikey
	サンプル要求:
	http://<ServerURL>/RequestCenter/nsapi/directory/people/id/10/updateapikey
	サンプル応答:
	{
	"status-message": {
	"code": "TA_054",
	"value": "API Key updated successfully"
	}
	}
	エラー応答:
	{
	"status-message": {
	"code": "TA_054",
	"value": "Cloud Center User ID does not exist"
	}
	}
	エラー応答:
	{
	"status-message": {
	"code": "TA_054",
	"value": "Cloud Center Profile for the User does not exist"
	}
	}

Catalog Deployer API

表 3-36　Catalog Deployer API

領域	例
Rex API	**サービス データの取得** Method: GET REST URL: http://<ServerURL>/RequestCenter/rexapi/entity/service/{name} 要求ヘッダーのパラメータ:ユーザ名およびパスワード 応答:rex xml
	カテゴリ データの取得 Method: GET REST URL: http://<ServerURL>/RequestCenter/rexapi/entity/category/{name} 要求ヘッダーのパラメータ:ユーザ名およびパスワード 応答:rex xml

表 3-36　Catalog Deployer API（続き）

領域	例
Rex API	**組織データの取得** Method: GET REST URL: http://<ServerURL>/RequestCenter/rexapi/entity/organizationalunit/{name} 要求ヘッダーのパラメータ:ユーザ名およびパスワード 応答:rex xml rex xml の詳細については、「Rex XML 構造の例」を*参照してください。*
	グループ データの取得 Method: GET REST URL: http://<ServerURL>/RequestCenter/rexapi/entity/group/{name} 要求ヘッダーのパラメータ:ユーザ名およびパスワード 応答:rex xml rex xml の詳細については、「Rex XML 構造の例」を*参照してください。*
	個人データの取得 Method: GET REST URL: http://<ServerURL>/RequestCenter/rexapi/entity/person/{name} 要求ヘッダーのパラメータ:ユーザ名およびパスワード 応答:rex xml rex xml の詳細については、「Rex XML 構造の例」を*参照してください。*
	キュー データの取得 Method: GET REST URL: http://<ServerURL>/RequestCenter/rexapi/entity/queue/{name} 要求ヘッダーのパラメータ:ユーザ名およびパスワード 応答:rex xml rex xml の詳細については、「Rex XML 構造の例」を*参照してください。*
	役職データの取得 Method: GET REST URL: http://<ServerURL>/RequestCenter/rexapi/entity/functionalposition/{name} 要求ヘッダーのパラメータ:ユーザ名およびパスワード 応答:rex xml rex xml の詳細については、「Rex XML 構造の例」を*参照してください。*

表 3-36　Catalog Deployer API (続き)

領域	例
Rex API	**ロール データの取得** Method: GET REST URL: http://<ServerURL>/RequestCenter/rexapi/entity/role/{name} 要求ヘッダーのパラメータ:ユーザ名およびパスワード 応答:rex xml rex xml の詳細については、「Rex XML 構造の例」を*参照してください。*
	電子メール テンプレートのデータの取得 Method: GET REST URL: http://<ServerURL>/RequestCenter/rexapi/entity/emailtemplate/{name} 要求ヘッダーのパラメータ:ユーザ名およびパスワード 応答:rex xml rex xml の詳細については、「Rex XML 構造の例」を*参照してください。*
	エージェント データの取得 Method: GET REST URL: http://<ServerURL>/RequestCenter/rexapi/entity/agent/{name} 要求ヘッダーのパラメータ:ユーザ名およびパスワード 応答:rex xml rex xml の詳細については、「Rex XML 構造の例」を*参照してください。*
	rex データの取得 Method: POST REST URL: http://<ServerURL>/RequestCenter/rexapi/entity/getdata 要求ヘッダーのパラメータ:ユーザ名およびパスワード 応答:rex xml rex xml の詳細については、「Rex XML 構造の例」を*参照してください。* **(注)**　戻される xml にはヘッダー パラメータは含まれません。
	データのインポート Method: POST REST URL: http://<ServerURL>/RequestCenter/rexapi/entity/putdata 要求ヘッダーのパラメータ:ユーザ名およびパスワード 要求の本文:rex xml rex xml の詳細については、「Rex XML 構造の例」を*参照してください。*

Rex XML 構造の例

```
Service:
<rex>
    <Header requestAction="put"/>
    <ServiceDefinitions>
        <ServiceDefinition name="Service A" serviceGroupName="0 Service Icons"
catalogName="" type="notspecified" isActive="true" isReportable="false"
isEntitlement="false" isOrderable="true" computePrice="false" authorizations="site"
canStartLater="false" dateQuality="tbd" guid="EBC80F54-01E3-45C1-8DF2-1C35C1165838"
orderingMode="Add review enabled" paginationViewMode="Classic View"
hideInServiceCatalog="false" isTemplate="false" showOrderSummary="false" maxQuantity="0"
isBaseService="false">
            <Transaction actionRequested="getXml" actionResult="Succeeded"
actionResultCode="0" actionTaken="fetched" detail=""/>
            <Duration value="0.0" units="hours" isDefinedDuration="false"/>
            <Billing rate="0.0" type="none"/>
            <Expensing expenseCode=""/>
            <Authorization>
                <ServiceGroupReview>
                    <AuthRoles/>
                    <Escalations/>
                </ServiceGroupReview>
                <ServiceGroupAuthorization>
                    <AuthRoles/>
                    <Escalations/>
                </ServiceGroupAuthorization>
            </Authorization>
            <Presentation>
                <Description>
                    <Icon documentId="0"/>
                </Description>
               <SectionOne isEnabled="false"/>
                <SectionTwo isEnabled="false"/>
                <SectionThree isEnabled="false"/>
            </Presentation>
            <PositionNames>
                <PositionName name="Author" type="Service" serviceDefinitionName="Service
A" serviceDefinitionGuid="EBC80F54-01E3-45C1-8DF2-1C35C1165838" objectType="Person"
instanceName="admin"/>
            </PositionNames>
            <Categories>
                <ConsumerServicesCategories/>
                <ServicePortfoliosCategories/>
            </Categories>
            <AssociatedKeywords/>
            <AssociatedObjectives/>
            <Pricing price="0.0" pricingSchema="fixedprice" pricingSchemaString="Fixed
Price" priceDisplaySchema="" priceDisplaySchemaString="">
                <Description>
                    <Text>
                        <![CDATA[]]>
                    </Text>
                </Description>
            </Pricing>
            <Costing>
                <Costs/>
           </Costing>
            <ServiceLevel/>
            <Bundling canBeBundled="true" isABundle="false">
                <ServiceContainsServices/>
                <ServiceIsContainedByServices/>
            </Bundling>
```

```
            <ServicePreRequisites/>
            <ServiceAccessories/>
            <Manufacturer name="" partNumber=""/>
            <Supplier name="" type="" partNumber=""/>
            <Plans>
                <Plan>
                    <PlanAttributes type="delivery" name="PrimaryDeliveryPlan"
subtaskFlow="sequential" maxDepth="1" hoursPerDay="8.0" isAutomaticStart="true">
                        <PlanSubject>
                            <![CDATA[<s ID="11"><p>#Name#</p></s>]]>
                        </PlanSubject>
                        <PlanMonitorTask>
                            <PlanMonitorWorkflowRoles>
                                <PlanMonitorWorkflowRole assignmentMethod="none"
expression=""/>
                            </PlanMonitorWorkflowRoles>
                            <Notifications>
                                <Notification eventType="plantaskcanceled"
emailTemplate=""/>
                            </Notifications>
                        </PlanMonitorTask>
                    </PlanAttributes>
                    <Tasks/>
                    <Escalations/>
                </Plan>
            </Plans>
            <DictionaryFormAccessPolicies/>
            <ServiceForms/>
            <RelatedEntities>
                <Forms/>
                <ServiceGroups>
                    <ServiceGroupName name="0 Service Icons"
guid="956A6E7B-1FB6-4B49-B258-C58F5AC727D3"/>
                </ServiceGroups>
                <Categories/>
                <EmailTemplates/>
                <Objectives/>
                <ServiceDefinitions/>
                <Keywords/>
                <Persons/>
                <OrganizationalUnits/>
                <Queues/>
                <Agents/>
            </RelatedEntities>
            <PermissionsAssignment>
                <OperationsWeArePermitted/>
                <OperationsTheyArePermitted>
                    <Operation name="service_change_rights"
displayName="service_change_rights">
                        <Objects/>
                    </Operation>
                    <Operation name="service_change_delivery"
displayName="service_change_delivery">
                        <Objects/>
                    </Operation>
                    <Operation name="service_change_forms"
displayName="service_change_forms">
                        <Objects/>
                    </Operation>
                    <Operation name="service_change_presentation"
displayName="service_change_presentation">
                        <Objects/>
                    </Operation>
```

```
                    <Operation name="service_order_service"
displayName="service_order_service">
                        <Objects/>
                    </Operation>
                </OperationsTheyArePermitted>
            </PermissionsAssignment>
            <Images/>
            <ServiceExtensions/>
            <BaseTemplateID>
                <![CDATA[0]]>
            </BaseTemplateID>
        </ServiceDefinition>
    </ServiceDefinitions>
</rex>
```

REST API 要求のレート制限の設定

Prime Service Catalog の管理者はレート リミッタにより、許可される要求数および許可される要求の時間帯（秒数）をグローバルに設定します。

要求（外部アプリケーションからの要求、または Prime Service Catalog UI 内の要求）は、設定に従って受け入れられ、設定の制限を超えると拒否されます。

要求が拒否された場合、レート制限に対する HTTP 応答コード- *429 Too Many Requests (RFC 6585)* が要求ユーザに返されます。

次のグローバルなデフォルト値がレート制限機能のために管理者によって設定されます。

- **enabled**：このフラグは **true**（REST 呼び出しのレート制限を有効にする）または **false**（REST 呼び出しのレート制限を無効にする）に設定されます（API または UI ベースの nsAPI REST 呼び出しに適用されるグローバル設定）。このフラグはデフォルトで無効ですが、nsApiRateLimit.json ファイルを設定してレート制限を有効にできます。フラグの設定は、グローバルおよびすべての上書きに均等に適用されます。さらに、UI または外部アプリケーションからの nsAPI REST 呼び出しに対して、別のレート制限を定義できます。
- **nsApiRateLimit**：許可される呼び出しの最大数を指定します。
- **nsApiRateIntervalSecs**：**nsApiRateLimit** により設定される要求が許可される時間帯（秒数）を指定します。

さらに、管理者は次の修飾子を使用して、グローバル（デフォルト）値を上書きできます（特定の URL、1 つ以上のメソッドが上書きされます）。

（注） 時間帯は上書きできません。上書きでは毎回、グローバル設定を使用します。

- **url**：nsAPI の Rest URI は完全 URL か、またはワイルドカード（*）を使った URL です。ワイルド カードが使用されない場合は、入力 URI と設定される URL が完全一致すると想定されます。設定のクエリー パラメータはすべて、レート リミッタにより省略されます。
- **method**：HTTP メソッドは 1 つまたは複数の ["GET", "PUT", "POST", "DELETE"] を含む文字列リテラル配列です。
- **nsApiRateLimit**：許可される呼び出しの最大数を指定します。これは、この URI だけに対する同様のグローバル設定を上書きします。

次の JSON 構造に従ってレート制限を設定します。

(注)　JSON の検証は手動で行います。設定ファイル nsAPIRateLimit.json を保存する際に、アプリケーションは JSON の検証を行いません。このため、無効な JSON の場合にアプリケーションが動作しないことがあります。

```
{
    "api": {
        "config": {
            "enabled": "false",
            "nsApiRateLimit": 2,
            "nsApiRateIntervalSecs": 10,
            "overrides": [
                {
                    "url": "/nsapi/serviceitem/SiServItem*",
                    "method": [
                        "GET"
                    ],
                    "nsApiRateLimit": 5
                },
                {
                    "url": "/nsapi/serviceitem/*",
                    "method": [
                        "GET"
                    ],
                    "nsApiRateLimit": 3
                },
                {
                    "url": "/nsapi/serviceitem/*",
                    "method": [
                        "POST",
                        "PUT"
                    ],
                    "nsApiRateLimit": 3
                },
                {
                    "url": "/nsapi/transaction/authorizations/*",
                    "method": [
                        "GET",
                        "PUT",
                        "POST",
                        "DELETE"
                    ],
                    "nsApiRateLimit": 8
                },
                {
                    "url":
"/nsapi/transaction/authorizations/ViewName=Authorizations%20for%20Self| Status=Approved",
                    "method": [
                        "GET",
                        "PUT",
                        "POST",
                        "DELETE"
                    ],
                    "nsApiRateLimit": 3
                },
                {
                    "url": "/nsapi/definition/categories",
                    "method": [
                        "GET",
                        "PUT",
                        "POST",
                        "DELETE"
```

```
                    ],
                    "nsApiRateLimit": 5
                }
            ]
        }
    },
    "ui": {
        "config": {
            "enabled": "false",
            "nsApiRateLimit": 4,
            "nsApiRateIntervalSecs": 10,
            "overrides": [
                {
                    "url": "/nsapi/transaction/requisitions/*",
                    "method": [
                        "GET",
                        "PUT",
                        "POST",
                        "DELETE"
                    ],
                    "nsApiRateLimit": 3
                },
                {
                    "url": "/nsapi/transaction/authorizations/*",
                    "method": [
                        "GET",
                        "PUT",
                        "POST",
                        "DELETE"
                    ],
                    "nsApiRateLimit": 3
                },
                {
                    "url": "/nsapi/definition/categories",
                    "method": [
                        "GET",
                        "PUT",
                        "POST",
                        "DELETE"
                    ],
                    "nsApiRateLimit": 5
                }
            ]
        }
    }
}
```

エラー メッセージ

例外の性質に応じて、異なる HTTP 応答コードが nsAPI により返されます。

- HTTP ステータス コード **400** Mandatory Field Missing Error / Redirect URL or EntityID from metadata already exists / Duplicate IDP configuration name：フィールド未入力エラー、メタデータ情報がすでに存在する、または IDP 設定名の重複。
- HTTP ステータス コード **401**（Unauthorized）および XML エラー応答メッセージ「**User does not have proper authentication**」、またはメッセージがパスワード ポリシー違反に関連する場合：無効な認証パラメータです、または認証パラメータがありません。

- HTTP ステータス コード **404**（Not Found）および XML エラー応答メッセージ「**Requested resource could not be found**」：指定されたパラメータ/URL の値では、データを取得できませんでした。

  次に、例を示します。

  ```
  nsapi/directory/people/id/-1
  nsapi/directory/people/id/foo
  nsapi/directory/people/id/1000 (there is no person with id = 1000)
  nsapi/directory/people/name/<non existent person>
  nsapi/directory/people/idxyz/1
  ```

- HTTP ステータス コード **403**（Forbidden）および XML エラー応答メッセージ「**The user does not have sufficient permissions to perform the operation this object**」：ユーザには、オブジェクトに対して指定された操作を実行するのに十分なアクセス許可がないため、データを取得できませんでした。
- HTTP ステータス **500**（Internal Error）および XML エラー応答メッセージ「**Internal Error: Invalid parameter values specified or unexpected error**」：不適切なパラメータ、nsAPI 内で発生するその他の例外、またはその他の一般的なサーバ エラー。

  次に、例を示します。

  ```
  nsapi/directory/people?startRow=5000 (non-existent 5000 row)
  nsapi/directory/people?sortBy=wrongColumn&sortDir=Asc (unsupported column)
  nsapi/directory/people?recordSize=-1 (negative or zero value for recordSize)
  ```

- Http ステータス **422**（Unprocessable Entity）、および POST/UPDATE 操作に対する XML 妥当性検査エラー応答メッセージ：要求データの必須フィールドに値がありません。または無効な値が入っています。

nsAPI は、指定されたフィルタについて結果が見つからない場合、または無効なフィルタ条件が指定された場合にエラー メッセージを返します。次に例を示します。

```
<nsapi-response>
<status-code>failed</status-code>
<status-message>Service item custom_sit201 does not exist, none of the service items were
updated. </status-message>
</nsapi-response>
```

nsAPI でのメソッドの実行中に発生したすべての例外について、nsAPI は Java から NSAPIException をスローします。

サポートされる操作の要約

次のチャートに、さまざまなエンティティ タイプでサポートされる操作についての要約を示します。

参照テーブル

エンティティ名	Id による取得	Name による取得	すべて取得	ワイルドカード名前検索	ソート	ページング	REST フィルタから Java クライアント メソッドへの変	更新	フィルタ	ネストされたエンティティ
認証										
認証:ログイン/ログアウト					X		X			
定義データ										
カテゴリ（Categories）	X	X	X	X	X	X	X		[カタログ タイプ（Catalog Type）]	サブカテゴリ、含まれるサービス、含まれる提供
サービス	X	X	X	X	X	X	X		カテゴリ ID、カテゴリ名、キーワード	カテゴリ、キーワード、組み込みサービス
契約	X	X	X	X	X	X	X		名前	
エージェント（Agents）	X	X	X	X	X	X	X		名前	
トランザクション データ										
要求	X		X		X	X	X		ビュー名、ステータス	
要求エントリ	X		X		X	X				
許可	X		X		X	X	X		ビュー名、ステータス	
タスク	X		X		X	X	X		ビュー名（View Name）	
タスク:特定の要求エントリに関するタスク			X		X	X	X		タスクのタイプ、ステータス（Skipped）	組み込みサービスのタスク
タスク:特定の要求に関するマイルストーン			X							
タスク:アクション								X		

エンティティ名	Id による取得	Name による取得	すべて取得	ワイルドカード名前検索	ソート	ページング	REST フィルタから Java クライアント メソッドへの変	更新	フィルタ	ネストされたエンティティ
ディレクトリ データ										
組織	X	X	X	X	X	X	X		OU Type	
Person	X	X	X	X	X	X	X	X	OU 名、グループ名、ロール名	OU、グループ、ロール、住所、連絡先、設定、代理人
グループ（Groups）	X	X	X	X	X	X	X		名前	
アカウント	X	X	X	X	X	X	X		名前	
サービス項目データ										
すべてのサービス項目			X		X	X			任意のカラム	
サービス項目（Service Items）			X		X	X	X		任意のカラム	
標準（Standards）			X		X	X	X		任意のカラム	
サービス カタログ（Service Catalog）のデータ										
カスタム コンテンツ（Custom Content）		X	X	X	X	X	X		任意のカラム	

PART 2

サウスバウンド統合

AMQP との統合

概要

Advanced Message Queuing Protocol（AMQP）は、アプリケーション間や組織間でビジネス メッセージを渡すためのオープン標準です。Service Designer モジュールを使用して、AMQP タスクを定義することができます。AMQP タスクはメッセージ ブローカを介して外部システム（Process Orchestrator）にサービス要求をパブリッシュします。

AMQP はサービス要求を交換に送信し、交換は Service Catalog からのメッセージを受け入れて、それをメッセージ キューにルーティングします。RabbitMQ は AMQP 標準を実装したオープン ソース メッセージ ブローカ ソフトウェアです。外部システム（Process Orchestrator）は、メッセージを取得し、それらを処理するために、必要な API を使用して RabbitMQ にアクセスします。

キュー サービス要求と呼ばれる新しいタスク タイプは Service Designer モジュールの [計画（Plan）] タブで利用でき、サービス要求データをメッセージ ブローカにパブリッシュします。AMQP タスクを使用してデータをパブリッシュする方法の詳細については、『Cisco Prime Service Catalog Designer Guide』を参照してください。

メッセージ キュー

交換はプロデューサ アプリケーションからのメッセージを受け入れ、それをメッセージ キューにルーティングします。事前、事後、および主要 AMQP タスクのために作成される交換はすべて「ファンアウト」タイプで、各タスクについてデフォルト キューが作成され、それらのキューが各タスクにバインドされます。デフォルト キューの名前は <topic-name>_queue です。トピック名は「キュー サービス要求」タスクに対してユーザが [Service Designer] > [計画（Plan）] タブで入力した名前です。

（注） プードル攻撃を防ぐために、RabbitMQ サーバとの Prime Service Catalog 統合は SSL プロトコル TLSv1.2 バージョンのみをサポートしています。TLSv 1.2 が指定されていない場合、クライアントは RabbitMQ サーバに接続してメッセージを利用することができず、次の例外が発生して接続に失敗します。javax.net.ssl.SSLException: Received fatal alert: protocol_version

交換とキューを作成し、メッセージを利用するためのサンプル コードは次のようになります。

```
package amqpProject;

import com.rabbitmq.client.ConnectionFactory;
import com.rabbitmq.client.Connection;
import com.rabbitmq.client.Channel;
```

```
import com.rabbitmq.client.QueueingConsumer;

public class SampleProcess {

  public static void main(String[] argv) throws Exception {
    String exchangeName = "testExchange";
    String queueName = exchangeName+"_queue";
    String brokerIpAddress = "10.142.10.77";
    String userName = "admin";
    String password = "cisco123";
    ConnectionFactory factory = new ConnectionFactory();
   factory.useSslProtocol("TLSv1.2"); !--Included to prevent Poodle attack
    factory.setHost(brokerIpAddress);
    factory.setUsername(userName);
    factory.setPassword(password);

    Connection connection = factory.newConnection();

    Channel channel = connection.createChannel();
    channel.exchangeDeclare(exchangeName, "fanout", true, false, null);

    channel.queueDeclare(queueName, true, false, false, null);
    channel.queueBind(queueName, exchangeName, "");

    QueueingConsumer consumer = new QueueingConsumer(channel);
    channel.basicConsume(queueName, true, consumer);
    while (true) {

      QueueingConsumer.Delivery delivery = consumer.nextDelivery();
      String message = new String(delivery.getBody());
      System.out.println(" [x] Received from "+queueName+":-");
      System.out.println(message);

    }
  }
}
```

REST-based nsAPIs

サービスおよびそのタスクに関するメッセージ ブローカの詳細を返すための REST API セットがあります。これにより返される署名および応答を以下に示します。必要な入力値はパス パラメータとして受け入れられるサービス名です。交換の詳細を含む、そのサービスに関するすべての AMQP タスクが発信者に返されます。

サービス固有の API のサンプル応答は次のようになります。

http://localhost:8088/RequestCenter/nsapi/messagebroker/service/<service name>

{"tasks":[{"name":"PRE:amqpTask1","connectionIdentifier":"am1","messageFormat":"JSON","exchangeName":"AQAB_Pre","payloadType":"All Message Details (large)","isAutoComplete":"true","transformationType":"JOLT","outboundTransformationName":"","inboundTransformationName":""},{"name":"amqpTask1","connectionIdentifier":"am1","messageFormat":"JSON","exchangeName":"post_exchange_0307","payloadType":"All Message Details (large)","isAutoComplete":"true","transformationType":"JOLT","outboundTransformationName":"","inboundTransformationName":""},{"name":"POST:amqpTask1","connectionIdentifier":"am1","messageFormat":"JSON","exchangeName":"AQAB_Post","payloadType":"All Message Details (large)","isAutoComplete":"false","transformationType":"JOLT","outboundTransformationName":"","inboundTransformationName":""}]}

Overview API

Overview API は、Service Designer モジュール内で設定されたすべての交換を収集し、RabbitMQ サーバにすでに作成されている交換とキューを検出するためのコールを行います。Overview API は、AMQP サーバに対してルックアップを実行し、キューに関する情報を取得します。出力は JSON 形式で送信されます。

Overview API と noCache パラメータの使用:

- UI を使用して RabbitMQ 情報を指定し、noCache パラメータなしで Overview API を使用すると、情報がキャッシュとデータベースの両方から取得されます。
- ただし、データベースに直接アクセスして RabbitMQ 情報を挿入する場合は、確実に最新情報をフェッチするために、noCache =1 を指定して Overview API を使用する必要があります。

RabbitMQ サーバからの応答例は次のようになります。

http://localhost:8088/RequestCenter/nsapi/messagebroker/overview

{"connections":[{"connectionIdentifier":"am1","host":"10.78.0.247","port":5671},{"connectionIdentifier":"am2","host":"10.78.0.247","port":5671},{"connectionIdentifier":"am3","host":"10.78.0.247","port":5671},{"connectionIdentifier":"AM4","host":"10.78.0.247","port":5671},{"connectionIdentifier":"am5","host":"10.78.0.247","port":5671}]}

http://localhost:8080/RequestCenter/nsapi/messagebroker/overview?connectionIdentifier=am1

{"rabbitmq_version":"3.6.1","username":"admin","password":"UhP0zDGRoC2R7isv9qCQ6A==","ipAddress":"10.78.0.247","recoveryInterval":300000,"vhost":"/","inboundQueue":"psc_inbound_queue","authorization_key":"owL7ViRfE4Sce0aG1jSzZInkIVw5CsM7acQ5rlswpXzF/kForoUjlfrVVUOuA+CqSFSTuJlLQ5GRUMmzzxbV2xtrMvhGed6x1WU08MbDLKydSSY5UQvoAS7aZ0dROdXvF+G4uD4nAlQ6HSIc6dBct3M+dDJcm02z9OshvMaCmkvZa380B8/MbbBhu5Q3FntzWkAVY/FobU9gvlidoDt1Ty0CmAgfvPbP6joLXYaTPHjaqjYaBaX2Y4m+1V7Wm3Rb+oLpHZkCVm7Pr1zlLByPs6d+qaQShIfr0yeXXbZrQNh3s8qHl+YbYFRtNin/JEeLa6pY5J19m6zUV78n2BstRQ==","message-time-to-live":"300000","useSSL":true,"port":"5671","exchanges":[{"name":"exchange2","vhost":"/","type":"fanout","created":"false"},{"name":"HeatStack","vhost":"/","type":"fanout","created":"true"},{"name":"exchange3","vhost":"/","type":"fanout","created":"false"}],"queues":[{"name":"exchange2_queue","vhost":"/","created":"false"},{"name":"HeatStack_queue","vhost":"/","created":"true"},{"name":"exchange3_queue","vhost":"/","created":"false"}],"certificate":"-----BEGIN CERTIFICATE-----\nMIIC5DCCAcygAwIBAgIBATANBgkqhkiG9w0BAQsFADATMREwDwYDVQQDDAhNeVRlc3RDQTAeFw0x\nNjA2MTExMTUwMTJaFw0xNzA2MTExMTUwMTJaMCcxFDASBgNVBAMMCyQoaG9zdG5hbWUpMQ8wDQYD\nVQQKDAZzZXJ2ZXIwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCq8v2yJT7tv+nOwFSo\nEE1c0oVg3skd86JN7jVJaz/mMOyJjDmf1147iUwZPMTTCB34ovYUXFkw+a+0ext2WRHgQLTMPvVO\nA86jwuPd/bhUxXg8jeEfE4V/1Seci9Xz+5VxqCybCNOzJQ12/vLXvIJK43U/+1GdXnpWxFaF0yd0\nht3iUy6mfUAHfNMI2SOfJwbbdUa

MyD0/Krsiu+X+vFQBUDmM7Y0priItiDVDq7rxug2UOPACzzMG\n5yJ5aJjNLSlJRwKKst/jjvesqHIgWNo0qKvk
TET3tIVsKDi1Fn9IdKQuuoI1n225+58cWSANmZ5M\n4BdSIf4z6QRuKliBRi55AgMBAAGjLzAtMAkGA1UdEwQCM
AAwCwYDVR0PBAQDAgUgMBMGA1UdJQQM\nMAoGCCsGAQUFBwMBMA0GCSqGSIb3DQEBCwUAA4IBAQC4L9fk4kuu/d
pLeFWNUhb5Uyk6AqbTRAyD\nlq1m11E09EhmTH7cnoFsz0ELBDppASIUADSb9jmUeNKJtjW94gq8luSem1Z8lQz
UCOCE6rsaznrw\nm9jJO7gXA5SSmy7PgdokdhbeTzlSYA3kkkR5ZE8M403Qv5cEPREqnshs6f6bQSm8tzSNETy3
OyHL\npUo7YVTBCfJMQ/e2nZxCJSDuEL6QaIL4kmEYeu8j/1RplBAofMkDfDe+yMx2MJYl+MopVggGexpa\nMqy
bCchrDTJ/8sI/R18Ld80TQ2Km70sQqvNVenCpGCtgGIcupRWaAOpsQH0PNaUK+lcwYRIvf0VS\n09QE\n-----E
ND CERTIFICATE--"}

パブリック キー GUID を使用してクレデンシャルを暗号化する

Prime Service Catalog は、nsAPI に渡されるパブリック キーの GUID を使用することにより、安全に AMQP クレデンシャルを返す方法をサポートしています。

外部システムのパブリック キー GUID がクエリ パラメータとして渡された場合、応答内のクレデンシャルは GUID に関連付けられているパブリック キーにより暗号化されます（PO のセキュリティ ストリング形式を使用）。

外部システムのパブリック キー GUID が渡されない場合は、DB で暗号化された文字列がそのまま返されます（注:外部システムではこれを復号化できません）。

暗号化された機密データは、以下に示す AMQP Overview API で返されます。

http://localhost:8088/RequestCenter/nsapi/messagebroker/overview?publicKeyGUID=...

暗号化は、接続ごとに設定された公開キーを使用して実行されます。

認証キー生成 API

create、update、および delete などのサービス項目操作では channel-id よりも認証キーのほうが重要です。しかし、外部のタスクを一意に識別するため、メッセージ内には channel-id も必要です。以下の nsAPI を使用して作成される認証キーは、ユーザに固有です。認証キーは、上述の nsAPI が呼び出されるたびに値が変更されるように見えますが、特定のユーザのすべての AMQP 接続に対して同じです。このキーは、ユーザを認証するため、また、各ユーザに与えられた許可に従ってさまざまなモジュールにアクセスするために使用されます。

認証キーは、以下の nsAPI から取得できます。

**http://<ServerURL>
/RequestCenter/nsapi/messagebroker/overview?connectionIdentifier=<particular amqp connection identifier>**

{"rabbitmq_version":"3.6.1","username":"admin","password":"UhP0zDGRoC2R7isv9qCQ6A==","ipAd
dress":"10.78.0.247","recoveryInterval":300000,"vhost":"/","inboundQueue":"psc_inbound_que
ue","authorization_key":"owL7ViRfE4Sce0aG1jSzZInkIVw5CsM7acQ5rlswpXzF/kForoUjlfrVVUOuA+CqS
FSTuJlLQ5GRUMmzzxbV2xtrMvhGed6x1WU08MbDLKydSSY5UQvoAS7aZ0dROdXvF+G4uD4nAlQ6HSIc6dBct3M+dDJ
cm02z9OshvMaCmkvZa380B8/MbbBhu5Q3FntzWkAVY/FobU9gvlidoDt1Ty0CmAgfvPbP6joLXYaTPHjaqjYaBaX2Y
4m+1V7Wm3Rb+oLpHZkCVm7Pr1zlLByPs6d+qaQShIfr0yeXXbZrQNh3s8qHl+YbYFRtNin/JEeLa6pY5J19m6zUV78
n2BstRQ==","message-time-to-live":"300000","useSSL":true,"port":"5671","exchanges":[{"name
":"exchange2","vhost":"/","type":"fanout","created":"false"},{"name":"HeatStack","vhost":"
/","type":"fanout","created":"true"},{"name":"exchange3","vhost":"/","type":"fanout","crea
ted":"false"}],"queues":[{"name":"exchange2_queue","vhost":"/","created":"false"},{"name":
"HeatStack_queue","vhost":"/","created":"true"},{"name":"exchange3_queue","vhost":"/","cre
ated":"false"}],"certificate":"-----BEGIN
CERTIFICATE-----\nMIIC5DCCAcygAwIBAgIBATANBgkqhkiG9w0BAQsFADATMREwDwYDVQQDDAhNeVRlc3RDQTAe
Fw0x\nNjA2MTExMTUwMTJaFw0xNzA2MTExMTUwMTJaMCcxFDASBgNVBAMMCyQoaG9zdG5hbWUpMQ8wDQYD\nVQQKDA
ZzZXJ2ZXIwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCq8v2yJT7tv+nOwFSo\nEE1c0oVg3skd86JN7j
VJaz/mMOyJjDmf1147iUwZPMTTCB34ovYUXFkw+a+0ext2WRHgQLTMPvVO\nA86jwuPd/bhUxXg8jeEfE4V/1Seci9

Xz+5VxqCybCNOzJQ12/vLXvIJK43U/+1GdXnpWxFaF0yd0\nht3iUy6mfUAHfNMI2SOfJwbbdUaMyD0/Krsiu+X+vF
QBUDmM7Y0priItiDVDq7rxug2UOPACzzMG\n5yJ5aJjNLSlJRwKKst/jjvesqHIgWNo0qKvkTET3tIVsKDi1Fn9IdK
QuuoI1n225+58cWSANmZ5M\n4BdSIf4z6QRuKliBRi55AgMBAAGjLzAtMAkGA1UdEwQCMAAwCwYDVR0PBAQDAgUgMB
MGA1UdJQQM\nMAoGCCsGAQUFBwMBMA0GCSqGSIb3DQEBCwUAA4IBAQC4L9fk4kuu/dpLeFWNUhb5Uyk6AqbTRAyD\n
lq1m11E09EhmTH7cnoFsz0ELBDppASIUADSb9jmUeNKJtjW94gq8luSem1Z8lQzUCOCE6rsaznrw\nm9jJO7gXA5SS
my7PgdokdhbeTzlSYA3kkkR5ZE8M403Qv5cEPREqnshs6f6bQSm8tzSNETy3OyHL\npUo7YVTBCfJMQ/e2nZxCJSDu
EL6QaIL4kmEYeu8j/1RplBAofMkDfDe+yMx2MJYl+MopVggGexpa\nMqybCchrDTJ/8sI/R18Ld80TQ2Km70sQqvNV
enCpGCtgGIcupRWaAOpsQH0PNaUK+lcwYRIvf0VS\n09QE\n-----END CERTIFICATE--"}

JOLT ワークフローを使用した JSON の変換

Prime Service Catalog 12.0 以降では、着信変換もサポートされています。ただし、変換は、サービス項目操作ではサポートされていません。着信メッセージについてサポートされている変換の種類は XSL および JOLT で、発信については XSL、JOLT、および FTL がサポートされています。

変換を作成するための大まかなワークフローは、次のとおりです。

手順 1　[管理（Administration）] > [接続を管理（Manage Connections）] > [AMQP] から AMQP 接続を追加します。詳細については、『*Cisco Prime Service Catalog Administration and Operation Guide*』の「Managing AMQP Connections」の項を参照してください。

手順 2　JOLT 形式で送信変換および着信変換を作成します。詳細については、変換の管理（5-18 ページ）を参照してください。

手順 3　Service Designer で、サービス要求のための AMQP タスクを設定します。詳細については、『*Cisco Prime Service Catalog Designer Guide*』の「Configuring AMQP Tasks for Publishing Service Request to an External System」の項を参照してください。

注意事項:

- 次については、AMQP タスク パラメータのポップアップ ページでオーバーライドすることで、デフォルト値を定義できます。
  - 公開キー
  - メッセージ タイプ、つまり、特定の接続において発信メッセージおよび着信メッセージの処理がデフォルトで行われるメッセージ フォーマット（XML/JSON）。
- メッセージ タイプは AMQP 接続ごとまたはタスクごとに設定できますが、タスクごとの設定が優先されます。
- 取得される変換は、選択されている変換の種類に応じて異なります。つまり、XML が選択されている場合は発信変換および着信変換には XSL 変換のみが使用でき、JSON の場合は JOLT 変換または FTL 変換のいずれかとなります。
- FTL 変換は、発信メッセージについてのみサポートされています。

JOLT を使用した JSON 変換の例

JSON 着信 AMQP の入力:

```
{
    "message": {
        "updateData": {
            "dataValue": [
                {
                    "namePO": "TextField.Text",
                    "valuePO": [
                        "QoE"
                    ],
                    "multiValuedPO": false
                }
            ]
        },
        "addCommentsPO": {
            "comment": [
                "test comment 1",
                "test comment 2",
                "test comment 3"
            ]
        },
        "takeAction": {
            "actionPO": "DONE"
        },
        "channelId": "51515F28-E36A-478C-8773-E35B215CF36C"
    }
}
```

JOLT 変換の仕様:

```
[
  {
    "operation": "shift",
    "spec": {
      "message": {
        "updateData": {
          "dataValue": {
            "*": {
              "namePO": "message.updateData.dataValue.[&1].name",
              "valuePO": "message.updateData.dataValue.[&1].value",
              "multiValuedPO": "message.updateData.dataValue.[&1].multiValued"
            }
          }
        },
        "*": "&1.&0",
        "channelId": "message.channelId"
      }
    }
  },
  {
    "operation": "shift",
    "spec": {
      "message": {
        "takeAction": {
          "actionPO": "message.&1.action"
        },
        "*": "&1.&0",
```

```
        "channelId": "message.channelId"
      }
    }
  },
  {
    "operation": "shift",
    "spec": {
      "message": {
        "addCommentsPO": "message.addComments",
        "*": "&1.&0",
        "channelId": "message.channelId"
      }
    }
  }
]
```

JSON 出力:

次は、上記の変換および入力に基づいて Prime Service Catalog 着信 AMQP コードによって生成された出力で、最終的に処理されたものです。

```
{
  "message" : {
    "addComments" : {
      "comment" : [ "test comment 1", "test comment 2", "test comment 3" ]
    },
    "channelId" : "51515F28-E36A-478C-8773-E35B215CF36C",
    "takeAction" : {
      "action" : "DONE"
    },
    "updateData" : {
      "dataValue" : [ {
        "multiValued" : false,
        "name" : "TextField.Text",
        "value" : [ "QoE" ]
      } ]
    }
  }
}
```

AMQP 着信 XML の例

```
<message channel-id="30FCC75D-83A0-4DD9-E050-007F01019778">
<update-data>
<data-value multi-valued="false">
<name>amqpDict1.field1</name>
<value>a_newer</value>
</data-value>
</update-data>
</message>

<message channel-id="30FCC75D-83A0-4DD9-E050-007F01019778">
<add-comments>
<comment>testing new comments</comment>
</add-comments>
</message>


<message channel-id="30FCC75D-83A0-4DD9-E050-007F01019778">
<take-action action="done">
</take-action>
</message>
```

AMQP 着信 JSON の例

```
{
  "message" : {
    "addComments" : {
      "comment" : [ "test comment 1", "test comment 2", "test comment 3" ]
    },
    "channelId" : "51515F28-E36A-478C-8773-E35B215CF36C",
    "takeAction" : {
      "action" : "DONE"
    },
    "updateData" : {
      "dataValue" : [ {
        "multiValued" : false,
        "name" : "TextField.Text",
        "value" : [ "QoE" ]
      } ]
    }
  }
}
```

着信メッセージ

他社製システムの要求操作およびサービス項目操作からの着信メッセージについては、2 種類の操作がサポートされています。

nsXML において最も重要な要素は channel-id です。これは、外部タスクを一意に識別する ID です。この ID は他社製システムに提供され、対応するデータ アップデートがビジネス エンジンによって正常に適用されるためには、応答に表示される必要があります。

AMQP での要求操作

AMQP でサポートされている要求操作は、Service Link の要求操作と同様です。ただし、AMQP 着信メッセージでは、XML および JSON 形式がサポートされています。各操作の詳細については、着信 nsXML メッセージ（5-25 ページ）を参照してください。

take-action

take-action 操作は、配信タスクを完了としてマークします。

```
<message channel-id="30FCC75D-83A0-4DD9-E050-007F01019778">
<take-action action="done">
</take-action>
</message>
```

update-data

要求のディクショナリ フィールドのデータは、新しい値に更新されます。

```
<message channel-id="30FCC75D-83A0-4DD9-E050-007F01019778">
<update-data>
<data-value multi-valued="false">
<name>amqpDict1.field1</name>
<value>a_newer</value>
</data-value>
</update-data>
</message>
```

add-comment

add-comments メッセージは、要求の [System Comments] セクションにコメントを追加するために使用します。

```
<message channel-id="30FCC75D-83A0-4DD9-E050-007F01019778">
<add-comments>
<comment>testing new comments</comment>
</add-comments>
</message>
```

AMQP でのサービス項目操作

(注)　変換は、サービス項目操作ではサポートされていません。

サービス項目操作では、認証キーを生成してメッセージに含める必要があります。認証キーを生成するには、次の nsAPI を使用します。

```
http://<ServerURL>
/RequestCenter/nsapi/messagebroker/overview?connectionIdentifier=<particular amqp
connection identifier>
```

この API の詳細については、認証キー生成 API（4-4 ページ）を参照してください。次は、AMQP でのサービス項目操作のサンプルです。

作成

create メッセージでは、サービス項目に属性値が追加されます。

```
<message channel-id="A984F860-DBE7-48EB-B3A2-F6A0159B093F"
authorization-key="liDBm1NeRVbVmwbBmAd+cca0/Z8jf863aYOKl3QXSWcYwn5aGuPxn09CjDzv5EKR7/CcJx5
+2gu1MhUopAuiX3WwjX/DP1Xw3nmFLMz6dmaiiq0+5v4XOKmNqf1J3GvBJYjTxAkH1VbCQ2y1fNxE8/cP5wyd3mU6M
LlD9tjc/Iro950gLlTq+9C2/QMis6ya52O2D8F652jnHnWbZrDf6zPdOSlFpCKkg05YSVpqifGgqjEh3RTyPs9w0Qc
NcxWcEqD1vvBMNW0VAL/YaW+MNmoUnpglR0cUeOJ3WFr8/1l/uyNyhfvAYfDUznVCNfVDtkMzCWhA4eH25XQUtmKwG
w==">
<create>
<serviceitem>
<name>AnandServiceItem</name>
<serviceItemData>
<serviceItemAttribute name="Name">Hannah</serviceItemAttribute>
<serviceItemAttribute name="DOB">1995-01-19</serviceItemAttribute>
<serviceItemAttribute name="Age">25</serviceItemAttribute>
</serviceItemData>
</serviceitem>
</create>
</message>
```

update

更新メッセージでは、サービス項目属性を省略すると、属性値は変更されません。メッセージで属性が明示的に指定されているものの、値が含まれない場合、そのサービス項目の属性値はテキスト フィールドでは空白に設定され、数値フィールドでは 0 に設定されます。

```
<message channel-id="A984F860-DBE7-48EB-B3A2-F6A0159B093F"
authorization-key="liDBm1NeRVbVmwbBmAd+cca0/Z8jf863aYOKl3QXSWcYwn5aGuPxn09CjDzv5EKR7/CcJx5
+2gu1MhUopAuiX3WwjX/DP1Xw3nmFLMz6dmaiiq0+5v4XOKmNqf1J3GvBJYjTxAkH1VbCQ2y1fNxE8/cP5wyd3mU6M
LlD9tjc/Iro950gLlTq+9C2/QMis6ya52O2D8F652jnHnWbZrDf6zPdOSlFpCKkg05YSVpqifGgqjEh3RTyPs9w0Qc
NcxWcEqD1vvBMNW0VAL/YaW+MNmoUnpglR0cUeOJ3WFr8/1l/uyNyhfvAYfDUznVCNfVDtkMzCWhA4eH25XQUtmKwG
w==">
 <update>
   <serviceitem>
   <name>AnandServiceItem</name>
```

```
   <serviceItemData>
   <serviceItemAttribute name="Name">Anand2update</serviceItemAttribute>
   <serviceItemAttribute name="RAM">Primarymemory3</serviceItemAttribute>
   <serviceItemAttribute name="MemoryInt">759001</serviceItemAttribute>
   <serviceItemAttribute name="Model512">Lenovo T440 updated</serviceItemAttribute>
   <serviceItemAttribute name="Money">80000</serviceItemAttribute>
   <serviceItemAttribute name="ManufTime">2016-05-01 16:45</serviceItemAttribute>
   </serviceItemData>
   </serviceitem>
  </update>
```

削除

delete サービス項目要求に必要なのは、サービス項目タイプおよびインスタンスの名前だけです。その他のサービス項目の属性およびサブスクリプション情報は無視されます。

```
<message channel-id="A984F860-DBE7-48EB-B3A2-F6A0159B093F"
authorization-key="liDBm1NeRVbVmwbBmAd+cca0/Z8jf863aYOKl3QXSWcYwn5aGuPxn09CjDzv5EKR7/CcJx5
+2gu1MhUopAuiX3WwjX/DP1Xw3nmFLMz6dmaiiq0+5v4XOKmNqf1J3GvBJYjTxAkH1VbCQ2y1fNxE8/cP5wyd3mU6M
LlD9tjc/Iro950gLlTq+9C2/QMis6ya52O2D8F652jnHnWbZrDf6zPdOSlFpCKkg05YSVpqifGgqjEh3RTyPs9w0Qc
NcxWcEqD1vvBMNW0VAL/YaW+MNmoUnpglR0cUeOJ3WFr8/1l/uyNyhfvAYfDUznVCNfVDtkMzCWhA4eH25XQUtmKwG
w==">
   <delete>
      <serviceitem>
      <name>ServiceItem</name>
      <serviceItemData>
      <serviceItemAttribute name="Name">Anand3</serviceItemAttribute>
      </serviceItemData>
   </serviceitem>
   </delete>
</message>
```

発信メッセージ

パブリッシュされるメッセージ形式は、Service Link で現在使用されている形式と同様で、nsXml、JSON、または FTL 形式です。

XML 発信メッセージの例

nsXml データ構造の例を以下に示します。

- データのメッセージ構造例、サービス詳細なし（デフォルト、小）

```
<?xml version="1.0" encoding="UTF-8"?>
<message channel-id="F203ACC7-E6CA-A2EA-E040-007F0101140E"
xmlns:fo="http://www.w3.org/1999/XSL/Format">
<task-started task-type="task">
<task>
<actual-duration>0.0</actual-duration>
<completed-date/>
<context-id>69</context-id>
<context-type>Requisition Entry</context-type>
<due-date>2014-04-25 20:00:00</due-date>
<effort>10.0</effort>
<estimated-date/>
<expected-duration>10.0</expected-duration>
<flag-id>0</flag-id>
<is-sharable>true</is-sharable>
<is-shared>true</is-shared>
<next-action-id>2</next-action-id>
```

```
<performer-actual-duration>0.0</performer-actual-duration>
<priority>2</priority>
<scheduled-start-date>2014-04-24 18:00:00</scheduled-start-date>
<start-date>2014-04-23 14:11:13</start-date>
<state-id>2</state-id>
<subject>queueServiceRequest1</subject>
<task-id>266</task-id>
</task>
<requisition>
<services>0</services>
<actual-cost>0.0</actual-cost>
<actual-duration>0.0</actual-duration>
<closed-on/>
<customer>
<company-address/>
<email>internal@newscale.com</email>
<fax/>
<first-name>admin</first-name>
<home-ou>
<name>&lt;s ID=&quot;847&quot;/></name>
<organizational-unit-id>1</organizational-unit-id>
</home-ou>
<home-phone/>
<last-name>admin</last-name>
<login-name>admin</login-name>
<person-id>1</person-id>
<personal-address/>
<supervisor-name/>
<timezone>Pacific Standard Time</timezone>
<work-phone/>
</customer>
<due-on>2014-04-25 20:00:00</due-on>
<expected-cost>0.0</expected-cost>
<expected-duration>0.0</expected-duration>
<external>false</external>
<initiator>
<company-address/>
<email>internal@newscale.com</email>
<fax/>
<first-name>admin</first-name>
<home-ou>
<name>&lt;s ID=&quot;847&quot;/></name>
<organizational-unit-id>1</organizational-unit-id>
</home-ou>
<home-phone/>
<last-name>admin</last-name>
<login-name>admin</login-name>
<person-id>1</person-id>
<personal-address/>
<supervisor-name/>
<timezone>Pacific Standard Time</timezone>
<work-phone/>
</initiator>
<organizational-unit>
<name>&lt;s ID=&quot;847&quot;/></name>
<organizational-unit-id>1</organizational-unit-id>
</organizational-unit>
<requisition-entry>
<closed-date/>
<data-values>
<data-value multi-valued="false">
<name>amqpDict1.field1</name>
<value>a</value>
</data-value>
```

```
<data-value multi-valued="false">
<name>amqpDict1.field2</name>
<value>b</value>
</data-value>
<data-value is-secure="true" multi-valued="false">
<name>amqpDict1.field3</name>
<value>ut3u4RC699wz7Jd2O+4cix23m9XXgAC/EsDiHp1ERsOpdKSKdrtgMjFWcVoO96aCFSkCgwa2tkBo
vKoOHzaillNiJ47+aY8zCWKwd17tCjmMLEkeQlTvrDvIHR/DT1iWSmN08DI9+Ns7hY3A/g6ijUoM
gcuMczH+5F/pGtgupLZF/L7FwOwu4VcKVWM/2N5tuXGz+1aHTRAAAAByQXr/yD/75Ysy57LnQLxc
</value>
</data-value>
<data-value is-secure="true" multi-valued="false">
<name>amqpDict1.field4</name>
<value>ut3u4RC699wz7Jd2O+4cix23m9XXgABReynDp5AdBgRi5ivhS05Unv98BgWgxc5YNcZrCihhhH/1
rzZqhjiIjAoRelIjADDb3IQP72armXsLRTvh0/fusd4jLdVIm4q1s+GaSTt6F3oqQeZ4RLhVUopo
p2zNAxmjaGj629C8gWREes3Z8EjDvXg5K1YVi90fXJV1jDoADhAAAABP1hct5TNV2d8R1cN/qDtK
</value>
</data-value>
</data-values>
<due-date>2014-04-25 20:00:00</due-date>
<item-number>1</item-number>
<price-per-unit>0.0</price-per-unit>
<priced>true</priced>
<quantity>1</quantity>
<rejected>false</rejected>
<rejected-date/>
<requisition-entry-id>69</requisition-entry-id>
<revision-number>105</revision-number>
<service>
<estimated-cost>0.0</estimated-cost>
<name>queueService1</name>
<parameters>
<default-duration>0.0</default-duration>
<priority>2</priority>
<start-date/>
<start-mode>0</start-mode>
</parameters>
<pricing-schema>0</pricing-schema>
<quantity>1</quantity>
<service-id>2</service-id>
<version>105</version>
<standard-duration>0.0</standard-duration>
</service>
<start-after/>
<start-date>2014-04-23 14:10:48</start-date>
<start-mode>0</start-mode>
<status>1</status>
</requisition-entry>
<requisition-id>64</requisition-id>
<started-on>2014-04-23 14:10:47</started-on>
<status>1</status>
</requisition>
<context>
<requisitionentryref itemnumber="1"/>
</context>
</task-started>
</message>
```

- セキュリティ ストリングを持つメッセージ構造例を次に示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<message channel-id="F203ACC7-E6CA-A2EA-E040-007F0101140E"
xmlns:fo="http://www.w3.org/1999/XSL/Format">
<task-started task-type="task">
```

```
<task>
<actual-duration>0.0</actual-duration>
<completed-date/>
<context-id>69</context-id>
<context-type>Requisition Entry</context-type>
<due-date>2014-04-25 20:00:00</due-date>
<effort>10.0</effort>
<estimated-date/>
<expected-duration>10.0</expected-duration>
<flag-id>0</flag-id>
<is-sharable>true</is-sharable>
<is-shared>true</is-shared>
<next-action-id>2</next-action-id>
<performer-actual-duration>0.0</performer-actual-duration>
<priority>2</priority>
<scheduled-start-date>2014-04-24 18:00:00</scheduled-start-date>
<start-date>2014-04-23 14:11:13</start-date>
<state-id>2</state-id>
<subject>queueServiceRequest1</subject>
<task-id>266</task-id>
</task>
<requisition>
<services>0</services>
<actual-cost>0.0</actual-cost>
<actual-duration>0.0</actual-duration>
<closed-on/>
<customer>
<company-address/>
<email>internal@newscale.com</email>
<fax/>
<first-name>admin</first-name>
<home-ou>
<name>&lt;s ID=&quot;847&quot;/></name>
<organizational-unit-id>1</organizational-unit-id>
</home-ou>
<home-phone/>
<last-name>admin</last-name>
<login-name>admin</login-name>
<person-id>1</person-id>
<personal-address/>
<supervisor-name/>
<timezone>Pacific Standard Time</timezone>
<work-phone/>
</customer>
<due-on>2014-04-25 20:00:00</due-on>
<expected-cost>0.0</expected-cost>
<expected-duration>0.0</expected-duration>
<external>false</external>
<initiator>
<company-address/>
<email>internal@newscale.com</email>
<fax/>
<first-name>admin</first-name>
<home-ou>
<name>&lt;s ID=&quot;847&quot;/></name>
<organizational-unit-id>1</organizational-unit-id>
</home-ou>
<home-phone/>
<last-name>admin</last-name>
<login-name>admin</login-name>
<person-id>1</person-id>
<personal-address/>
<supervisor-name/>
<timezone>Pacific Standard Time</timezone>
```

```
<work-phone/>
</initiator>
<organizational-unit>
<name>&lt;s ID=&quot;847&quot;/></name>
<organizational-unit-id>1</organizational-unit-id>
</organizational-unit>
<requisition-entry>
<closed-date/>
<data-values>
<data-value multi-valued="false">
<name>amqpDict1.field1</name>
<value>a</value>
</data-value>
<data-value multi-valued="false">
<name>amqpDict1.field2</name>
<value>b</value>
</data-value>
<data-value is-secure="true" multi-valued="false">
<name>amqpDict1.field3</name>
<value>ut3u4RC699wz7Jd2O+4cix23m9XXgAC/EsDiHp1ERsOpdKSKdrtgMjFWcVoO96aCFSkCgwa2tkBo
vKoOHzaillNiJ47+aY8zCWKwd17tCjmMLEkeQlTvrDvIHR/DT1iWSmN08DI9+Ns7hY3A/g6ijUoM
gcuMczH+5F/pGtgupLZF/L7FwOwu4VcKVWM/2N5tuXGz+1aHTRAAAAByQXr/yD/75Ysy57LnQLxc
</value>
</data-value>
<data-value is-secure="true" multi-valued="false">
<name>amqpDict1.field4</name>
<value>ut3u4RC699wz7Jd2O+4cix23m9XXgABReynDp5AdBgRi5ivhS05Unv98BgWgxc5YNcZrCihhhH/1
rzZqhjiIjAoRelIjADDb3IQP72armXsLRTvh0/fusd4jLdVIm4q1s+GaSTt6F3oqQeZ4RLhVUopo
p2zNAxmjaGj629C8gWREes3Z8EjDvXg5K1YVi90fXJV1jDoADhAAAABP1hct5TNV2d8R1cN/qDtK
</value>
</data-value>
</data-values>
<due-date>2014-04-25 20:00:00</due-date>
<item-number>1</item-number>
<price-per-unit>0.0</price-per-unit>
<priced>true</priced>
<quantity>1</quantity>
<rejected>false</rejected>
<rejected-date/>
<requisition-entry-id>69</requisition-entry-id>
<revision-number>105</revision-number>
<service>
<estimated-cost>0.0</estimated-cost>
<name>queueService1</name>
<parameters>
<default-duration>0.0</default-duration>
<priority>2</priority>
<start-date/>
<start-mode>0</start-mode>
</parameters>
<pricing-schema>0</pricing-schema>
<quantity>1</quantity>
<service-id>2</service-id>
<version>105</version>
<standard-duration>0.0</standard-duration>
</service>
<start-after/>
<start-date>2014-04-23 14:10:48</start-date>
<start-mode>0</start-mode>
<status>1</status>
</requisition-entry>
<requisition-id>64</requisition-id>
<started-on>2014-04-23 14:10:47</started-on>
<status>1</status>
```

```
</requisition>
<context>
<requisitionentryref itemnumber="1"/>
</context>
</task-started>
</message>
```

JSON 発信メッセージの例

```
{
  "message": {
    "taskStarted": {
      "task": {
        "actualDuration": 0.0,
        "calendarEntries": {},
        "checkLists": {},
        "completedDate": "",
        "contextId": 17,
        "contextType": "Requisition Entry",
        "dueDate": "2016-11-29 19:00:00",
        "effort": 10.0,
        "estimatedDate": "",
        "expectedDuration": 10.0,
        "flagId": 0,
        "isSharable": true,
        "isShared": true,
        "nextActionId": 2,
        "performer": {
          "companyAddress": "",
          "email": "",
          "fax": "",
          "firstName": "Default Service Delivery",
          "homeOu": {
            "name": "<s ID=\"847\"/>",
            "organizationalUnitId": 1
          },
          "homePhone": "",
          "lastName": "Queue",
          "loginName": "",
          "personId": 2,
          "personalAddress": "",
          "supervisorName": "",
          "timezone": "Pacific Standard Time",
          "workPhone": ""
        },
        "performerRole": {
          "name": ""
        },
        "performerActualDuration": 0.0,
        "priority": 2,
        "queue": {
          "companyAddress": "",
          "email": "",
          "fax": "",
          "firstName": "Default Service Delivery",
          "homeOu": {
            "name": "<s ID=\"847\"/>",
            "organizationalUnitId": 1
          },
          "homePhone": "",
          "lastName": "Queue",
```

```
      "loginName": "",
      "personId": 2,
      "personalAddress": "",
      "supervisorName": "",
      "timezone": "Pacific Standard Time",
      "workPhone": ""
    },
    "scheduledStartDate": "2016-11-28 17:00:00",
    "startDate": "2016-11-26 12:32:36",
    "stateId": 2,
    "subject": "amqpTask1",
    "supervisor": {
      "companyAddress": "",
      "email": "",
      "fax": "",
      "firstName": "Default Service Delivery",
      "homeOu": {
        "name": "<s ID=\"847\"/>",
        "organizationalUnitId": 1
      },
      "homePhone": "",
      "lastName": "Queue",
      "loginName": "",
      "personId": 2,
      "personalAddress": "",
      "supervisorName": "",
      "timezone": "Pacific Standard Time",
      "workPhone": ""
    },
    "supervisorRole": {
      "name": ""
    },
    "taskId": 67,
    "waiting": false
  },
  "requisition": {
    "services": 0,
    "actualCost": 0.0,
    "actualDuration": 0.0,
    "closedOn": "",
    "comments": {},
    "costCenterCode": "",
    "customer": {
      "companyAddress": "",
      "email": "internal@newscale.com",
      "fax": "",
      "firstName": "admin",
      "homeOu": {
        "name": "<s ID=\"847\"/>",
        "organizationalUnitId": 1
      },
      "homePhone": "",
      "lastName": "admin",
      "loginName": "admin",
      "personId": 1,
      "personalAddress": "",
      "supervisorName": "",
      "timezone": "Pacific Standard Time",
      "workPhone": ""
    },
    "dueOn": "2016-11-26 12:32:35",
    "expectedCost": 0.0,
    "expectedDuration": 0.0,
    "external": false,
```

```
"initiator": {
  "companyAddress": "",
  "email": "internal@newscale.com",
  "fax": "",
  "firstName": "admin",
  "homeOu": {
    "name": "<s ID=\"847\"/>",
    "organizationalUnitId": 1
  },
  "homePhone": "",
  "lastName": "admin",
  "loginName": "admin",
  "personId": 1,
  "personalAddress": "",
  "supervisorName": "",
  "timezone": "Pacific Standard Time",
  "workPhone": ""
},
"invocations": {},
"organizationalUnit": {
  "name": "<s ID=\"847\"/>",
  "organizationalUnitId": 1
},
"requisitionEntry": [
  {
    "closedDate": "",
    "dataValues": {
      "dataValue": [
        {
          "name": "amqpDict1.field1",
          "value": [
            "a"
          ],
          "multiValued": false
        },
        {
          "name": "amqpDict1.field2",
          "value": [
            "b"
          ],
          "multiValued": false
        }
      ]
    },
    "dueDate": "2016-11-29 19:00:00",
    "itemNumber": 1,
    "pricePerUnit": 0.0,
    "priced": true,
    "quantity": 1,
    "rejected": false,
    "rejectedDate": "",
    "rejector": {
      "companyAddress": "",
      "email": "",
      "fax": "",
      "firstName": "",
      "homeOu": {},
      "homePhone": "",
      "lastName": "",
      "loginName": "",
      "personId": 0,
      "personalAddress": "",
      "supervisorName": "",
      "timezone": "",
```

```
            "workPhone": ""
          },
          "requisitionEntryId": 17,
          "revisionNumber": 3,
          "service": {
            "estimatedCost": 0.0,
            "form": {
              "fields": {}
            },
            "name": "amqpService1",
            "parameters": {
              "defaultDuration": 0.0,
              "priority": 2,
              "startDate": "",
              "startMode": 0
            },
            "pricingSchema": 0,
            "quantity": 1,
            "serviceId": 142,
            "version": 3,
            "standardDuration": 0.0
          },
          "startAfter": "",
          "startDate": "2016-11-26 12:32:36",
          "startMode": "0",
          "status": 1,
          "bundle": false
        }
      ],
      "requisitionId": 17,
      "startedOn": "2016-11-26 12:32:35",
      "status": 1
    },
    "context": {
      "requisitionentryref": {
        "itemnumber": 1
      }
    },
    "taskType": "task"
  },
  "channelId": "42346EF5-D2A7-EC03-E050-11AC02003EE1"
 }
}
```

Service Link 標準アダプタによる統合の設計

概要

Service Link は、外部システムとの統合を実現するサービス カタログ（Service Catalog）モジュールです。他のシステムによってサービス カタログ（Service Catalog）ワークフロー内に定義された提供タスク、承認、または確認を実行できるようにするインターフェイスを設定するためのフレームワークと、それらのインターフェイスの動作をモニタリングするためのユーザ インターフェイスを提供します。

Service Link の使用に関する最も一般的なシナリオは、サービスが問題なく確実に提供されるように、提供計画タスクに関連するデータをサービス カタログ（Service Catalog）の外部に渡す必要がある状況です。たとえば Service Link がサービス要求を満たすために Cisco Process Orchestrator を呼び出すと、メッセージがハードウェア ベンダーに渡されて調達アクションが実行されたり、メッセージがインベントリ管理システムや資産管理システムに渡されてデータ レコードが更新されたりします。すると、外部アプリケーションがサービス カタログ（Service Catalog）に 1 つ以上のメッセージを返信します。続いて、各メッセージにより、外部システム内でサービス カタログ（Service Catalog）がタスクの現在のステータスでアップデートされ、最後に、タスクが完了したこと、およびサービス カタログ（Service Catalog）ワークフロー（提供計画）を後続のタスクに継続できることが示されます。

Service Link は多数の組み込みアダプタがあり、ファイルの交換、データベースの更新、HTTP POST 要求または Web サービスを介した Web 通信、およびキュー ベースのメッセージングなどのさまざまな転送メカニズムを使用して、外部アプリケーションとの通信に役立ちます。これらのデフォルトのアダプタに加えて、開発者は Service Link Adapter Development Kit（ADK）を使用してカスタム アダプタを開発し、導入できます。

Service Link 統合を開発するための前提条件

Service Link 統合の開発には、幅広い技術スキルが必要です。これには次が含まれます。

- Active Form Components（AFC）でのディクショナリ使用の設定方法や配信計画でのタスクの設計方法など、サービス設計に関する理解。
- アプリケーションをホストしているサーバを含め、ターゲットのサードパーティ システムに関する詳細な知識。
- すべてのアダプタについて、XML タグ構造に関する理解（Service Link は XML メッセージをサービス カタログ（Service Catalog）と外部システム間で送信することで動作するため）。

- すべてのアダプタの XML スタイルシート言語（XSL）変換の設定に関するある程度の理解（Service Link ウィザードを使用して適用できる XML 変換を補完するため）。
- データベース アダプタを使用する場合は、SQL に関する知識も必要です。
- http/Web サービス アダプタを使用して、サービス カタログ（Service Catalog）と Web サービス間でメッセージを渡す場合は、SOAP、WSDL、RESTful API、および Web サービス セキュリティのような Web サービス コンポーネントの知識が役立ちます。

Service Link の設計コンポーネント

サービス カタログ（Service Catalog）では、2 つの方法で統合を設計できます。

- Service Designer で使用できる Integration ウィザードは、Web サービスの統合を作成するためのウィザード形式のアプローチです。
- Service Link モジュールには、外部システムとの通信に使用されるメッセージング プロトコルに関係なく、すべての統合の作成および維持するための機能があります。

統合が追加されると、Service Link から使用できる高度な設定機能によって表示および維持が可能になります。上級ユーザは、場合によってはウィザードを使わずに、この機能を使用して Web サービスを統合することもできます。

また、管理者は Service Link を使用して、実稼働環境での統合の管理およびトラブルシューティングを実行します。

統合環境は、次のコンポーネントで構成されます。

- アダプタ

  アダプタは、サービス カタログ（Service Catalog）から他社製システムに XML ドキュメントまたはその他のメッセージを送信するためのトランスポート コンポーネントを論理的に表現する用語です。あらかじめパッケージ化されたアダプタは、ファイル、http/Web サービス、JMS、IBM MQ、データベースなど、さまざまなメッセージ トランスポート プロトコルをサポートしています。

  アダプタは、2 つのコンポーネントで構成されます。

  – 着信アダプタ

    受信アダプタは、外部システムからの着信メッセージを管理します。外部システム メッセージは、データを Service Catalog で解釈できるように、変換によって「標準」の nsXML（旧称 newScale XML）形式に変更できます。

    受信アダプタには、ポーラーとリスナーの 2 種類があります。ポーラーは定期的に起動して、着信メッセージを探すスレッドです。これに対し、リスナーは待機し、着信した外部メッセージによって起動されます。ポーラーの例としては、メッセージを定期的にチェックする必要がある受信ファイル アダプタがあります。リスナー アダプタの例としては、HTTP 応答が受信されるまで待機する Web サービス リスナー アダプタがあります。

  – 発信アダプタ

    送信アダプタは、サービス カタログ（Service Catalog）からの着信 XML メッセージを管理し、そのメッセージを設定済みの外部システムに送信します。「標準」の nsXML 発信メッセージが Service Link に送信されると、変換によってメッセージが変更され、外部システムに転送できるメッセージの形式になります。その後、送信アダプタは、正しいプロトコルおよびロジックを適用して、メッセージを外部システムに送信します。

- エージェント

  エージェントは、サービス カタログ（Service Catalog）が他社製システムとの間で双方向の通信を行うためのトランスポート メカニズムを論理的に表現する用語です。エージェントは、サービス設計者がタスクを適切な他社の宛先に送信するために使用できます。タスクに加えて、このアクションを外部システムに送信するエージェントを指定することにより、承認や確認を外部で行うことができるようになります。

  エージェントは、着信アダプタおよび発信アダプタ、オプションのメッセージ変換（XSLT）コンポーネント、オプション パラメータ、エラー状態を解決するためのその他の設定で構成されます。

- 変換

  XML スタイルシート（XSL）変換により、送信メッセージは他社製システムが理解できる形式に変換され、着信メッセージはサービス カタログ（Service Catalog）が理解できる形式に変換されます。

  発信アダプタが含まれるエージェントは、現在の要求およびタスクに関連する情報が含まれる nsXML メッセージを自動的に作成します。エージェントに関連付けられた変換では、メッセージを外部メッセージに変換でき、その外部メッセージはエージェントに対して設定された発信アダプタを介して外部システムに配信されます。同様に、受信エージェントは、関連付けられたアダプタを介して外部メッセージを受信します。変換では、サービス カタログ（Service Catalog）のワークフロー マネージャ Business Engine によって認識および処理される着信メッセージ タイプにメッセージを変換する必要があります。

- ディクショナリとアクティブ フォーム コンポーネント

  ディクショナリは、特定のサービス要求を処理するために必要なデータ フィールドを保持するサービス デザイン コンポーネントです。ディクショナリ フィールド（またはサービス要求で使用できるその他のデータ）にマッピングされたエージェント パラメータにより、外部システムが容易に理解できる標準の送信メッセージ形式になります。外部システムから受信した着信メッセージのエージェント パラメータは、Service Link に、これらのパラメータにマッピングされたディクショナリ フィールドの値を更新するように指示します。変更されたフォーム データは、サービス フォームですぐに使用できます。また、ディクショナリが含まれるアクティブ フォーム コンポーネントは、Service Link 統合を実装するサービスに含まれている必要があります。

  必要に応じ、newscale.properties ファイルの *serviceform.simtask.validation.skip* プロパティを true に設定すると、検証チェックをバイパスし、1 または複数のサービス項目ベースのディクショナリを含むアクティブ フォーム コンポーネントの削除を防ぐことができます。

[統合（Integration）] ウィザードを使用すると、エージェントと変換、およびエージェントの設定を完了するための統合ディクショナリとアクティブ フォーム コンポーネントが自動的に作成されます。これらのコンポーネントが作成されると、Service Link および Service Designer によって維持されるようになります。

Business Engine および nsXML と Service Link の相互作用

Service Link を理解するうえで重要なことは、Business Engine との相互動作を理解することです。Business Engine は、すべてのワークフローを処理するコンポーネントです。ワークフロー アクションには、次のものがあります。

- 正しい順序で配信計画のタスクを開始する。
- 完了のすべての要件を満たしている場合に、タスクを完了とマークする。
- イベントのトリガーが発生したときに、設定されたとおりに電子メールを送信する。

ServiceLink を使用しない（つまり、すべてのタスクがサービス カタログ（Service Catalog）の内部に存在する）タスク プランでは、BusinessEngine の動作はほとんど見えません。Business Engine の使用は Service Link 内部で認識されます。これは、外部タスクのステータスを変更するために、Business Engine が理解できるメッセージを Service Link が処理または生成する必要があるためです。

Business Engine は外部タスク（Service Link で処理されるタスク）を開始するときに、発信 nsXML メッセージを生成します。発信タスクを処理する Service Link エージェントは、ターゲット システムが理解できる形式に nsXML メッセージを変換し、エージェントで指定された発信転送メカニズム（アダプタ）を使ってターゲット システムに配信します。

同様に、Service Link が外部システムからの着信メッセージを受信するように設定されている場合、そのメッセージを Business Engine が理解できる着信 nsXML メッセージに変換する必要があります。受信メッセージを使用して、現在の要求に対するサービス フォーム データのアップデートや、現在のタスクの完了、現在の要求へのユーザ コメントの追加が行えます。

有効な nsXML メッセージについては、この後で詳細に説明します。

Service Link 統合の設計

Cisco Prime Service Catalog では、Service Link とサードパーティ システムとの統合を設計、開発、導入するための 2 つのアプローチがあります。

- [統合（Integration）] ウィザード：統合を Web サービスを使って行う場合、Service Designer から [統合（Integration）] ウィザードを使用して、すべての Service Catalog 統合コンポーネントを作成できます。この方法を使用するには、Web Services Description Language（wsdl）を使用できる必要があります。
- Service Link コンフィギュレーション：統合で他の転送メカニズムを使用する場合や、Integration ウィザードを介して作成されたコンポーネントの確認や変更を行う場合、Service Link および Service Designer によって表示される画面を使用して、統合コンポーネントを設定します。

Service Link および Service Designer コンフィギュレーションでは、次の方法を使用して、サードパーティ システムとの統合を設計、開発、および導入します。

- 他社製対象システムで使用する通信プロトコルを設計します。これには、受信アダプタと送信アダプタの選択、メッセージの形式と内容が含まれます。
- 必要に応じて、カスタム Service Link アダプタを作成して展開します。
- Service Designer を使用して、Service Link の統合を実装するサービスを設計します。デザイン コンポーネントには、通常、エージェント パラメータを通じて外部システムに渡されるデータを格納する 1 つ以上のディクショナリ、およびそれらのディクショナリを含み、ディクショナリ内のフィールドの表示プロパティを設定するアクティブ フォーム コンポーネントが含まれます。
- 適切な受信アダプタと送信アダプタを選択し、それぞれのプロパティおよびいずれかの方向に渡されるパラメータを定義することにより、エージェントを設定します。Service Link には、エージェント パラメータの定義を部分的に自動化するためのウィザードとドロップダウン リストが含まれています。
- 必要に応じて、他社製システムからのメッセージをサービス カタログ（Service Catalog）が理解し、サービス カタログ（Service Catalog）からのメッセージを他社製システム理解できるようにするための変換を追加します。

- Service Designer を使用して、サービスの配信計画内のタスクにエージェントを関連付けます。該当する場合は、そのサービスに使用されているディクショナリ フィールドにエージェント パラメータが適切にマッピングされていることを確認します。
- タスクが含まれるサービスを要求し、対応する Service Link ページからメッセージおよび外部タスクをモニタリングすることにより、設定をテストします。

このプロセスについては、この後の項で詳細に説明します。

Integration ウィザードについては、Service Designer での [統合（Integration）] ウィザードの使用で説明します。

Service Link へのアクセス

Service Link にアクセスするには、[モジュール（Module）] メニューから [Service Link] を選択します。Service Link ホームページが表示されます。

Service Link のホームページには、次の Service Link ホーム ページの要素に示す要素が含まれます。

表 5-1　Service Link ホーム ページの要素

要素	説明
メニュー バー	ページの上部にあり、Service Link の環境を管理し、Service Link の統合の開発および維持に使用されるタブが含まれています。
一般的なタスク	ページの左側にあり、エラー メッセージの一覧の表示およびエージェントの作成のためのクイック リンクがあります。
Service Link ステータス（Service Link Status）	ページの左側にあり、Service Link の現在のステータスを表示します。
Messages (Last 30 days)	ページの右ペインにあります。このグラフは、最近の 30 日間のメッセージの量を要約したものです。
最近失敗したメッセージ（Recent Failed Messages）	ページの右下にあり、正常に配信されなかった最新の Service Link メッセージが一覧表示されます。メッセージ、要求、およびエージェントの詳細へのハイパーリンクが含まれます。

通信プロトコルの設計

Service Link で実行される設定およびテスト作業に加えて、同等の作業が他社製システムを担当する技術リソースによって実行される必要があります。インターフェイスが堅牢に動作するためには、熟慮された設計が必須です。

通常は、通信相手となるシステムのインターフェイス機能によって、統合の基本設計、つまり使用されるアダプタが決まります。

ファイル アダプタやデータベース アダプタは最も簡単に設計できます。JMS、MQ、または http/Web サービス アダプタを展開した場合は、接続の問題がシステム間のデータの移動を妨げないように、ネットワーク管理チームの専門知識が盛り込まれることがあります。ターゲット システムが企業のネットワーク外に存在している場合は、データ セキュリティの懸念も要因となります。たとえば、外部ベンダーとの通信に http または https で送信される SOAP メッセージを使用する場合です。

通常、発信 Service Link 通信に必要なデータを最初に評価します。(nsXML 送信メッセージを通じて)どのフィールドがすぐに使用でき、(フォーム データおよびエージェント パラメータを通じて)どの追加データを提供する必要があるのかを考慮する必要があります。送信通信はタスクごと(タスクの開始時)にのみ発生しますが、複数の独立した着信通信をサポートできます。作業の実行命令を受信すると、他社製システムは、完了時に単一の(着信)通信を発行する可能性があります。また、作業が完了する前に、複数のアップデートが送信されることも考えられます。たとえば、参照 ID を伝達する場合には、テキストのステータス アップデート返信する必要があり、最後に完了確認が伝達されます。Service Link アダプタの管理

統合の管理

ここでは、統合の管理について説明します。

アダプタの管理

Service Link のアダプタは、事前構成されます。追加のアダプタは、Adapter Development Kit を使用して開発できます。Service Link の [統合の管理(Manage Integrations)] タブの [アダプタ(Adapters)] ページを使用して、使用可能なアダプタを確認できます。

手順 1　Service Link ホームページで [統合の管理(Manage Integrations)] をクリックします。次に、[アダプタ(Adapters)] サブタブをクリックします。

[アダプタ(Adapters)] ページが表示されます。

手順 2　[名前(Name)] カラムで、目的のアダプタをクリックします。

選択したアダプタの [アダプタの管理(Manage Adapter)] ページが表示されます。データベース アダプタの詳細は次のとおりです。

図 5-1　[アダプタの管理(Manage Adapter)] ページ

これらの一般的なプロパティの大半は、通常、Service Link 開発者が変更すべきではありません。[Polling Interval]、[Retry interval]、および [Maximum Attempts] は、要件に応じて変更する必要があります。すべての変更は、このアダプタ タイプを使用するすべてのエージェントによって継承されます。

追加の受信プロパティおよび送信プロパティは、アダプタをエージェントで使用する場合に指定します。これらのプロパティについては、Service Link アダプタの管理で説明します。

エージェントの管理

エージェント ウィザードを使用すると、エージェントの設定手順を、順を追って実行できます。このウィザードは 8 ページで構成されますが、前のページで選択したオプションによって一部のページがスキップされる場合があります。

エージェント ウィザードのページの概要は、次のエージェント ウィザード ページの表のとおりです。

表 5-2　エージェント ウィザード ページの表

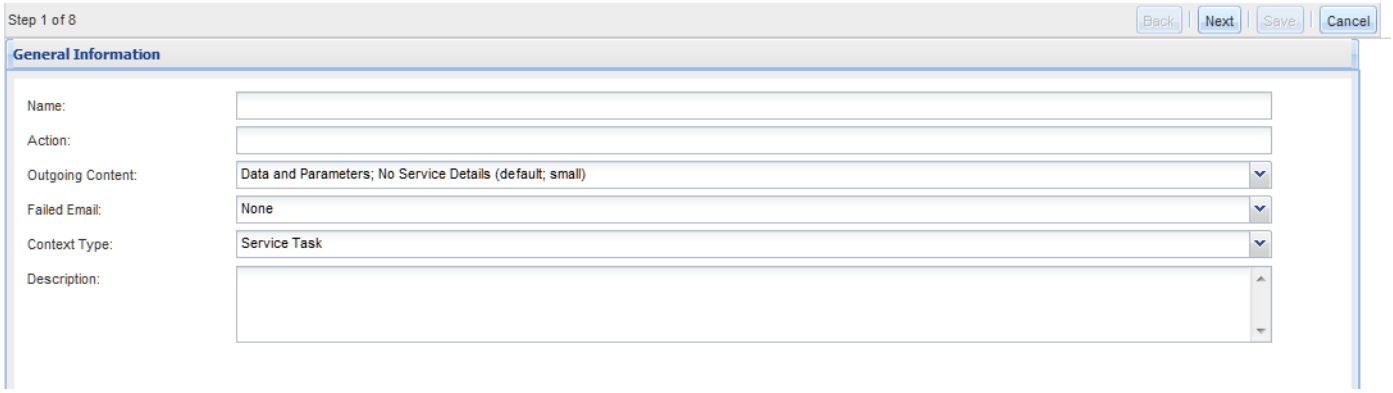
ページ	名前/説明
全般情報（General Information）	エージェントの名前、アクション、説明、およびエージェントの設定と動作に関するその他の一般情報
End Points	エージェントが使用するアダプタおよび変換
送信プロパティ	送信アダプタの詳細設定オプション
受信プロパティ	受信アダプタの詳細設定オプション
送信要求パラメータ	VMware アダプタを除くすべてのアダプタ タイプの送信パラメータ
送信応答パラメータ	http/ws アダプタに送信された送信メッセージに対する同期応答で受信されるパラメータ
Inbound Parameters	受信メッセージの一部として受信されるパラメータ
要約	これまでに入力したすべての情報を表示する要約ページ

エージェントを管理するには:

手順 1　Service Link ホームページの、[共通タスク（Common Tasks）] 領域で [エージェント（Agent）] をクリックするか、[統合の管理（Manage Integrations）] > [エージェント（Agents）] > [エージェント（Agent）] を選択します。

エージェント ウィザードの [一般情報（General Information）] ページが表示されます。これは、このウィザードを構成する 8 ページのうちの最初のページです。特定のエージェント設定には関係のないページは、スキップできます。

図 5-2　[一般情報(General Information)] ページ



次のエージェントの作成:一般属性の表で説明するフィールドに値を入力し、[Next] をクリックします。

表 5-3　エージェントの作成:一般属性の表

設定	説明
名前(Name)	エージェントの名前。名前は一意であることが必要です。
操作(Action)	「サービス カタログ(Service Catalog)解決策のチケット」など、エージェントによって実行されるアクションの説明。アクションに一意性は求められませんが、タスクへのエージェントの割り当てを求められたときに選択リストに表示されるため、一意になるようにしてください。
Outgoing Content	このエージェントに対して生成される nsXML 送信メッセージのノードを指定するためのオプション。詳細については、後述します。
失敗した電子メール(Failed Email)	送信メッセージを宛先に配信に送信できない場合に送信される電子メール テンプレート。電子メール テンプレートは、Administration モジュールの [通知(Notifications)] オプションを通じて定義する必要があります。
Context Type	エージェントのタイプ。提供計画内でエージェントを外部タスクとして使用できるようにする Service Link エージェントは「サービスタスク(Service Tasks)」です。「サービス項目(Service Item)」エージェントは、Service Item Manager を通じて設定した定義やデータ、またはサービス項目をインポートする目的で使用します。
説明	エージェントの詳細な説明。この詳細説明は、統合をサポートするうえで役立ちます。

失敗した電子メール(Failed Email)

Failed Email による通知は、送信メッセージを配信できない場合に生成できます。Service Link の失敗した電子メールには、すべてのタスク関連電子メールに使用できる標準の名前空間セットに加えて、障害時に生成されるメッセージに関する詳細を含めることができます。これらの名前空間を電子メール テンプレートに含めると、問題の診断時に役立てられます。使用可能な名前空間の詳細については、『*Cisco Prime Service Catalog Designer Guide*』を参照してください。

Failed Email は着信メッセージに適用されません。Service Link が着信メッセージの処理に失敗した場合、通知は生成されません。

送信メッセージの内容

発信アダプタは、サービス カタログ（Service Catalog）データベースに保存される nsXML メッセージを生成します。その後、このメッセージは変換され、指定したアダプタを介して、結果としての外部メッセージが任意の宛先に配信されます。

メッセージのフォーマットは ISEE.war/WEB-INF/classes/nsxml.xsd のアプリケーション サーバ上の対応する XML スキーマごとに、Adapter Development Kit による統合の設計に記載されています。完了メッセージには、サービス要求に関するすべての情報が含まれます。このようなメッセージはサイズが非常に大きくなる可能性があり（サービスで使用されるディクショナリやフィールドの数に応じて、500 K を超えることがよくあります）、結果的にデータベース内で大量のストレージを消費し、生成するために CPU の使用率も非常に高くなります。

リソースの消費量を削減するため、サービス カタログ（Service Catalog）には次のオプションが用意されています。

- [管理（Administration）] 設定を [メッセージの圧縮（Compress messages）] に設定すると、データベースに格納されている Service Link メッセージを圧縮できます。これにより、ストレージ要件が大幅に軽減されますが、DBA または担当者がメッセージを判読できなくなるため、トラブルシューティングが複雑になる可能性があります。
- Service Link Message Purge スクリプトが使用可能です。このスクリプトにより、完了済みのタスクのメッセージをトランザクション データベースから削除できます。その結果、データベース内のメッセージのストレージ要件が軽減されます。Message Purge スクリプトの使用に関する詳細については、Service Link のトラブルシューティングと管理を参照してください。
- 「Outgoing message content」プロパティを操作すると、nsXML メッセージの選択したノードだけを含めるように発信内容を設定できます。これにより、送信メッセージを処理するためのメモリ要件および CPU 要件が軽減されます。

デフォルトのメッセージ内容は「Data and parameters; no Service Details (small)」で、要求されたサービスを説明するコンテンツ ノードが含まれない nsXML メッセージが生成されます。エージェント パラメータおよび変換は、発信内容のタイプを考慮して設計する必要があります。要求されるすべての内容が nsXML メッセージに含まれていることを確認してください。特に、送信メッセージからディクショナリ データを排除する場合は、エージェント パラメータを適切なフォーム フィールド（または定数）にマッピングする必要があります。外部タスクに必要のない多数のフォーム フィールドがサービスに含まれる場合は、XML サイズの削減および関連する CPU 使用率の削減が非常に重要です。

発信内容のオプションの概要については、次の発信メッセージの内容：オプションの表を参照してください。

表 5-4　発信メッセージの内容：オプションの表

オプション	説明
All Message Details (large)	Service Link メッセージ全体が生成されます。
Data, Form and Parameters (medium-large)	サービスおよびそのタスクに関する情報は省略されます。メッセージはデータ（サービス フォームのフィールド値）、フォーム（サービス フォームのディクショナリおよびフィールドについての完全なメタデータ）、およびパラメータ（エージェント パラメータで指定される値）に制限されます。
Data and Parameters (medium)	メッセージは、要求に関する情報、サービス フォーム上で入力されたすべてのデータ値、およびパラメータ値に制限されます。

表 5-4　発信メッセージの内容:オプションの表(続き)

オプション	説明
Data and Parameters; No Service Details (small)	メッセージは、要求に関する情報、サービス フォーム上で入力されたすべてのデータ値、およびパラメータ値に制限されます(デフォルト設定)。VMware アダプタに対しては、「small」オプションを指定する必要があります。
Only Parameters (minimal)	メッセージは、要求に関する情報およびエージェント パラメータに制限されます。

アダプタの選択

エージェントは、送信通信と着信通信の両方、送信通信のみ、または着信通信のみを管理するように設定できます。方向ごとに異なるアダプタ タイプを使用できます。たとえば、データベース アダプタは送信データを書き込むために使用できますが、システムは、受信ファイル アダプタによって読み取られるディレクトリにファイルを書き込むことによって応答します。

アダプタ タイプを選択すると、ウィザードの後続のページは、アダプタ タイプおよび使用方法(送信または受信)に関するプロパティを表示するように調整されます。プロパティ値は、エージェント定義の一部として指定する必要があります。

エージェント ウィザードの [エンドポイント(End Points)] ページ(ページ 2/8)では、エージェントで使用するアダプタ、および送信メッセージまたは着信メッセージに適用する変換を指定できます。変換は、[統合の管理(Manage Integrations)] オプションの [変換(Transformation)] サブタブを使用して、事前に定義しておく必要があります。

図 5-3　エージェント ウィザードの [エンドポイント(End Points)] ページ

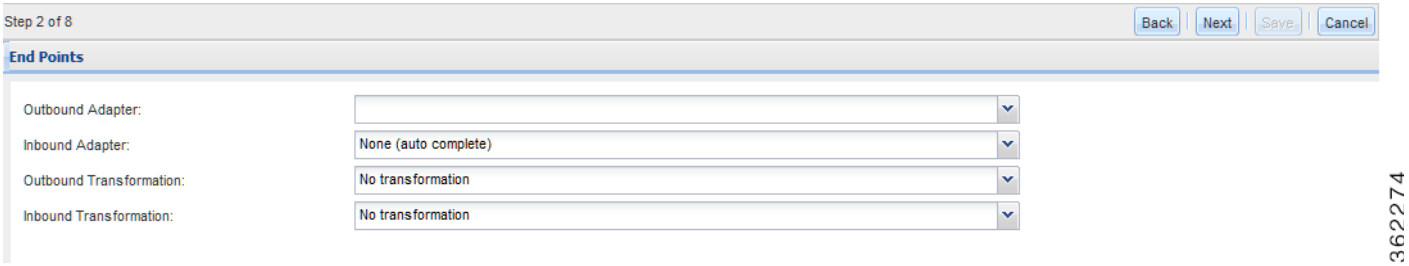


表 5-5　エージェントの作成:エンドポイント プロパティの表

設定	説明
Outbound Adapter	サービス カタログ(Service Catalog)から外部システムにメッセージを送信するために使用するデフォルト アダプタまたはカスタム アダプタ。
Outbound Transformation	外部システムが理解できる外部メッセージを生成するために、送信アダプタによって生成された nsXML メッセージに適用される変換(または none)。
Inbound Adapter	外部システムからメッセージを受信するために使用するアダプタ、または [自動入力(Auto Complete)](受信メッセージが予期されない場合)。
Inbound Transformation	Business Engine が理解できる nsXML メッセージを生成するために、着信メッセージに適用される変換(または none)。サービス タスクにのみ適用可能。

各タイプのアダプタに適用できるプロパティについては、この章で後述します。実際には、このウィザードの次に続く 2 ページ（送信アダプタおよび受信アダプタの構成専用）は、選択しているアダプタによって異なります。なお、2 つの特殊なケース、ダミー アダプタ（Dummy adapter）と自動完了（Auto Complete）のオプションについて説明します。

ダミー アダプタは、エージェント内に、送信アダプタまたは受信アダプタとして設定できます。ダミー アダプタを選択することは、エージェントが単一方向にのみ動作することを意味します。たとえば、ダミー受信アダプタを使用するように設定されているエージェントは、エージェントが送信通信だけに関与することを意味します。同様に、アップデート タスクおよび締め切りタスク専用の個別（受信のみ）エージェントが存在することも考えられます。

[Auto Complete] オプションは、エージェントの受信アダプタ部の選択肢としてのみ使用できます。その効果は「ダミー アダプタ」を選択することと同様で、エージェントは発信の通信だけを管理します。主な違いは、タスクに関連付けられた送信通信が送信されると、そのタスクは自動的に完了しますが、残りの提供計画は引き続き実行されることです。

エージェント パラメータの使用

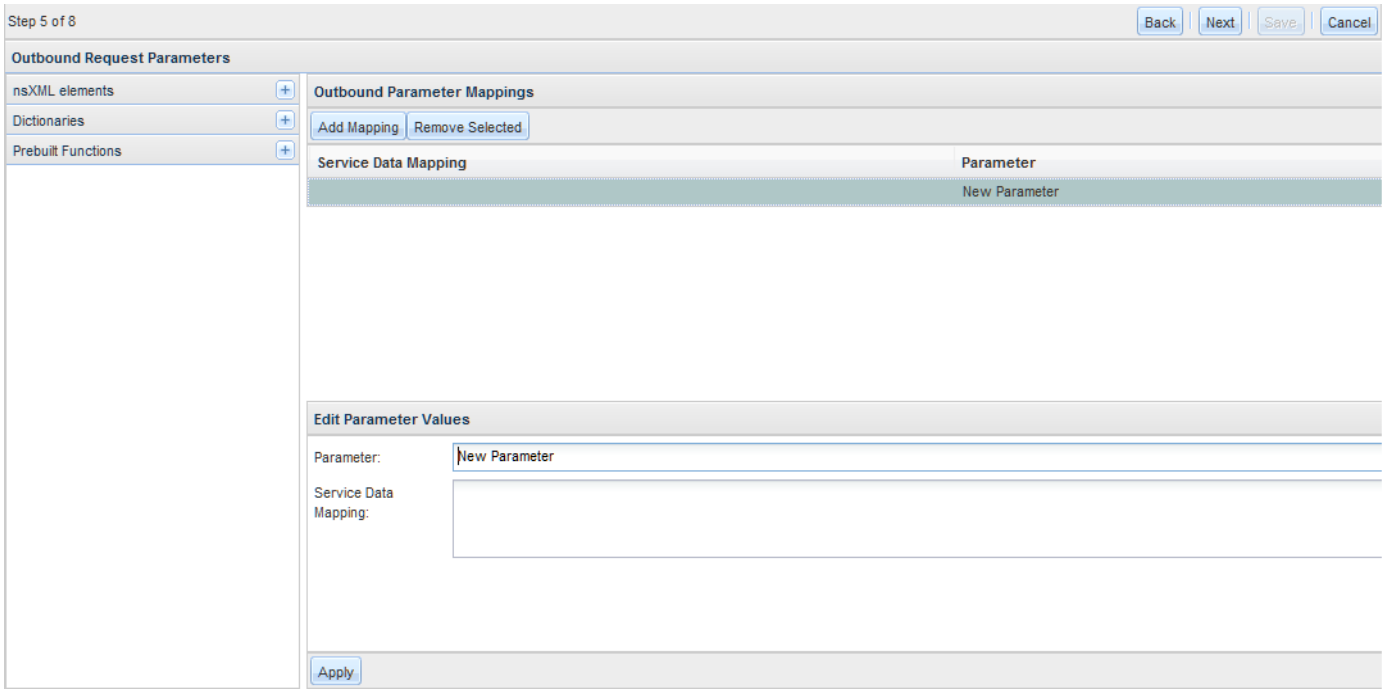
エージェント パラメータは、送信アダプタおよび受信アダプタと組み合わせて使用できます。

エージェント定義の一部として指定したパラメータ マッピングには、デフォルト値が使用されます。このマッピングは、Service Designer でサービスのタスク定義を編集することにより、サービスごとに上書きできます。

送信要求パラメータ

送信メッセージ内で使用されるエージェント パラメータは、追加データを含む標準 nsXML 送信メッセージを補足し、コンテンツ ノードを対応しやすい形式に整理するための方法を提供します。このパラメータには、外部メッセージに含めることができるように、XSL 変換を通じて簡単にアクセスできます。

図 5-4　送信要求パラメータ



パラメータ マッピングは、[サービスデータマッピング（Service Data Mapping）] 領域にマッピング元の要素を入力するか、パラメータ マッピング ペインの左にあるドロップダウン リストに表示されている要素を使用して式をビルドすることによって割り当てられます。マッピングは、次の組み合わせで構成できます。

- 定数値
- サービス フォーム上のディクショナリ フィールド
- nsXML 要素
- 上記のいずれかの要素に適用される事前作成済みの関数

エージェント パラメータをマッピングすると、次の利点があります。

- パラメータ値を抽出する変換では、値が入力されるディクショナリ フィールド名を参照する必要はありませんが、単にエージェント パラメータ名を参照することはできます。これにより、異なるサービスおよびディクショナリにわたるエージェントの再使用が促進されます。
- パラメータはすべてのコンテンツ タイプに含まれるため、メッセージで必要とされる他のすべてのコンテンツがサポートされる場合、サイズの小さい発信メッセージ コンテンツ タイプを使用できます。
- 変換を使用せずにはアクセスできない nsXML 要素および XPATH 操作を、外部メッセージに含めることができます。

送信パラメータを作成するには、次の手順を実行します。

手順 1　[送信要求パラメータ（Outbound Request Parameters）] ページで、[パラメータの追加（Add Mapping）] をクリックします。

ページの下部に [パラメータ値の編集（Edit Parameter Values）] ダイアログ ボックスが表示されます。

手順 2　パラメータの名前を入力します。

パラメータ名にはスペースを使用できますが、XML メッセージで意味を持つ特殊文字（「>」や「&」など）は使用できません。

手順 3　以下に示すガイドラインを使用して、パラメータの値/マッピングを指定します。

定数値

場合によっては、要求またはサービスの詳細に依存しない定数値を外部システムに渡す必要があります。たとえば、外部システムのマッピング元の名前を指定する必要がある場合は、[サービスデータマッピング（Service Data Mapping）] に「サービス カタログ（Service Catalog）」とだけ入力します（かぎカッコは入力しません）。

送信 nsXML へのマッピング

標準の nsXML 発信メッセージの選択された要素を、エージェント パラメータへのマッピングに使用できます。次の表は、これをまとめたものです。

表 5-6　送信 nsXML マッピング

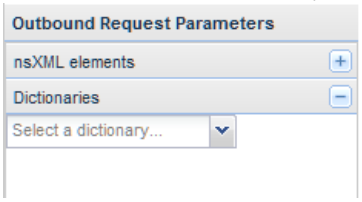
nsXML 要素	内容/説明
Customer	要求に対するカスタマーのログイン名
Initiator	要求の要求者（イニシエータ）のログイン名
requisition-entry-id	要求内のサービスを一意に識別する番号（通常は、ショッピング カート内の複数のサービスを区別するために使用されます）
expected-cost	サービスのトランザクション価格
expected-duration	サービスの標準期間
requisition-id	要求を一意に識別する番号（My Services および Service Manager 内で参照されます）
channel-id	外部タスクを識別するグローバルな固有識別子（GUID）

[辞書（Dictionary）] フィールドへのエージェント パラメータのマッピング

エージェント パラメータをディクショナリ フィールドにマッピングするには、次の手順を実行します。

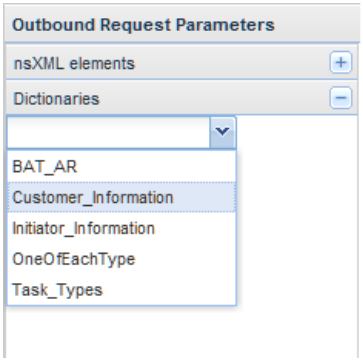
手順 1　[辞書（Dictionaries）] ノードを展開し、[辞書を選択（Select a dictionary）] オプションを表示します。

図 5-5　[辞書（Dictionary）] フィールドへのエージェント パラメータのマッピング



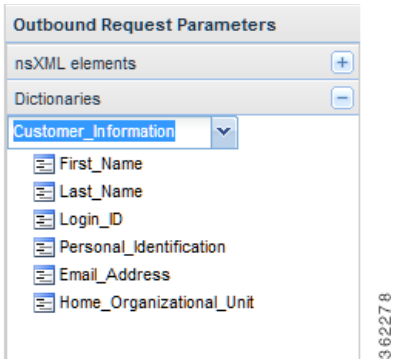
手順 2　すべての Service Catalog ディクショナリのリストを表示するには、[辞書を選択（Select a dictionary）] ドロップダウン メニューをクリックします。

図 5-6　ディクショナリ ドロップダウンの選択



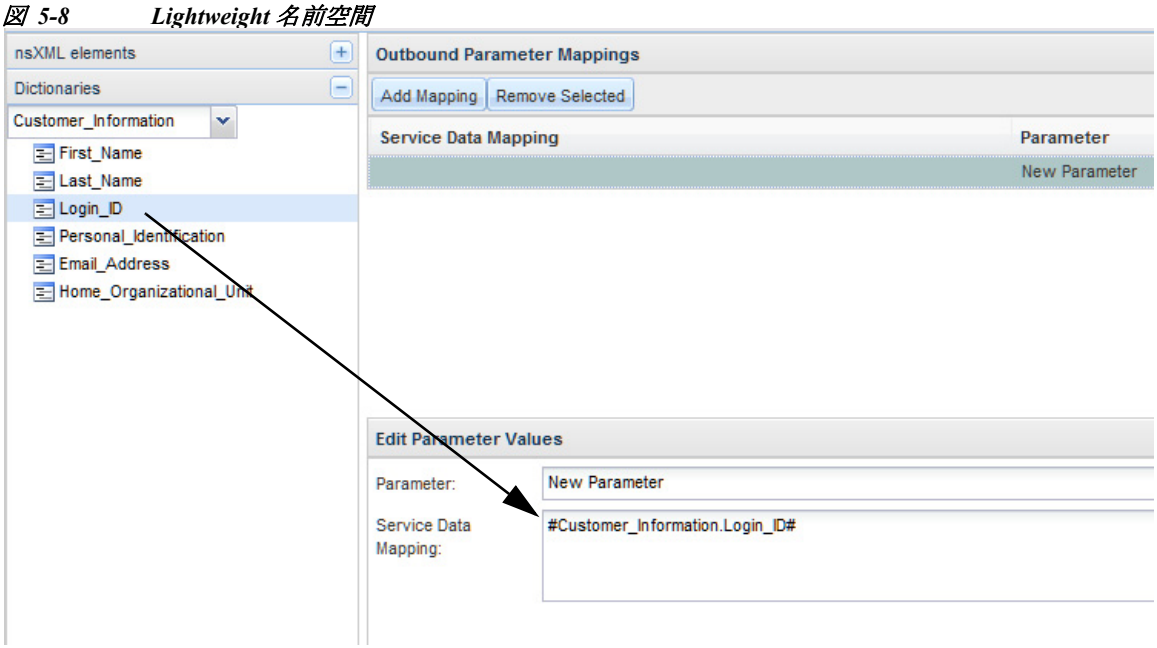
手順 3　エージェント パラメータにマッピングするフィールドが含まれるディクショナリを選択します。ディクショナリ内のすべてのフィールドのリストが表示されます。

図 5-7　ディクショナリのフィールド



手順 4　エージェント パラメータにマッピングするフィールドをクリックし、[サービスデータマッピング（Service Data Mapping）] テキスト領域にドラッグします。ドラッグ アイコンが緑色のチェック マークに変化したら、マウスを放します。選択したフィールドの lightweight 名前空間が表示されます。

図 5-8　Lightweight 名前空間



グリッド ディクショナリを除き、エージェントはサービスに関係なく定義されるため、任意のディクショナリ フィールドを選択できます。このエージェントが使用されているサービスに、実際に、参照先のディクショナリが含まれることを確認するのは、サービス設計者の役割です。

1 つ以上のグリッド ディクショナリのコンテンツを渡す統合の場合、各グリッド ディクショナリの行を扱うために変換が必要です。これは、それらの行が複数のディクショナリ インスタンスとして保存されているからです。この変換は、発信 nsXML の <data-values> セクションで、「DictionaryName-n」の命名規則に従います。n はグリッド行番号です。たとえば、VMOperation という名前のグリッド ディクショナリで、サービス要求に 2 行のデータがある場合、その値は次のように表現されます。

```
<data-values>
   <data-value multi-valued="true">
      <name>VMOperation-1.Name</name>
```

```
      <value>vmgw01</value>
   </data-value>
   <data-value multi-valued="false">
      <name>VMOperation-1.GuestOS_Name</name>
      <value>winNetStandardGuest</value>
   </data-value>
   <data-value multi-valued="false">
      <name>VMOperation-1.CPUCount</name>
      <value>1</value>
   </data-value>
   <data-value multi-valued="false">
      <name>VMOperation-1.Memory</name>
      <value>2048 MB</value>
   </data-value>
   <data-value multi-valued="true">
      <name>VMOperation-2.Name</name>
      <value>vmgw01</value>
   </data-value>
   <data-value multi-valued="false">
      <name>VMOperation-2.GuestOS_Name</name>
      <value>winNetStandardGuest</value>
   </data-value>
   <data-value multi-valued="false">
      <name>VMOperation-2.CPUCount</name>
      <value>1</value>
   </data-value>
   <data-value multi-valued="false">
      <name>VMOperation-2.Memory</name>
      <value>2048 MB</value>
   </data-value>
</data-values>
```

着信エージェント パラメータ マッピングおよびグリッド ディクショナリ フィールドの更新はサポートされません。

ビルド済み関数の適用

ビルド済み関数はマッピングされた要素に適用できるため、パラメータ値は、対象システムで予期された意味または書式設定要件に適合します。たとえば、対象システム内のフィールドのデータ定義に含まれる文字数がサービス カタログ（Service Catalog）に保持されている文字数よりも少ない場合は、substring 関数を適用することによってフィールドを短縮できます。ビルド済み関数の概要は以下のとおりです。詳細については事前作成済み関数で説明します。

表 5-7　ビルド済み関数

機能	名前/説明
trim	値から先頭と末尾のスペースを削除します。特に、値が CDATA タグで囲まれる、データベース アダプタのフォーム データや着信メッセージに役立ちます。
replace	1 文字またはパターンを別の文字またはパターンと置換します。
substring	開始点とオプションの長さで指定して、文字列の一部を選択します。
lowerCase	値をすべて小文字に変換します。
upperCase	値をすべて大文字に変換します。
length	文字列の文字数を返します。

表 5-7　ビルド済み関数（続き）

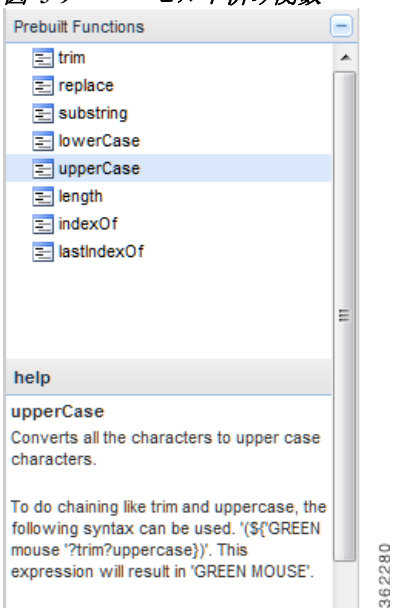
機能	名前/説明
indexOf	この文字列内の、指定された部分文字列の最初のインデックスを返します。
lastIndexOf	この文字列内の、指定された部分文字列の最後のインデックスを返します。

ビルド済み関数をエージェント パラメータに適用するには、次の手順を実行します。

手順 1　関数名が表示されるように、ビルド済み関数を展開します。

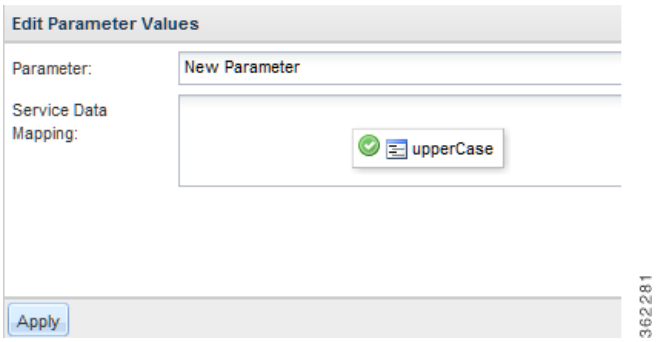
手順 2　使用する関数を強調表示します。ペインの下部に、関数の使用法を説明する [Help] が表示されます。

図 5-9　ビルド済み関数



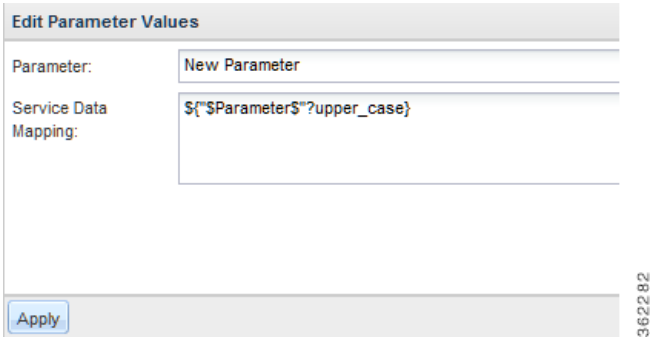
手順 3　この関数を、パラメータの [Service Data Mapping] ボックスにドラッグします。ドラッグ アイコンが緑色のチェック マークに変化したら、マウスを放します。

図 5-10　緑色のチェック マーク



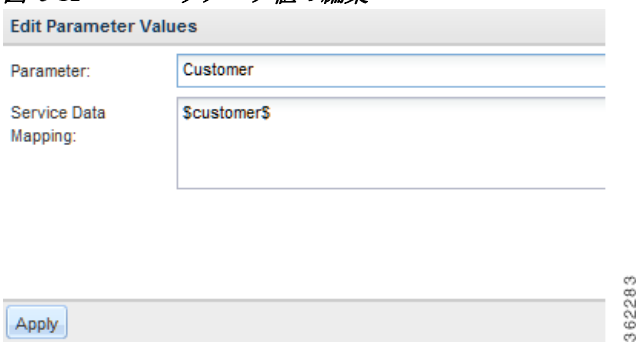
手順 4　$Parameter$ を実際の値のプレースホルダとして、関数が定義されます。

図 5-11　定義済み関数



手順 5　プレースホルダを、使用する nsXML メッセージのディクショナリ フィールドまたは要素と置換します。このフィールドまたは nsXML 要素を [Service Data Mapping] テキスト ボックスにドラッグし、パラメータ定義を手動で編集する必要があります。

図 5-12　パラメータ値の編集



送信 nsXML へのエージェント パラメータの追加

エージェント パラメータは、送信 nsXML メッセージの末尾に追加されます。たとえば、以下に示すエージェント パラメータは、直後に nsXML スニペットを生成します。

図 5-13　sXML スニペット

Service Data Mapping	Parameter
$initiator$	Initiator
$expectedCost$	Transactional Price
$customer$	Customer

362284

```
<agent-parameter multi-valued="false">
    <name>Initiator</name><value>ltierstein</value></agent-parameter>
<agent-parameter multi-valued="false">
    <name>Transactional Price</name><value>0.0</value></agent-parameter>
<agent-parameter multi-valued="false">
    <name>Customer</name><value>Customer</value></agent-parameter>
```

送信応答パラメータ

送信応答パラメータは、送信　http/Web サービスアダプタと組み合わせて使用できます。アダプタの [プロセス応答（Process Response）] 設定が true の場合、元の要求への応答が処理されます。この応答には、「送信パラメータ」メッセージ タイプが含まれる可能性があります。パラメータは、受信エージェント パラメータとして定義します。

受信エージェント パラメータ

エージェント パラメータを「送信パラメータ」受信メッセージ タイプと組み合わせて使用すると、そのパラメータがマッピングされているディクショナリ フィールドを外部タスクでアップデートできます。

表 5-8　受信エージェント パラメータの設定

設定	説明
パラメータ	パラメータ名。
Dictionary Field	すべてのディクショナリ フィールドが表示されているドロップダウン リストから、ディクショナリ フィールドを選択します。フィールド名は次の形式になります。 DictionaryName.FieldName ハッシュ記号（#）では囲みません。
マッピング	指定したディクショナリ フィールドのアップデートに使用される値を導出するために適用するビルド済み関数。
必須	フィールドを受信メッセージに含める場合は、[Mandatory] チェックボックスをオンにします。サービスを変更したためにパラメータが不要になった場合には、通常、このチェック ボックスをオフにします。不要になったパラメータを削除しないでください。

変換の管理

変換を管理するには:

手順 1　[統合の管理（Manage Integrations）] タブで、[変換（Transformations）] サブタブをクリックします。

手順 2　[変換（Transformation）] をクリックします。

[変換（Transformation）] ページが表示されます。

[Direction] を指定することにより、変換を送信/受信メッセージのいずれかに適用できます。[Process Response] 設定がオンになっている場合、Web サービス発信メッセージに 2 つの変換の使用が必要になる場合があります。1 つは発信要求に適用され、もう 1 つはその要求への応答に適用されます。

[Validate] ボタンをクリックすると、XSLT を解析して、変換が整形式であることが確認されます。適切でない（たとえば、XML タグのつづりの誤りや欠落がある）場合は、診断メッセージが表示されます。変換を保存する前に、エラーを修正する必要があります。

ただし、Service Link では、変換の検証は行われません。たとえば、ソース メッセージに存在しない要素を変換が参照していても、エラーは検出されません。その場合、整形式 XML メッセージが生成されますが、対象システムに対して有効ではありません。したがって、Service Link がターゲット アプリケーションで認識できない外部メッセージ、または Business Engine で認識できない着信メッセージを生成した場合は、ランタイム エラーが生成されます。

XML 開発環境にアクセスしている場合、使用方法に精通していれば、その環境を使用して変換のテストを行うと効率的です。送信変換の場合は、(変換を適用する前に)エージェントによって作成された nsXML を XML 開発環境にコピーし、それをソース XML として使用するだけです。変換を検証した後、XSLT コードを Service Link 変換にコピーして貼り付け、適切なエージェントに関連付けます。

または、特に既存の変換に小さな変更を加える場合には、このページのテスト関数を使用すれば、変換の出力 XML を素早くプレビューできます。[テスト(Test)] ボタンをクリックするとポップアップ ウィンドウが表示されるので、そこで nsXML をソース XML パネルに貼り付け、変換を実行して出力 XML を取得できます。

Integration ウィザードを使用して開発された送信 Web サービス統合の場合、手動での変換の開発およびデバッグ プロセスは排除されます。Web サービス用の変換が自動的に作成され、これにより、発信 nsXML が、指定された WSDL と互換性のある形式に変換されます。ただし、wsdl または統合要件が変更された場合は、前述の手順に従って変換をアップデートする必要があります。

手順 3　次の変換設定の表で説明するフィールドに値を入力します。

表 5-9　変換設定の表

設定	説明
名前	変換の名前。変換名には、インターフェイスの性質を記述します。たとえば、「サービス カタログ(Service Catalog)To Remedy」のように、ソース システムや対象システムの名前を含めることができます。
方向(Direction)	[インバウンド(Inbound)] または [アウトバウンド(Outbound)]。
説明	変換の説明(必須)。
フォーマット(Format)	サポートされている変換フォーマットは、XST、JOLT、および FTL です。
Created By	変換の作成者を入力します。
作成日(Created On)	変換の作成日を入力します。
更新者(Updated By)	変換の更新者を入力します。
最終更新日	変換が最後に更新された日付を入力します。
[Request] サブタブ	送信アダプタの nsXML メッセージおよび受信アダプタの外部メッセージに適用される XSLT コード
[Response] サブタブ	エージェントの [プロセス応答(Process Response)] 設定が true に設定されている場合に、http/Web service 要求から受信した応答に適用される XSLT コード。

手順 4　[Validate] をクリックして、変換に整形式 XSL が含まれることを確認します。

手順 5　[保存(Save)] をクリックします。

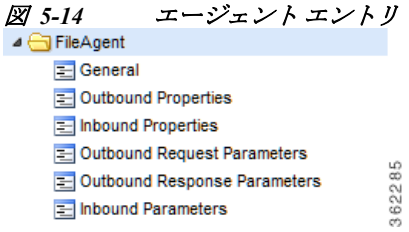
エージェント定義とプロパティ シートの確認

エージェントが Service Link または Integration ウィザードのどちらで作成された場合でも、すべてのエージェントの定義を確認または変更できます。[統合の管理（Manage Integrations）] タブの [エージェント（Agents）] サブタブが表示されたら、次のいずれかを実行できます。

- ページの左側にあるリスト ペインで、エージェント名をクリックします。
- ページの右側に一覧表示されたエージェント情報をスクロールし、エージェント名をクリックします。

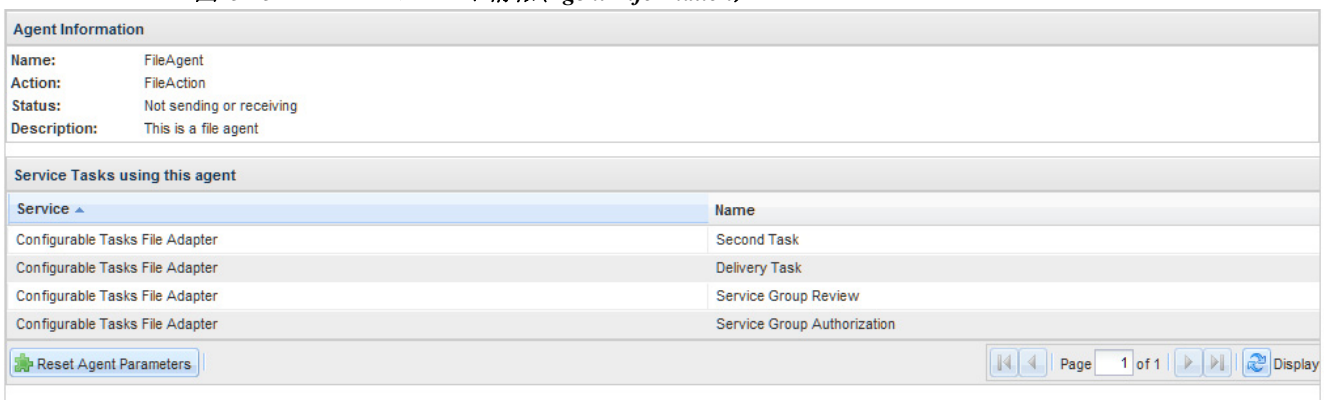
リスト ペインのエージェント エントリが展開されます。編集または確認するエージェント定義の部分のプロパティ シートをクリックします。プロパティ シートが表示されたら、変更を入力し、[保存（Save）] をクリックして保存します。

図 5-14　エージェント エントリ



エージェント名をクリックすると、エージェント定義の概要が表示されます。

図 5-15　エージェント情報（Agent Information）



[エージェントパラメータのリセット（Reset Agent Parameters）] ボタンを除いて、このページは読み取り専用です。

Service Link エージェントの作成と導入

次の手順では、ファイル アダプタを使用する Service Link の統合の導入に必要なタスクの一般的なシーケンスを示します。これは Service Link のインストールの検証にも使用できます。

手順 1　Service Link ホーム ページの [共通タスク（Common Tasks）] 領域から、エージェント ウィザードをクリックし、場所フィールドに入力して他の送信アダプタのプロパティを指定することにより、送信ファイル アダプタを使用するエージェントを選択します。（このようなプロパティの詳細については、ファイル アダプタで説明します）。

手順 2　[エージェントの管理(Control Agents)] タブに移動し、エージェントを見つけ、エージェント名(エージェント定義へのリンク)を除く境界線上の任意の場所でマウスをクリックし、ページの右上にある [選択項目の開始(Start Selected)] をクリックして、エージェントを起動します。起動したかどうかを確認します。起動しない場合は、Service Link コンフィギュレーション設定のいずれかが間違っているか、Integration Server(ISEE)が正しく起動していません。

手順 3　入力したファイル ディレクトリがアプリケーション サーバ上に存在することを確認し、存在しない場合は作成します。サービス カタログ(Service Catalog)と外部アプリケーションの両方に、これらのディレクトリへの適切なアクセス権(書き込み/読み取り)があることを確認します。これらの条件が満たされていない場合は、実行時にファイル転送が失敗します。

手順 4　[Service Designer] に移動し、このエージェントを使用するサービスを作成します。

手順 5　[My Services] に移動し、サービスをオーダーします。

手順 6　要求が正常に作成されていれば、ISEE 送信キューは動作しています。[our apologies] ページが表示された場合は、JMS キューが動作していません。

手順 7　[トランザクションの表示(View Transactions)] タブからアクセスできる [メッセージ(Messages)] ページに移動します。作成した要求からメッセージが表示されていれば成功です。メッセージのステータスが「Message sent」になっているはずです。

手順 8　発信ファイル ディレクトリ(C:\cisco\SL\OutboundFiles など)に移動します。ここに XML ファイルがある場合は(XML ファイルの日時スタンプを調べて、要求に対応する新しいファイルであることを確認してください)、ファイル エージェントの送信が完了しています。おめでとうございます。発信 XML ファイルは有効な nsXML メッセージになります。

手順 9　[メッセージタイプ(Message Type)] カラムの要求に対して、[タスクの実行(Execute Task)] リンクをクリックします。[Message Details] ページが表示されます。

手順 10　Requisition ID が正しいことを確認します。メッセージの詳細画面から「Channel ID」をコピーします。

手順 11　以下に示すように、SampleInbound.xml という名前の XML ファイルを作成します。「insert your Channel ID here」と表示されている場所に、前のステップでコピーした Channel ID の値を貼り付けます。(二重引用符はそのまま残しておきます)。

```
<?xml version="1.0" encoding="UTF-8"?>
<message channel-id="insert your Channel ID here">
    <take-action action="done"/>
</message>
```

たとえば、Channel ID の値を貼り付けると、SampleInbound.xml ファイルは次のようになります。

```
<?xml version="1.0" encoding="UTF-8"?>
<message channel-id="3F2504E0-4F89-11D3-9A0C-0305E82C3301">
    <take-action action="done"/>
</message>
```

手順 12　着信ファイル ディレクトリ(C:\cisco\SL\InboundFiles など)に SampleInbound.xml ファイルを配置します。

手順 13　ファイル エージェントが入力をポーリングすると、受信ファイルが自動的に選択されます。(ファイル アダプタのポーリング間隔時間のデフォルトの設定は 10 秒ごとです)。SampleInbound.xml ファイルは、正常に処理されるとディレクトリから削除されます。

手順 14　[トランザクションの表示(View Transactions)] タブの [メッセージ(Messages)] ページに移動し、要求を探します。Type=Take Action が含まれる要求に対する別のメッセージがあり、Status=Inbound Message Completed と表示されている場合は、往復が実行されたことを意味します。

手順 15　[要求ID(Requisition ID)] リンクをクリックして、[要求ステータス(Requisition Status)] ページを開きます。要求のステータスが「Closed (1 of 1 completed)」になっていることを確認します。

Service Link エージェントを使用するタスクの設定

エージェントを定義すると、そのエージェントを呼び出すワークフローが含まれる外部タスクを作成することにより、サービスで使用できるようになります。ワークフローを設定したら、中に含まれるエージェントに定義されている任意のエージェント パラメータを確認したり、上書きしたりできます。

外部タスクの作成

Service Link エージェントを使用する外部（他社製）アプリケーションにタスクを送信するには、次の手順を実行します。

手順 1　Service Designer を起動します。外部タスクに含めるサービスを選択します。[計画（Plan）] タブをクリックします。

手順 2　外部タスクの指定には、[Tasks] サブタブまたは [Graphical Designer] サブタブを使用できます。

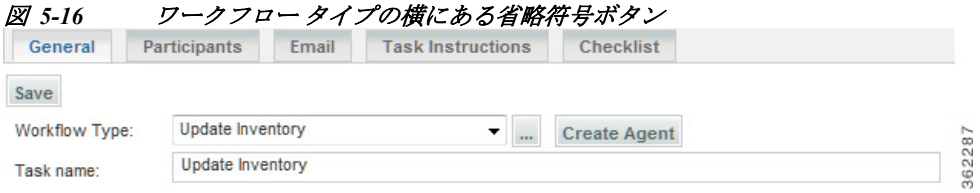
- サービスの [Tasks] サブタブを使用して、タスクを定義し、配信計画に正しいシーケンスで配置します。
- [Graphical Designer] サブタブを使用して、図上にタスクを作成し、Associate ツールを使用して、図に正しいシーケンスで配置します。そのタスクをダブルクリックして、プロパティ シートを表示します。

手順 3　[General] タブの [Workflow Type] ドロップダウン リストを使用して、ドロップダウン リストから目的のアクションを選択します。これは Service Link モジュール定義済みのアクションであり、このドロップダウン リストに一覧表示されるエージェント名ではないことに注意してください。

手順 4　ワークフロー/タスク プランを保存します。Workflow Designer を使用した場合は、[Tasks] サブタブに戻ります。

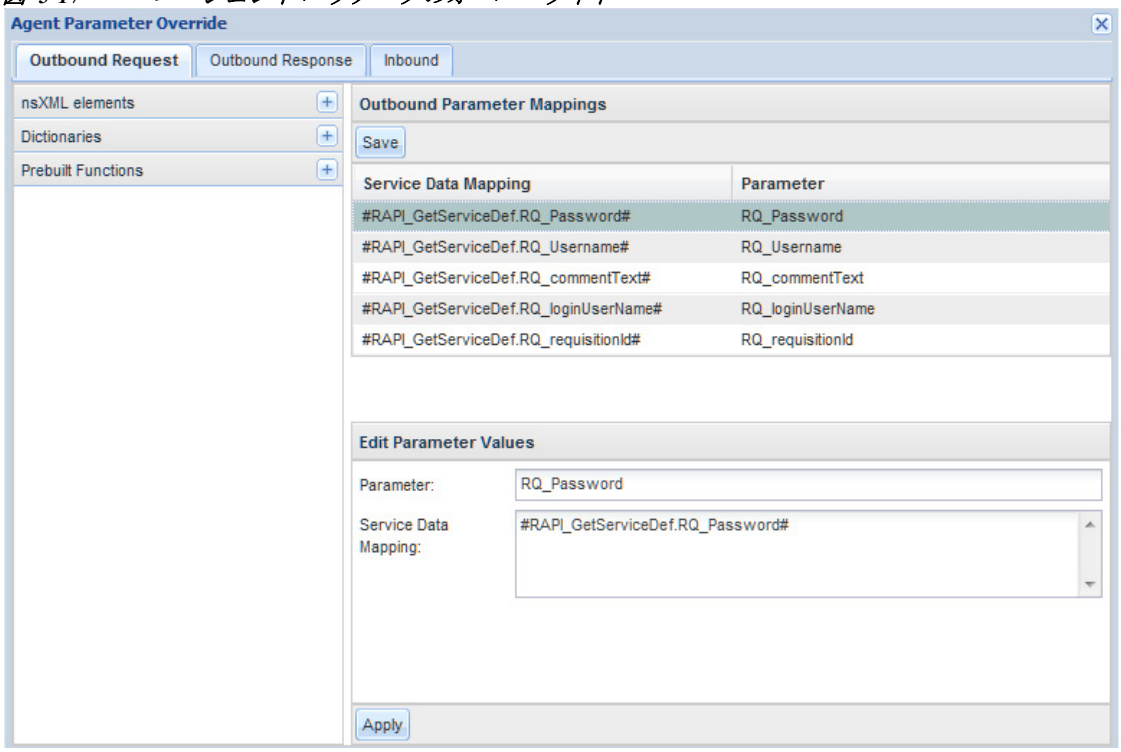
手順 5　外部タスクを保存すると、[Workflow Type] の横に省略符号ボタンが表示されるようになります。この省略記号をクリックすることにより、現在エージェントに有効なパラメータ マッピングを確認したり（存在する場合）、この特定のサービスのマッピングを変更したりできます。**省略符号**ボタンをクリックします。

図 5-16　ワークフロー タイプの横にある省略符号ボタン



手順 6　[エージェントパラメータのオーバーライド（Agent Parameter Override）] ポップアップ ウィンドウが表示されます。エージェント パラメータを確認します。または、1 つ以上のパラメータのマッピングを変更します。各パラメータを変更するたびに [適用（Apply）] をクリックし、ウィンドウを閉じる前に [保存（Save）] をクリックしてください。

図 5-17　エージェント パラメータのオーバーライド



手順 7　[参加者（Participants）] タブで、パフォーマー（個人、キュー、または役職）を定義することもできます。これにより、その実行エンティティのカレンダーを使用してタスクの実行期日が計算されます。外部タスクの [参加者（Participants）] タブで値を設定していない場合は、デフォルト サービス キューのカレンダーを使用して実行期日が計算されます。この方法でプランに実行期日を設定すると、サービス カタログ（Service Catalog）で、外部タスクの提供動作レベル契約（OLA）の順守、および当該タスクを含むサービスのサービス レベル契約（SLA）の順守を計算できます。

エージェント マッピングとサービス定義の同期

エージェントを使用するタスクを作成して保存すると、そのエージェントに指定されているエージェント パラメータ マッピングが個々のタスクに自動的に継承されます。前述のように、サービス設計者は、すべてのエージェント マッピングをタスク レベルで上書きできます。

ただし、後で異なるエージェント パラメータ マッピング セットが含まれるようにエージェントを変更すると、そのエージェントを使用するように事前に定義されているタスクは、変更を自動的に継承しません。これには、次のような変更が含まれます。

- エージェント パラメータの追加
- エージェント パラメータの削除
- 既存のエージェント パラメータのマッピングの変更

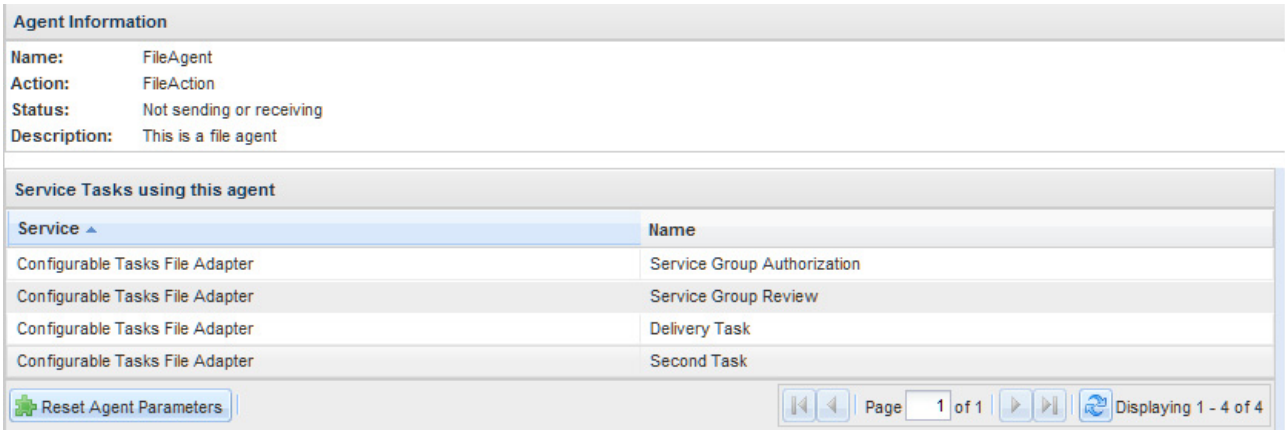
エージェントを使用するサービスにこれらの変更を伝播するには、次の手順に従います。

手順 1　[Service Link Manage Integrations] タブの [Agents] サブタブをクリックします。

手順 2　パラメータ マッピングが変更されたエージェントの名前をクリックします。

[エージェント情報（Agent Information）] ページが表示されます。

図 5-18　[エージェント情報（Agent Information）] ページ



手順 3　エージェント パラメータ マッピングをアップデートされたエージェント定義と再同期する必要があるサービスを選択します。

手順 4　[選択されたタスクのリセット（Reset Selected Tasks）] をクリックします。このボタンはすべてのパラメータ マッピングをエージェントのデフォルトにリセットするため、タスク固有のすべてのマッピングを再適用する必要があります。

nsXML メッセージ

変換には正しい形式の XML を含める必要があるだけではなく、正しい形式の有効な nsXML メッセージを生成する必要があります。すべての nsXML メッセージが nsXML スキーマ（XML 文書の構造を記述する XML 文書）に従っている必要があります。このスキーマは、アプリケーション サーバ（ISEE.war/WEB-INF/classes/xsl/nsxml.xsd）で使用できます。

発信 nsXML メッセージ

外部タスクのステータスが Ongoing に変わると、発信 nsXML メッセージが生成されます。

各メッセージに生成された nsXML は、[Message Details] ページで nsXML メッセージをクリックすると、Service Link モジュールに表示できます。タスクに関連するデータおよび親要求に関連付けられているデータが表示されます。

nsXML において最も重要な要素は channel-id です。これは、外部タスクを一意に識別する ID です。この ID は他社製システムに提供され、対応するデータ アップデートがビジネス エンジンによって正常に適用されるためには、応答に表示される必要があります。

channel-id は、Globally Unique Identifier（GUID）として書式設定されます。GUID は、通常、次のような 16 進数のシーケンスとして、テキスト形式で書き込まれます。

```
3F2504E0-4F89-11D3-9A0C-0305E82C3301
```

このテキスト表記は、ハイフンで区切られた 5 つのデータ セットで構成されます。GUID の合計長は 36 文字（32 の文字と 4 つのハイフン）です。

2 つの送信メッセージ タイプがあります。

task-started

task-started メッセージ タイプは、外部タスクが開始されると生成されます。task-started メッセージの要素の詳細については、Adapter Development Kit による統合の設計を参照してください。

```
<message channel-id="3F2504E0-4F89-11D3-9A0C-0305E82C3301">
    <task-started task-type="task">
       <task>
       .   .    .
```

task-canceled

task-canceled メッセージ タイプは、外部タスクが含まれる要求が取り消されると生成されます。(サービスの配信計画の対応する設定から)外部タスクが実行された後、ユーザが要求のキャンセルを許可されていない場合、このメッセージは生成されません。しかし、要求の取り消しが許可されている場合は、外部タスクに関与エージェント内で使用されている変換によって task-started メッセージと task-canceled メッセージの両方が「スマートに」処理される必要があります。変換によって task-canceled メッセージ タイプをテストし、適切なメッセージを外部システムに送信する必要もあります。

```
<xsl:if test="/message/task-started">
    <-- Original XSLT goes here/>
</xsl:if>

<xsl:if test="/message/task-canceled">
    <-- XSLT for the cancel message goes here/>
</xsl:if>
```

着信 nsXML メッセージ

他社製システムの要求操作およびサービス項目操作からの着信メッセージについては、2 種類の操作がサポートされています。要求操作は、要求データおよびタスク ステータスの更新に使用されます。サービス項目操作は、要求に関連付けられたサービス項目の追加、変更、削除および取得に使用されます。特定の操作は「複合メッセージ」と呼ばれる 1 個の着信メッセージに統合されることがあります。各操作の詳細および制約事項は次の項で説明します。

要求操作

他社製システムは、対応する発信メッセージの channel-id を参照して、外部タスク用に 1 つ以上の着信メッセージを送信できます。外部タスクは処理操作の 1 つが送信されると完了し、これによって許可/配信計画の次のタスクが進行します。

take-action

take-action メッセージは、タスクのステータスを変更するために、承認または提供タスクに適用できます。take-action タグの action 属性は、実行されるアクションを識別します。有効なアクションの概要は次のtake-action メッセージの表のとおりです。

表 5-10　take-action メッセージの表

操作	タスク タイプ	説明
done	提供タスク	提供タスクを完了済みとしてマーキングします。
cancel	提供タスク	配信タスクをキャンセルします。

表 5-10　take-action メッセージの表(続き)

操作	タスク タイプ	説明
OK	確認タスク	確認タスクを完了済みとしてマーキングします。
reject	承認タスク	承認を拒否します。
approve	承認タスク	承認を許可します。

タスク プランの最後の提供タスクが done とマーキングされている場合は、要求が閉じられます(完了します)。take-action タグの「アクション」属性を対応する値に設定すると、許可タスクを Approved または Rejected とマークできます。

```
<?xml version="1.0" encoding="UTF-8"?>
<message channel-id="3F2504E0-4F89-11D3-9A0C-0305E82C3301">
    <take-action action="done"/>
</message>
```

send-parameters

パラメータは、エージェント定義内のディクショナリ フィールドにバインドされたデータ要素です。send-parameters メッセージ タイプを使用すると、指定した 1 つ以上のパラメータがアップデートされ、続いてサービス内の対応するディクショナリ フィールドがアップデートされます。このタイプの着信メッセージの使用は、サービス要求で使用されるディクショナリ フィールドを外部システムで更新する推奨方法です。

```
<?xml version="1.0" encoding="UTF-8"?>
<message channel-id="3F2504E0-4F89-11D3-9A0C-0305E82C3301">
    <send-parameters>
           <agent-parameter>
            <name>Status</name>
              <value>Resolved</value>
           </agent-parameter>
           <agent-parameter>
            <name>ResolvedBy</name>
            <value>Help Desk</value>
           </agent-parameter>
    </send-parameters>
</message>
```

add-comments

add-comments メッセージは、要求の [System Comments] セクションにコメントを追加するために使用します。

```
<?xml version="1.0" encoding="UTF-8"?>
<message channel-id="3F2504E0-4F89-11D3-9A0C-0305E82C3301">
    <add-comments>
        <comment>Test Comment</comment>
    </add-comments>
</message>
```

サービス項目操作

サービス項目は Service Item Manager で定義されるエンティティです。そのライフサイクルは、サービス項目インスタンスがプロビジョニングされた時点から、廃棄される時点までで、サービス要求に関連付けられます。サービス項目のライフサイクル イベントが外部システムによって処理される場合、Service Link サービス項目の作成、更新、削除、および取得のメッセージを介して、サービス項目データを Lifecycle Center と同期することができます。これらのメッセージ タイプは、Web サービスベースのサービス項目リスナー アダプタだけでサポートされます(サービス項目リスナー アダプタを参照)。

これらのメッセージには、1 つまたは複数のサービス項目タイプおよびサービス項目インスタンスを含めることができます。1 つのメッセージに複数のサービス項目操作を組み合わせることはできません。言い換えると、操作の作成、更新、または削除は個別の着信メッセージで送信する必要があります。エラー状態が発生すると、同じメッセージのすべてのサービス項目の操作がロールバックされます。

日時タイプのサービス項目属性は、YYYY-MM-DD HH:MI:SS または YYYY-MM-DD の形式で指定する必要があります。すべての時刻は UTC 時間で保存されます。

サービス項目のサブスクリプションは、サービス項目インスタンスの作成時にオプションで含めることができます。操作でサブスクリプション情報が提供されていないと、要求のカスタマーとその個人のホーム組織単位に項目が割り当てられます。メッセージのサブスクリプション セクションにカスタマーのログイン ID または組織単位名が指定されている場合、これらの値はデフォルトのサービス項目の割り当てを上書きするために使用されます。サブスクリプションの処理規則の詳細については、『*Cisco Prime サービス カタログ(Service Catalog)Designer Guide*』の「Service Designer」の章を参照してください。

```
<?xml version="1.0" encoding="UTF-8"?>
<message channel-id="3F2504E0-4F89-11D3-9A0C-0305E82C3301">
<>
      <serviceitem>
         <name>LaptopComputer</name>
         <serviceItemData>
            <serviceItemAttribute name="Name">LT-LENVT60-17032</serviceItemAttribute>
            <serviceItemAttribute name="Model">Thinkpad T60</serviceItemAttribute>
            <serviceItemAttribute name="Brand">LENOVO</serviceItemAttribute>
            <serviceItemAttribute name="Price">899.99</serviceItemAttribute>
            <serviceItemAttribute name="Memory">3</serviceItemAttribute>
            <serviceItemAttribute name="ManufactureDate">2009-04-15
12:00:00</serviceItemAttribute>
            <subscription>
               <loginID>jsmith</loginID>
               <ouname>Finance</ouname>
               <accountName>account1</accountName>
               <agreementName>agreement</agreementName>
            </subscription>
         </serviceItemData>
      </serviceitem>
      <serviceitem>
         <name>DesktopComputer</name>
         <serviceItemData>
            <serviceItemAttribute name="Name">DT-DELLV200-02274</serviceItemAttribute>
            <serviceItemAttribute name="Model">Vostro 200</serviceItemAttribute>
            <serviceItemAttribute name="Brand">DELL</serviceItemAttribute>
            <serviceItemAttribute name="Price">755.99</serviceItemAttribute>
            <serviceItemAttribute name="Memory">4</serviceItemAttribute>
            <serviceItemAttribute name="ManufactureDate">2010-03-01
12:00:00</serviceItemAttribute>
         </serviceItemData>
      </serviceitem>
   </>
</message>
```

update

更新メッセージでは、サービス項目属性を省略すると、属性値は変更されません。メッセージで属性が明示的に指定されているものの、値が含まれない場合、そのサービス項目の属性値はテキスト フィールドでは空白に設定され、数値フィールドでは 0 に設定されます。

```
<?xml version="1.0" encoding="UTF-8"?>
<message channelId="3F2504E0-4F89-11D3-9A0C-0305E82C3301">
<update>
      <serviceitem>
         <name>LaptopComputer</name>
         <serviceItemData>
            <serviceItemAttribute name="Name">LT-LENVT60-6122</serviceItemAttribute>
            <serviceItemAttribute name="Memory">4</serviceItemAttribute>
            <subscription>
               <loginID>dcohen</loginID>
            </subscription>
         </serviceItemData>
      </serviceitem>
      <serviceitem>
         <name>DesktopComputer</name>
         <serviceItemData>
            <serviceItemAttribute name="Name">DT-DELLV200-00394</serviceItemAttribute>
            <subscription>
               <loginID></loginID>
               <ouname></ouname>
               <accountName>account</accountName>
               <agreementName>agreement1</agreementName>
            </subscription>
         </serviceItemData>
      </serviceitem>
   </update>
</message>
```

削除

delete サービス項目要求に必要なのは、サービス項目タイプおよびインスタンスの名前だけです。その他のサービス項目の属性およびサブスクリプション情報は無視されます。

```
<?xml version="1.0" encoding="UTF-8"?>
<message channelId="3F2504E0-4F89-11D3-9A0C-0305E82C3301">
<delete>
      <serviceitem>
         <name>LaptopComputer</name>
         <serviceItemData>
            <serviceItemAttribute name="Name">LT-TOSH900-0021</serviceItemAttribute>
         </serviceItemData>
      </serviceitem>
      <serviceitem>
         <name>DesktopComputer</name>
         <serviceItemData>
            <serviceItemAttribute name="Name">DT-DELLV100-0394</serviceItemAttribute>
         </serviceItemData>
      </serviceitem>
   </delete>
</message>
```

getRequest

getRequest 操作はサービス項目インスタンスを取得するために使用されます。update および delete サービス項目要求と異なり、channel-id と topic-id 属性は任意選択です。各着信メッセージには getRequest 操作を 1 つだけ含めることができ、その中にサービス項目タイプを 1 つだけ含めることができます。Service Link ユーザ インターフェイスのような要求のロギングは行われません。

サービス項目属性とサブスクリプション データ（お客様のログイン ID、組織単位名、アカウント ID、および契約）を使用して、要求 XML で指定した検索フィルタに従って、サービス項目インスタンスが取得されます。getRequest では最高 5 個のフィルタを使用でき、これらは AND 結合と解釈されます。次のgetRequest の検索フィルタ演算子の表に、検索フィルタでサポートされる演算子を示します。

表 5-11　getRequest の検索フィルタ演算子の表

データタイプ	サポートされるフィルタ演算子
STRING(32)	Equals、LessThan、LessThanOrEqualTo、GreaterThan、GreaterThanOrEqualTo、StartsWith、EqualsIgnoreCase、isNull、isNotNull、Between、NotEqualsTo
STRING(128)	Equals、LessThan、LessThanOrEqualTo、GreaterThan、GreaterThanOrEqualTo、StartsWith、EqualsIgnoreCase、isNull、isNotNull、Between、NotEqualsTo
STRING(512)	Equals、LessThan、LessThanOrEqualTo、GreaterThan、GreaterThanOrEqualTo、StartsWith、EqualsIgnoreCase、isNull、isNotNull、Between、NotEqualsTo
INTEGER	Equals、LessThan、LessThanOrEqualTo、GreaterThan、GreaterThanOrEqualTo、StartsWith、EqualsIgnoreCase、isNull、isNotNull、Between、NotEqualsTo
MONEY	Equals、LessThan、LessThanOrEqualTo、GreaterThan、GreaterThanOrEqualTo、StartsWith、EqualsIgnoreCase、isNull、isNotNull、Between、NotEqualsTo
LONG INTEGER	Equals、LessThan、LessThanOrEqualTo、GreaterThan、GreaterThanOrEqualTo、StartsWith、EqualsIgnoreCase、isNull、isNotNull、Between、NotEqualsTo
DOUBLE FLOAT	Equals、LessThan、LessThanOrEqualTo、GreaterThan、GreaterThanOrEqualTo、StartsWith、EqualsIgnoreCase、isNull、isNotNull、Between、NotEqualsTo
DATE TIME	Equals、LessThan、LessThanOrEqualTo、GreaterThan、GreaterThanOrEqualTo、EqualsIgnoreCase、isNull、isNotNull、Between、NotEqualsTo

getRequest からの応答データにはサービス項目の属性名と値、およびサブスクリプション情報が含まれます。それぞれの getRequest 操作で返されるレコードの最大数は 100 です。次のレコードセットは、要求で「startRow」および「count」要素を指定することによって取得できます。startRow 要素は結果セットの開始行番号を示します。count 要素は返されるレコード数を示します。「startRow」および「count」のデフォルト値はそれぞれ 1 と 100 です（要求 XML にない場合）。count の値は、パフォーマンス上の理由から 100 に制限されています。

以下は getRequest XML の例です。

```
<getRequest>
   <serviceItemType>LaptopComputer</serviceItemType>
   <startRow>1</startRow>
   <count>1</count>
   <subscription>
      <loginID>jsmith</loginID>
      <ouname>Finance</ouname>
   </subscription>
   <filters>
   <!--1 to 5 repetitions:-->
      <filter attributeName="Name" operator="Equals" value="LT-LENVT60-17032" />
      <filter attributeName="Price" operator="GreaterThan"  value="800"/>
     <filter attributeName="ManufactureDate" operator="GreaterThan"  value="2004-04-10"/>
   </filters>
</getRequest>
```

上記の要求の応答

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soap:Body xmlns:ns1="http://externaltask.api.newscale.com">
      <response channel-id="CHANNELID not retrieved"
xmlns="http://externaltask.api.newscale.com">
         <status-code>success</status-code>
         <status-message>Service item data read successfully.</status-message>
         <getResponse>
            <serviceitem>
               <name>LapTopComputer</name>
               <serviceItemData>
                  <serviceItemAttribute
name="Name">LT-LENVT60-17032</serviceItemAttribute>
                  <serviceItemAttribute name="Brand">LENOVO</serviceItemAttribute>
                  <serviceItemAttribute name="Memory">3</serviceItemAttribute>
                  <serviceItemAttribute name="Model">Thinkpad T60</serviceItemAttribute>
                  <serviceItemAttribute name="Price">899.99</serviceItemAttribute>
                  <serviceItemAttribute name="ManufactureDate">Fri Apr 16 00:00:00
GMT+05:30 2004</serviceItemAttribute>
                  <subscription>
                     <loginID>jsmith</loginID>
                     <ouname>Finance</ouname>
                    <accountID>1<<accountID>
                    <accountName>tenantaccount<accountName>
                    <agreementID>1<agreementID>
                    <agreementName>agreement<agreementName>
                     <requisitionID>0</requisitionID>
                     <requisitionEntryID>0</requisitionEntryID>
                     <assignedDate>2012-07-20T05:21:29.187+05:30</assignedDate>
                     <submittedDate>2012-07-20T05:17:21.503+05:30</submittedDate>
                  </subscription>
               </serviceItemData>
            </serviceitem>
         </getResponse>
      </response>
   </soap:Body>
</soap:Envelope>
```

getDefinitionRequest

getDefinitionRequest 操作はサービス項目タイプのメタデータまたは定義を取得するために使用されます。getRequest 操作同様、channel-id と topic-id 属性は任意選択です。各着信メッセージには getRequestDefinition 操作を 1 つだけ含めることができ、その中にサービス項目タイプを 1 つだけ含めることができます。Service Link ユーザ インターフェイスのような要求のロギングは行われません。

以下はサービス項目の getDefinitionRequest の例です。

```
<getDefinitionRequest>
   <serviceItemType>LaptopComputer<serviceItemType>
<getDefinitionRequest>
```

上記の要求の応答

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soap:Body xmlns:ns1="http://externaltask.api.newscale.com">
      <response channel-id="" xmlns="http://externaltask.api.newscale.com">
         <status-code>success</status-code>
         <status-message>Service item definition read successfully.</status-message>
```

```
            <getDefinitionResponse>
               <serviceItemDef>
                  <name>LaptopComputer</name>
                  <classification>Laptops</classification>
                  <displayName>LaptopComputer</displayName>
                  <serviceItemProperty name="Name" type="string" />
                 <serviceItemProperty name="Model" type="string" />
                  <serviceItemProperty name="Brand" type="string" />
                  <serviceItemProperty name="Price" type="real64" />
                  <serviceItemProperty name="Memory" type="sint32" />
                  <serviceItemProperty name="ManufactureDate" type="datetime" />
               </serviceItemDef>
            </getDefinitionResponse>
         </response>
      </soap:Body>
   </soap:Envelope>
```

複合メッセージ

上記のメッセージ タイプは、単一の受信メッセージに組み合わせることができます。このような組み合わせは、「複合」メッセージと呼ばれます。実行順序が重要です。take-action タグを含める前に、パラメータの送信やコメントの追加を実行し、最後にサービス項目操作タグを配置する必要があります。

```
<?xml version="1.0" encoding="UTF-8"?>
<message channel-id="3F2504E0-4F89-11D3-9A0C-0305E82C3301">
    <add-comments>
          <comment>Task closed per override …</comment>
    </add-comments>
    <send-parameters>
           <agent-parameter>
            <name>Status</name>
               <value>Resolved</value>
           </agent-parameter>
    </send-parameters>
    <take-action action="done"/>
<update>
      <serviceitem>
         <name>LaptopComputer</name>
         <serviceItemData>
            <serviceItemAttribute name="Name">LT-LENVT60-6122</serviceItemAttribute>
            <serviceItemAttribute name="Memory">4</serviceItemAttribute>
            <subscription>
               <loginID>dcohen</loginID>
            </subscription>
         </serviceItemData>
      </serviceitem>
   </update>
</message>
```

SIM インポート メッセージ

Service Item Manager（SIM）インポート メッセージ タイプは、外部システムからサービス カタログ（Service Catalog）へのサービス項目と標準定義およびデータをサポートしています。サービス項目の更新および削除の操作とは異なり、SIM のインポートは、特定のディレクトリに配置された着信ファイルをポーリングする File Adapter プロトコルに基づいています。サービス項目インスタンスの操作に加えて、SIM Import ではサービス項目グループおよびサービス項目タイプのメンテナンスもサポートされます。Service Item Manager のインポートの詳細については、『*Cisco Prime サービス カタログ（Service Catalog）Designer Guide*』を参照してください。

変換と nsXML

発信 nsXML メッセージは通常、非常にサイズが大きく複雑で、500 KB を超えることもよくあります。変換を使用してメッセージ形式を変更することは必須ではありませんが、外部システムが nsXML を読み取るように設定されていることはまずありません。結果として、通常、変換を使用して送信メッセージ形式を変更することは不可避です。

ただし、着信メッセージの形式についてはサードパーティ システムの担当者と相談することが多いため、nsXML メッセージ形式に近い仕様に合意することも可能です。その場合、受信変換の方が、対応する送信変換よりもはるかに簡素化できる可能性があります。

XSL Transformations（XSLT）と呼んではいますが、使用されているテクノロジーは、実際には拡張可能スタイルシート言語と呼ばれ、XPATH も含みます。XPATH は、XML ドキュメント内で情報を検索し、要素および属性全体を移動するための言語です。XPATH には、文字列値、数値、日付と時刻の比較、シーケンス操作、ブール値、およびその他のメソッドのための組み込み関数が用意されています。

Service Link トランザクションのモニタリング

Service Link の使用状況をモニタリングするには、次のように複数の方法があります。

- Service Link のホーム ページに、過去 30 日間のメッセージ量のグラフが表示され、他のモニタリング オプションにアクセスするための [Common Tasks] タブや [View Transactions] タブが表示されます。
- また、Service Link のホーム ページには [Recent Failed Messages] を表示するオプションもあり、配信できなかったすべてのメッセージが表示されます。
- [View Transactions] タブからアクセス可能なメッセージを表示するオプションでは、Service Link で送受信するすべてのメッセージが表示され、管理者は目的のメッセージを表示するためにフィルタリングや検索を実行できます。
- [View Transactions] タブからアクセスできる外部タスクを表示するためのオプションでは、Service Link メッセージを配信できなかったために進行中のままになっているすべてのタスクが表示され、管理者は目的のタスクを表示するためにフィルタリングや検索を実行できます。

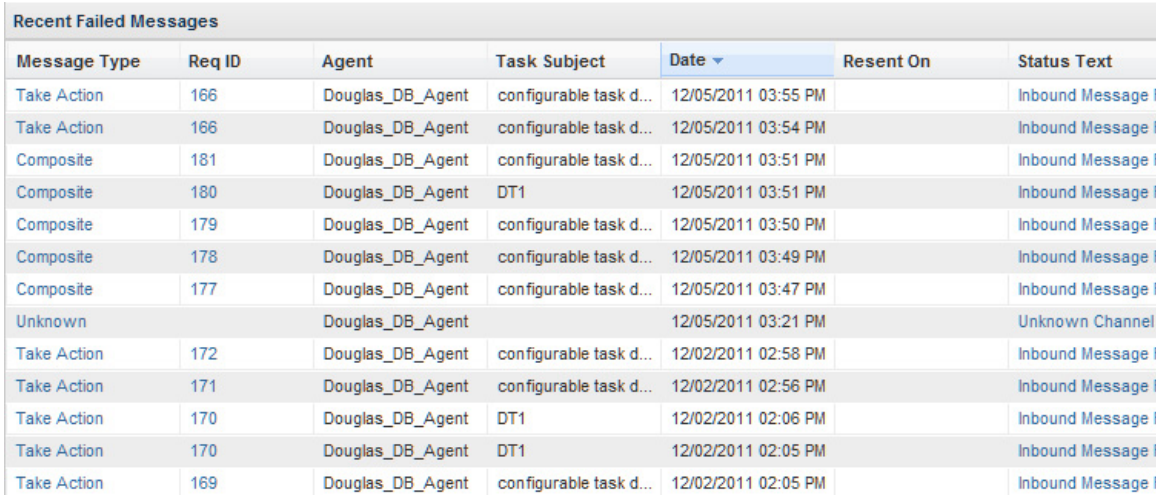
Service Link のモニタリングや管理のためのすべてのページが、設定可能な「データ テーブル」を使用して表示されます。このテーブルの外観（表示されるカラム、各カラムの幅、およびデータの表示順序）はカスタマイズ可能です。また、フィルタと検索機能により、管理者は必要な行だけを表示できます。

レート制限によって簡単に REST 呼び出しを制限でき、悪意あるユーザからの分散 Dos（DDoS）攻撃というセキュリティ上の脅威を防ぎます。アプリケーション レベルのレート制限によって、効果的なロード バランシングも行われます。REST API 要求のレート制限の設定を参照してください。

Service Link ホーム ページからのメッセージの表示

Service Link のホーム ページの [Recent Failed Messages] ペインには、過去 30 日以内に宛先に配信できなかった Service Link メッセージが表示されます。デフォルトでは、メッセージは、送信日時に基づいて、最新のものから順に表示されます。

図 5-19　失敗メッセージ



Recent Failed Messages

Message Type	Req ID	Agent	Task Subject	Date	Resent On	Status Text
Take Action	166	Douglas_DB_Agent	configurable task d...	12/05/2011 03:55 PM		Inbound Message
Take Action	166	Douglas_DB_Agent	configurable task d...	12/05/2011 03:54 PM		Inbound Message
Composite	181	Douglas_DB_Agent	configurable task d...	12/05/2011 03:51 PM		Inbound Message
Composite	180	Douglas_DB_Agent	DT1	12/05/2011 03:51 PM		Inbound Message
Composite	179	Douglas_DB_Agent	configurable task d...	12/05/2011 03:50 PM		Inbound Message
Composite	178	Douglas_DB_Agent	configurable task d...	12/05/2011 03:49 PM		Inbound Message
Composite	177	Douglas_DB_Agent	configurable task d...	12/05/2011 03:47 PM		Inbound Message
Unknown		Douglas_DB_Agent		12/05/2011 03:21 PM		Unknown Channel
Take Action	172	Douglas_DB_Agent	configurable task d...	12/02/2011 02:58 PM		Inbound Message
Take Action	171	Douglas_DB_Agent	configurable task d...	12/02/2011 02:56 PM		Inbound Message
Take Action	170	Douglas_DB_Agent	DT1	12/02/2011 02:06 PM		Inbound Message
Take Action	170	Douglas_DB_Agent	DT1	12/02/2011 02:05 PM		Inbound Message
Take Action	169	Douglas_DB_Agent	configurable task d...	12/02/2011 02:05 PM		Inbound Message

[Failed Messages] グリッドで、いずれかのカラム リンクをクリックすると、関連情報が表示されます。

表 5-12　Service Link の [Failed Messages] のクリック可能なカラム

カラム	リンク
メッセージ タイプ	[サービスリンクメッセージの詳細（Service Link Message Details）] ポップアップ ページのメッセージの詳細
Req ID	要求の詳細
Agent（エージェント）	Service Link の [エージェント（Agents）] ページに表示されるエージェントの詳細

[View Transactions] タブから使用できる [Messages] ページでは、ステータスに関係なく、すべての送受信メッセージを表示し、ページに表示されたメッセージを明示的にフィルタリングし、指定した検索条件に一致するメッセージを検索できます。

メッセージの表示

[Messages] ページには、設定したフィルタに応じて、すべてまたは選択した Service Link メッセージが表示されます。デフォルトでは、完了したメッセージは表示されません。[メッセージ（Messages）] ページを表示するには、Service Link のホーム ページの [トランザクションの表示（View Transactions）] タブをクリックします。次に、[メッセージ（Messages）] サブタブをクリックします。Service Link ホーム ページの [Common Tasks] 領域にある [View Failed Messages] リンクでは、ステータスが「Failed」のメッセージだけを表示するようにフィルタが設定された [Messages] ページが表示されます。

次のように、[メッセージ（Messages）] ページが表示されます。

図 5-20　[メッセージ(Messages)] ページ

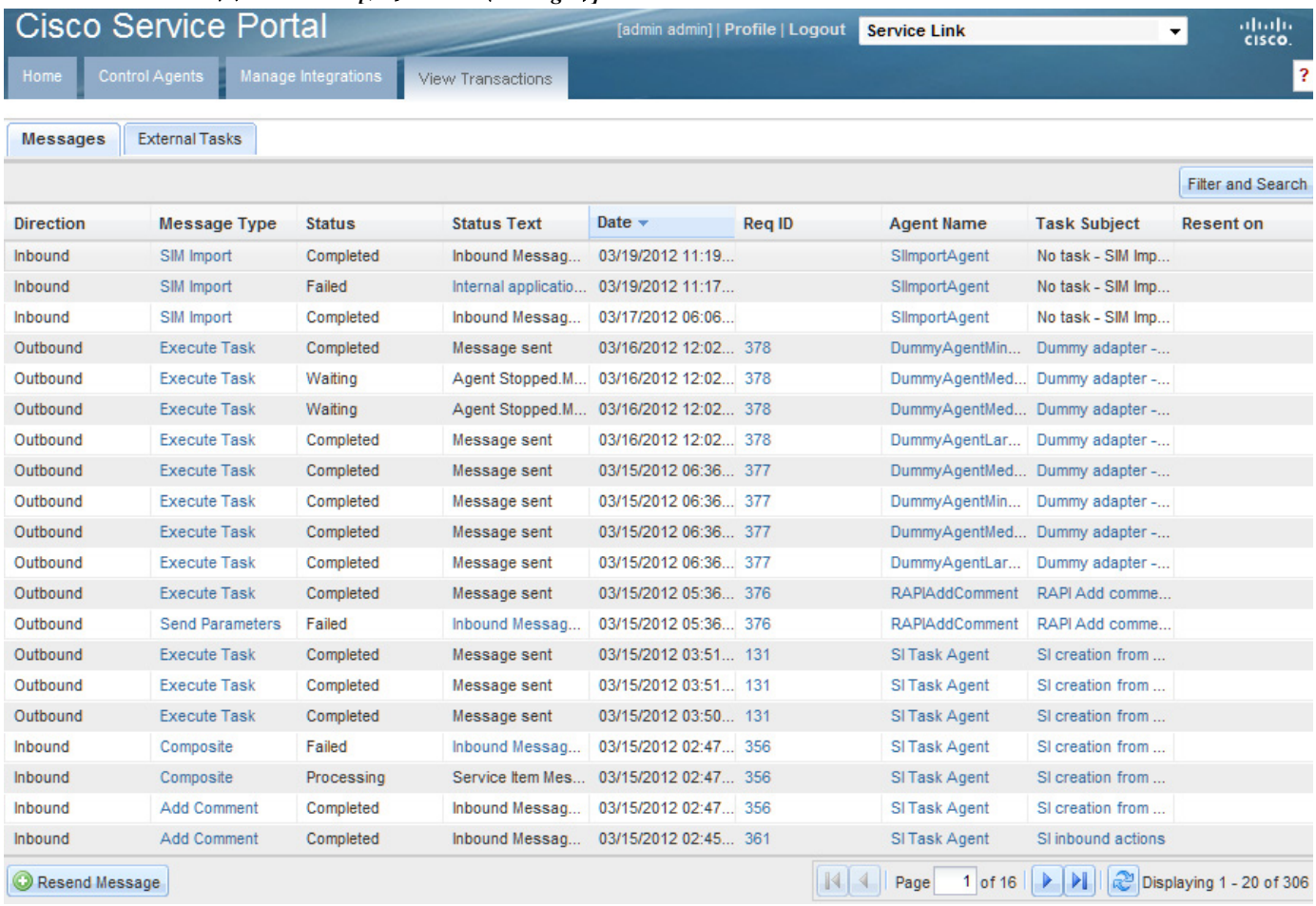


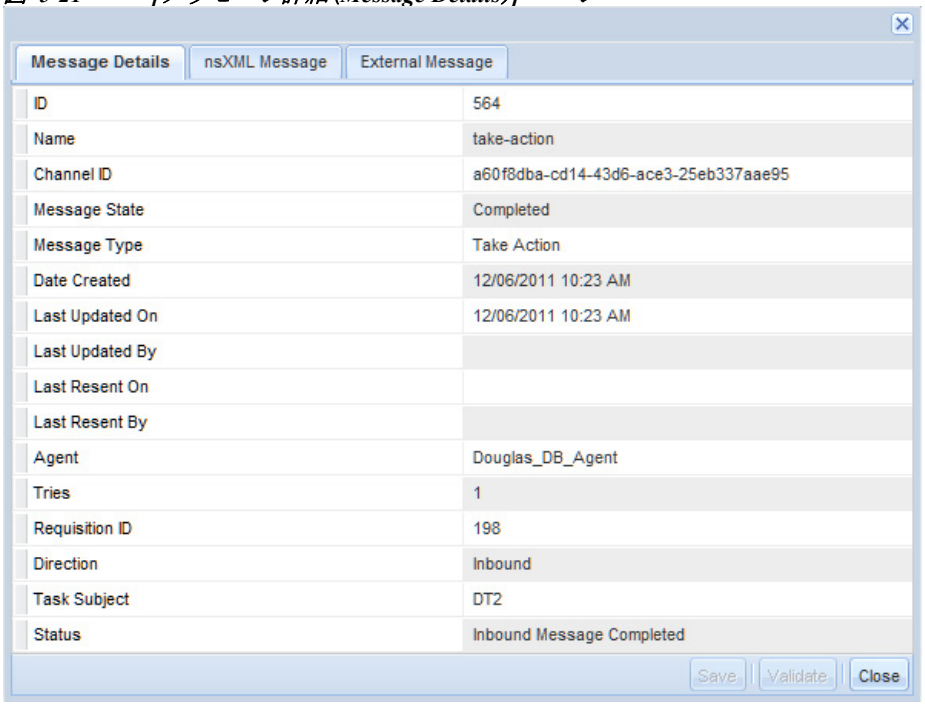
表 5-13　Service Link の [Messages] のクリック可能なカラム

カラム	リンク
メッセージ タイプ	Service Link の [メッセージ詳細(メッセージ詳細(Message Details))] ポップアップ ページに表示されるメッセージの詳細
Status Text	失敗したメッセージを参照するために、アダプタに固有のログ ファイルおよびサーバ ログに書き込まれたエラー メッセージへのリンクを使用できます。詳細については、Service Link のトラブルシューティングと管理を参照してください。
Req ID	要求の詳細。
Agent(エージェント)	Service Link の [エージェント(Agents)] ページに表示されるエージェントの詳細。
Task Subject	Service Manager のタスクの詳細。

メッセージ詳細（Message Details）

[メッセージ詳細（Message Details）] ポップアップ ページには、サービス カタログ（Service Catalog）メッセージと外部メッセージの両方が表示されます。このページには、この要求内のタスクを一意に識別するチャネル ID も表示されます。この ID は、他社製システムの問題を解決する際に使用できます。

図 5-21　[メッセージ詳細（Message Details）] ページ



[メッセージ詳細（Message Details）] ポップアップ ページのいずれかのタブをクリックすると、関連情報が表示されます。

表 5-14　Service Link の [Message Details] のサブタブ

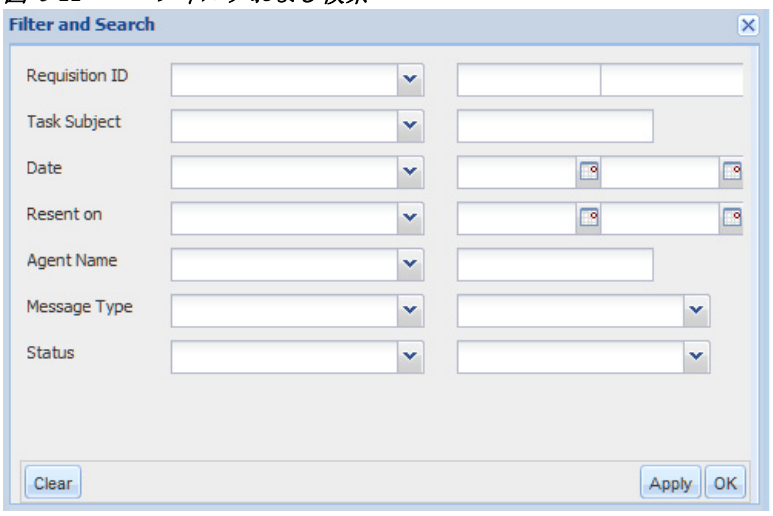
カラム	リンク
メッセージの詳細	メッセージの詳細。
nsXML Message	発信メッセージの場合は、Business Engine によって生成されたメッセージが、Service Link エージェントによって処理（変換）されます。着信メッセージの場合は、外部システムから受信したメッセージが、エージェント変換（存在する場合）によって変換され、Business Engine によって処理されます。
External System Message	発信メッセージの場合、エージェントに関連付けられた変換が適用された後のメッセージです。着信メッセージの場合、外部システムから受信したメッセージです。

フィルタおよび検索

検索機能を使用すると、「**Failed**」ステータスを持つすべてのメッセージなど、メッセージのサブセットを表示できます。検索では、検索対象として [Messages] ウィンドウのいずれかのカラムを指定し、照合する値を選択または入力します。

[フィルタおよび検索（Filter and Search）] をクリックします（[メッセージ（Messages）] ページの最上部にあります）。

図 5-22　フィルタおよび検索



[Filter and Search（Filter and Search）] ダイアログ ボックスでは、次の操作もサポートされます。

- カラムの意味に適した関係演算子を使用して、特定のカラムをフィルタリングする。たとえば、日付範囲の選択や、特定のステータスと同じでない任意のステータスの選択が可能です。
- カラムに指定されたすべての条件の論理「AND」によってフィルタリングする。

[フィルタおよび検索（Filter and Search）] ダイアログボックスは、非モーダルです。任意の基準を入力して、[適用（Apply）] をクリックすると、現在の設定の結果が表示されます。必要であれば、設定を調整して、[適用（Apply）] を再度クリックします。任意のカラムの昇順または降順でメッセージを表示したり、設定操作により表示されるカラムを変更することもできます。

失敗したメッセージの再送信

Service Link の開発中は、エージェントまたは変換設定のエラーのために配信に失敗した、多数のメッセージが生成されることがあります。これらのメッセージは再送信しないでください。同様に、Service Item Import タスクによって生成されたメッセージも再送信しないでください。インポート ファイル形式を変更して、インポート タスクを再試行してください。

ただし、公開環境では、外部システムの停止または修正可能なその他の外部要因のためにメッセージの配信が失敗することがあります。配信障害の原因を修正すると、エラー メッセージを再送信できます。

エラー メッセージを再送信するには、次の手順を実行します。

手順 1　[トランザクションの表示（View Transactions）] タブの [メッセージ（Messages）] ページで、失敗したメッセージが含まれる行をクリックします。

手順 2　[メッセージ（Messages）] ページの左下隅にある [メッセージの再送信（Resend Message）] をクリックします。

Service Link は指定された宛先に、メッセージを再送信しようとします。再送信が正常に行われると、メッセージのステータスと日付がアップデートされ、再送信日付が記録されて [再送信日付（Resent On）] カラムに表示されます。

メッセージの再送信中には、変換は再適用されません。エージェントは、変換済みのメッセージを宛先に送信しようとします。

サービス項目の操作に関する失敗した着信メッセージの再送信はサポートされません。プロセスはタスク処理の再試行を試みます。したがって、このようなメッセージの宛先は、Service Item インポート プロセッサではなく、Business Engine になります。

外部タスクの表示

外部タスクを表示するには、次の手順を実行します。

手順 1　Service Link のホーム ページで、[トランザクションの表示（View Transactions）] をクリックします。次に、[外部タスク（External Tasks）] サブタブをクリックします。

次に示す [外部タスク（External Tasks）] ページが表示されます。

図 5-23　[外部タスク（External Tasks）] ページ

Messages | External Tasks

Filter and Search

Task Subject	Started On	Req ID	Status	Completed On	Agent Name	Service
DT2	12/06/2011 10:22 AM	198	Completed	12/06/2011 10:23 AM	Douglas_DB_Agent	configurable task db
DT1	12/06/2011 10:22 AM	198	Completed	12/06/2011 10:22 AM	Douglas_DB_Agent	configurable task db
configurable task db needs...	12/06/2011 10:22 AM	198	Skipped	12/06/2011 10:22 AM	Douglas_DB_Agent	configurable task db
configurable task db needs...	12/06/2011 10:22 AM	198	Skipped	12/06/2011 10:22 AM	Douglas_DB_Agent	configurable task db
DT2	12/06/2011 10:21 AM	197	Completed	12/06/2011 10:22 AM	Douglas_DB_Agent	configurable task db
DT1	12/06/2011 10:21 AM	197	Completed	12/06/2011 10:21 AM	Douglas_DB_Agent	configurable task db
configurable task db needs...	12/06/2011 10:21 AM	197	Skipped	12/06/2011 10:21 AM	Douglas_DB_Agent	configurable task db
configurable task db needs...	12/06/2011 10:21 AM	197	Skipped	12/06/2011 10:21 AM	Douglas_DB_Agent	configurable task db
DT2	12/06/2011 10:20 AM	196	Completed	12/06/2011 10:21 AM	Douglas_DB_Agent	configurable task db
DT1	12/06/2011 10:19 AM	196	Completed	12/06/2011 10:20 AM	Douglas_DB_Agent	configurable task db
configurable task db needs...	12/06/2011 10:19 AM	196	Skipped	12/06/2011 10:19 AM	Douglas_DB_Agent	configurable task db
configurable task db needs...	12/06/2011 10:19 AM	196	Skipped	12/06/2011 10:19 AM	Douglas_DB_Agent	configurable task db
DT2	12/06/2011 10:19 AM	195	Completed	12/06/2011 10:19 AM	Douglas_DB_Agent	configurable task db
DT1	12/06/2011 10:18 AM	195	Completed	12/06/2011 10:19 AM	Douglas_DB_Agent	configurable task db
configurable task db needs...	12/06/2011 10:18 AM	195	Skipped	12/06/2011 10:18 AM	Douglas_DB_Agent	configurable task db
configurable task db needs...	12/06/2011 10:18 AM	195	Skipped	12/06/2011 10:18 AM	Douglas_DB_Agent	configurable task db
DT1	12/05/2011 05:08 PM	194	Ongoing		Douglas_DB_Agent	configurable task db
configurable task db needs...	12/05/2011 05:08 PM	194	Skipped	12/05/2011 05:08 PM	Douglas_DB_Agent	configurable task db
configurable task db needs...	12/05/2011 05:08 PM	194	Skipped	12/05/2011 05:08 PM	Douglas_DB_Agent	configurable task db
DT1	12/05/2011 05:07 PM	193	Ongoing		Douglas_DB_Agent	configurable task db

Send Manual Message　Page 1 of 30　Displaying 1 - 20 of 593

362312

手順 2　次のカラム リンクのいずれかをクリックすると、関連情報が表示されます。

:

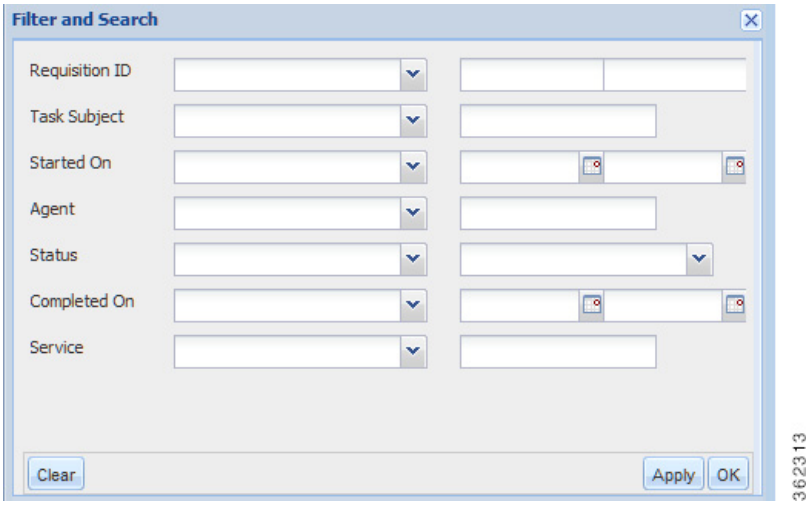
表 5-15　Service Link の [External Tasks] のクリック可能なカラム

カラム	リンク
Task Subject	Service Manager のタスクの詳細
Req ID	[My Services] の要求の概要
エージェント名（Agent Name）	Service Link の [Agents] ページに表示されるエージェントの詳細

フィルタおよび検索

メッセージ表示と同様、[External Tasks] ページには、データ テーブルに表示されているカラムおよびデータの順序をカスタマイズし、そのデータのフィルタリングと検索を行うための機能が用意されています。

図 5-24　フィルタおよび検索



手動メッセージの送信

起動され、着信メッセージを受信することが予期されるタスクは、「Ongoing」状態にあります。着信メッセージは、通常、タスクのアップデートまたはステータスの変更を行います。メッセージが受信され、タスクが完了するまでは、要求の配信計画にある後続のタスクを実行できません。予期されるメッセージはすでに送信されているが、何らかの理由で「消失した」ことが疑われる（または、外部システムの管理者と話をして確認できる）場合は、手動メッセージを送信してメッセージの受信をエミュレートすることができます。

手動メッセージを使用して、失敗したサービス項目の操作をエミュレートすることはできません。

(注)　この機能を利用する場合は注意が必要です。この機能はシステム内のすべての通信プロトコルを上書きします。この機能を使用すると、サードパーティ システムの作成物がそのままになり、Service Link が応答できなくなる可能性があります。また、この機能を使用して要求をキャンセルした場合、たとえば、Service Link が関係先に通知を行わないため、自分でフォロー アップする必要があります。

Business Engine に手動メッセージを送信するには、次の手順を実行します。

手順 1　Service Link のホーム ページで、[トランザクションの表示(View Transactions)] をクリックします。次に、[External Tasks] をクリックします。

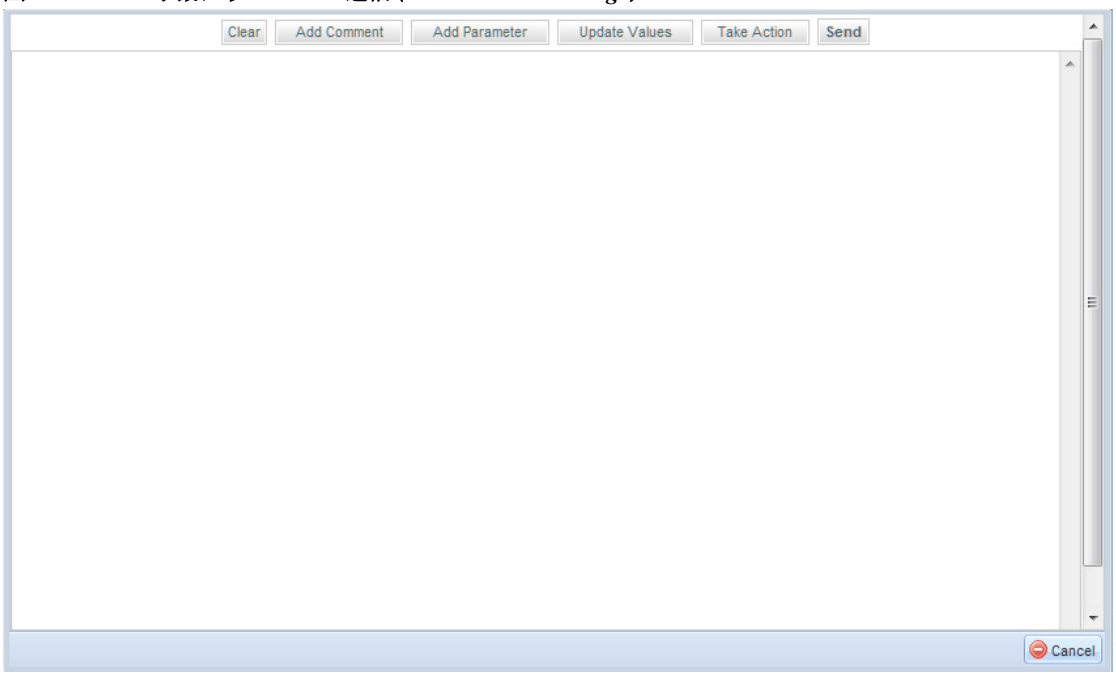
[External Tasks] ページが表示されます。

手順 2　[外部タスク(External Tasks)] ページの左下隅で、手動メッセージを送信するタスクが含まれている行をクリックします。

手順 3　[手動メッセージの送信(Send Manual Message)] をクリックします。

次に示す [手動メッセージの送信(Send Manual Message)] ダイアログ ボックスが表示されます。

図 5-25　手動メッセージの送信(Send Manual Message)



手順 4　送信するメッセージのタイプに応じて [Add Comment]、[Add Parameter]、[Update Values]、または [Take Action] をクリックします。

表 5-16　操作と説明

アクション	説明
Add Comment	add-comments メッセージを送信し、システム コメントを要求に追加します。
Add Parameter	send-parameters メッセージを送信します。1 つまたは複数の受信エージェント パラメータ値、およびこのパラメータがバインドされた対応するディクショナリ フィールドの値を変更します。
Update Values	指定されたディクショナリ フィールドの内容を変更するメッセージを送信します。(このメッセージ タイプは、主に旧バージョンとの下位互換性を提供する目的で提供されており、フィールドの内容は、通常、受信エージェント パラメータを使用してアップデートする必要があります)。
Take Action	take-action メッセージを送信します。Mark タスクが実行された(完了した)かキャンセルされたかをマークし、承認を許可または拒否し、あるいは確認が OK とマークします。

手順 5　選択したメッセージ タイプに関連付けられているポップアップ ダイアログボックスに応答します(ポップアップ ブロッカはオフにしてください)。これにより、メッセージ ウィンドウに、適切なタイプの整形式 XML メッセージが入力されます。また、このメッセージが通常のチャネルを通じて受信されたのではなく、手動で生成されたことを示す <add-comments> メッセージも含められます。

手順 6　必要に応じて、生成されたメッセージを編集できます。メッセージ全体を作成したら、[送信(Send)] をクリックします。受信メッセージが Business Engine に送信されます。

Service Link メッセージの再発行

まれに、Service Link アプリケーションが長時間停止したり、コンフィギュレーションが正しくないために、外部タスクに Service Link で作成された対応する発信メッセージを含めることができないことがあります。

Service Link で根本原因が解決し、アプリケーションが再び起動すると、問題の外部タスクを Service Link に再発行し、発信メッセージを作成して配信プロセスを再開することができます。

発信メッセージを再発行するには、次の手順に従います。

手順 1　Service Link のホーム ページで、[トランザクションの表示(View Transactions)] をクリックします。その後、[メッセージの再発行(Message Republish)] をクリックします。

手順 2　左側のペインに、1 つまたは複数の不明な発信 Service Link メッセージがある要求の Requisition ID を入力します。要求に関連するすべての承認タスクおよび配信タスクが評価され、発信メッセージの作成のために再発行が必要なタスクだけが処理されます。一度に最大 20 の要求を入力できます。

手順 3　[Republish] をクリックします。

手順 4　再発行プロセスが完了したら、右側のペインで処理のステータスを確認します。

Service Link アダプタの管理

すべての Service Link アダプタで、データ交換形式として nsXML がサポートされます。nsXML 形式の詳細については、Adapter Development Kit による統合の設計を参照してください。

すべてのポーラーベースのアダプタで、一度の呼び出しにつき 1 つのメッセージの処理だけがサポートされます。

すべてのアプリケーション インスタンスに次の Service Link アダプタがインストールされます。

- ダミー アダプタ
- データベース アダプタ
- ファイル アダプタ
- HTTP/WS アダプタ
- JMS アダプタ
- MQ アダプタ
- サービス項目リスナー アダプタ
- Web サービス リスナー アダプタ
- クラウド リソース マネージャのアダプタ

これらのアダプタに加えて、Service Link では Auto-Complete アダプタがサポートされます。

追加アダプタは、Service Link Adapter Development Kit（ADK）を使用してインストールおよび設定できます。[Adapters] ページには、このようなカスタム アダプタもすべて表示され、プロパティを確認できます。カスタム アダプタの作成およびインストールの詳細については、Adapter Development Kit による統合の設計を参照してください。

ここでは、以下のアダプタについて説明します。

Auto-Complete アダプタ

Auto-Complete アダプタを使用すると、エージェントは、送信メッセージを送信し、外部システムからの確認応答を待たずに、タスクを完了としてマーキングできます。送信メッセージが正常に送信される（たとえば、送信ファイル アダプタによって指定されたディレクトリにファイルが書き込まれる）と、auto-complete アダプタは、同じタスクの着信メッセージを生成します。着信メッセージは、メッセージ タイプ「take- action」を持ちます。このメッセージは、通常、Business Engine によって処理され、アクションを done としてマーキングして外部タスクを完了します。

ダミー アダプタ

ダミー アダプタはプレースホルダです。いくつかの処理シナリオで使用できます。

- ダミー アダプタを着信アダプタとして使用すると、Service Link で外部タスクを開始し、「Ongoing」ステータスのままにすることができます。
- ダミー アダプタを発信アダプタとして使用し、auto-complete アダプタを着信アダプタとして使用すると、サービス設計者は外部タスクで Auto-Complete エージェントを実装できます。その後、このタスクをワークフローの一部で使用し、参加者への電子メールを生成したり、他のタスクが含まれない要求を終了したりできます。この組み合わせは、Service Catalog と Service Link 間の通信が正しく機能しているかどうかを確認するためにも使用できます。

データベース アダプタ

データベース（DB）アダプタは、データベース内の 1 つ以上のテーブルを使用して、サービス カタログ（Service Catalog）と外部アプリケーション間でデータを渡します。

データベース接続

受信および送信データベース アダプタは、ANSI 標準 SQL をサポートする任意の JDBC 準拠リレーショナル データベースと通信できます。JDBC URL およびデータベース ドライバとともに、有効な接続基準を指定する必要があります。外部データベースが SQLServer または Oracle の場合、シスコ提供のドライバを使用できます。シスコが提供するドライバには、次のものがあります。

```
com.microsoft.sqlserver.jdbc.SQLServerDriver
oracle.jdbc.OracleDriver
```

JDBC URL の形式は次のとおりです。

```
jdbc:sqlserver://<host>:<port>;databaseName=<db_name>;selectMethod=direct;sendStringParame
tersAsUnicode=true
jdbc:oracle:thin:@<host>:<port>:<SID>
jdbc:oracle:thin:@//<host>:<port>/<service_name>
```

説明:

- dbtype は sqlserver または oracle です
- host は、データベース サーバの名前です
- port はデータベースに接続するポートです。通常は SQLServer の場合は 1433、Oracle の場合は 1521 です
- SQLServer にはデータベース名を指定し、Oracle には SID（システム ID）とサービス名を指定する必要があります

次に、例を示します。

```
jdbc:sqlserver://mysqlserver.cisco.com:1433;databaseName=RequestCenter;selectMethod=direct
;sendStringParametersAsUnicode=true

jdbc:oracle:thin:@myoracle.cisco.com:1521:DEVRC

jdbc:oracle:thin:@//myoracle.cisco.com:1521/PRODRC
```

サポート jar ファイルが Service Catalog ディレクトリ構造のディレクトリ ISEE.war/WEB-INF/lib にインストールされている場合、ユーザ指定のドライバを使用できます

手順 1　適切な他社製 JDBC ドライバを入手します。たとえば、Sybase JDBC ドライバは、Sybase の Web サイトからダウンロードできます。

手順 2　必要な jar ファイルを ISEE.war/WEB-INF/lib フォルダにコピーします。

手順 3　カスタム ドライバと正しい JDBC URL 形式を使用するように Agent 設定を変更します。たとえば、Sybase ドライバ用 JDBC URL の形式は次のとおりです。

```
jdbc:sybase:Tds:host:port/database
```

手順 4　Service Link と Service Catalog サービスを再起動します。

JDBC Url の形式は、Service Link が導入されたアプリケーション サーバによっても異なります。たとえば、WebSphere アプリケーション サーバから SQLServer データベースへの接続を確立する場合、考えられる JDBC URL は次のようになります。

```
jdbc:sqlserver://<host>:<port>;databaseName=<db_name>;selectMethod=direct;sendStringParame
tersAsUnicode=true
```

受信プロパティ

データベース アダプタを着信アダプタとして使用する場合、エージェントのプロパティに、指定したデータベース接続に対して実行される SQL ステートメントを含めます。この SQL は、一般的には、行のセットを返す select コマンドです。次に、これらの行が外部 XML メッセージに書式設定されます。メッセージは、(エージェントで指定される)着信変換によって有効な nsXML 着信メッセージに変換する必要があります。次に、メッセージが Business Engine に渡されます。Business Engine は、受信メッセージに指定されている Channel ID によって識別されるオープン タスクを見つけると、その受信メッセージを処理し、指定されたアクションを実行します。

データベース着信アダプタのプロパティ シートでは、次に示すプロパティ名に「DBInboundAdapter」というプレフィックスが付けられます。

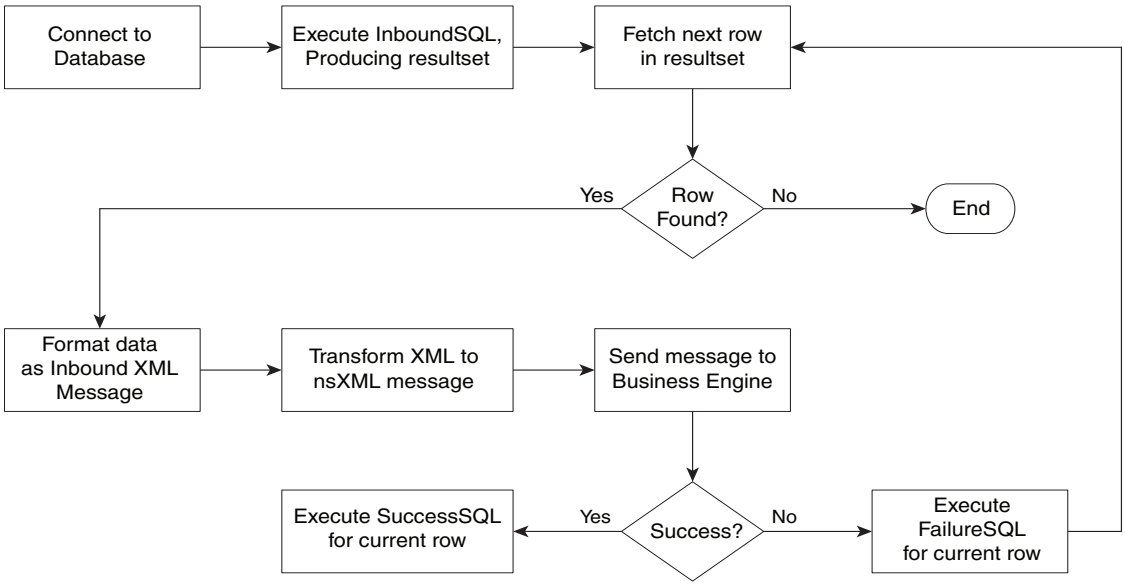
表 5-17　DB アダプタの受信プロパティ

プロパティ	説明
DBPassword	指定されたユーザのパスワード。
DBUserName	データベース ユーザの名前。
InboundSql	受信トランザクションに対して実行される SQL ステートメント。これは、行のセットを返す SELECT ステートメントである必要があります。トランザクション SQL(つまり、プロシージャ)はサポートされていません。
InboundSuccessSql	着信トランザクションの成功時に実行する SQL です。通常は現在の行を正常に処理されたとマークする SQL の update または delete ステートメントです。
InboundFailureSql	着信トランザクションの失敗時に実行する SQL です。通常は現在の行を正常に処理されたとマークする SQL の update または delete ステートメントです。
JDBCUrl	データベースに接続する JDBC URL
JDBCDriverClass	データベースへの接続に使用されるドライバのクラス名。

受信メッセージとワークフロー

受信データベース アダプタのプロセス フローは次のとおりです。

図 5-26　受信データベース アダプタのプロセス・フロー



結果セットの行ごとに、アダプタは次の構造を持つ XML メッセージを生成します。

- メッセージのルート要素は <inbound-results> です。
- 必要な子要素は <row> です。各メッセージは <row> 要素を 1 つだけ持ちます。
- 各 <row> 要素には複数の <column> 要素があり、各カラムの要素は、アダプタに指定された InboundSQL ステートメントに含まれます。
- <row> 要素にはカラム名（<name>）と JDBC データ タイプ（<type>。文字の場合は 12、数値の場合は 1）の属性があります。
- 各 <column> 要素の値は、SQL ステートメントの対応するカラムに返される値です。

たとえば、SQL ステートメント

```
SELECT channel-id, task, status, processType FROM rcInterface
  WHERE status = 'UPDATED'
```

は、次のような XML ストリームを生成する可能性があります。

```
<?xml version='1.0' encoding='UTF-8'?>
<inbound-results>
    <row>
       <column name="channel-id" type="12" >
            "3F2504E0-4F89-11D3-9A0C-0305E82C3301"
       </column>
       <column name="task" type="12" >Task</column>
       <column name="status" type="12" >UPDATED</column>
       <column name="processtype" type="1" >null</column>
    </row>
</inbound-results>
```

次に、変換がこの XML ストリームに適用され、有効な nsXML 着信メッセージが生成される必要があります。たとえば、継続中のタスクを完了する変換に、次のコードが含まれる場合があります。

```
<xsl:template match="/inbound-results/row">
   <xsl:variable name="status"  select="column[@name='status']" />
   <xsl:choose>http://www.w3schools.com/xsl
      <xsl:when test="$status='Complete'">http://www.w3schools.com/xsl
      <message>http://training2.cisco.com/RequestCenter
```

```
      <xsl:attribute name="channel-id">
          <xsl:value-of select="column[@name='channel-id']" />
      </xsl:attribute>http://training2.cisco.com/ServiceLink
      <take-action action="done" />
      </message>
   <xsl:otherwise>
     .       .          .
```

得られた nsXML メッセージを Business Engine が処理します。メッセージが正常に適用された場合は、エージェント内に指定されている SuccessSQL が実行されます。SuccessSQL は、通常、処理で行が選択されたソース テーブル内の列をアップデートするため、次のポーリング間隔で行が再び見つかることはありません。Service Link が現在の行を更新するように指定するには、行の固有識別子を構成するカラムを識別します。これらの列は、受信 SQL ステートメントに含まれる必要があります。次に例を示します。

```
UPDATE rcInterface
   SET status = 'DONE'
 WHERE channel-id = #channel-id#
```

同様に、Business Engine が nsXML メッセージの適用に失敗した場合、たとえば、メッセージの処理中にエラーが発生した場合、FailureSQL が実行されます。FailureSQL は、通常、行が正しく処理されなかったことを示すために、現在の行のステータスをアップデートします。次に例を示します。

```
UPDATE rcInterface
   SET status = 'FAILED'
 WHERE channel-id = #channel-id#
```

送信プロパティ

データベース アダプタを発信アダプタとして使用する場合は、サービス カタログ（Service Catalog）と外部システムの間に「ステージング テーブル」スタイルのインターフェイスを提供します。Business Engine によってエージェントに提供される nsXML 発信メッセージは、1 つまたは複数の SQL ステートメントが含まれる外部メッセージに変換する必要があります。その後、これらの SQL ステートメントは、指定された接続を使用して、指定されたデータベース内で実行されます。

DB 発信アダプタのプロパティ シートでは、次に示すプロパティ名に「DBOutboundAdapter」というプレフィックスが付けられます。

表 5-18　DB アダプタの発信プロパティ

プロパティ	説明
DBPassword	データベース ユーザのパスワード
DBUserName	データベース ユーザ名
JDBCUrl	データベースに接続するための JDBC URL
JDBCDriverClass	データベースに接続するために使用される特定のドライバのクラス名

送信メッセージとワークフロー

XSLT 変換によって生成された送信メッセージは次の形式を持つ必要があります。

```
<?xml version="1.0" encoding="UTF-8"?>
<outbound-message>
    <execute-sql-list>
        <execute-sql> SQLStatement
       </execute-sql>
    </execute-sql-list>
</outbound-message>
```

このメッセージには、それぞれが <execute-sql> タグ内にある複数の SQL ステートメントが含まれます。これらのステートメントは、通常、SQL テーブル内で行の挿入またはアップデートを実行します。アダプタに指定された JDBC ドライバがサポートしている SQL ステートメントを使用できます。SQL ステートメントにはユーザ定義関数を使用できますが、(SQLServer Transact-SQL または Oracle PL/SQL では) ストアド プロシージャはサポートされません。各外部タスクは Channel ID で一意に識別されるため、受信メッセージによってタスクを更新できるように、送信 SQL ステートメントのターゲット テーブルにはチャネル ID に対するカラムが含まれる必要があります。

ファイル アダプタ

ファイル アダプタは、指定されたディレクトリからのファイルの読み取り、または指定されたディレクトリへのファイルの書き込みをサポートしています。

- 複数のディレクトリ、または指定されたディレクトリのサブディレクトリにあるファイルを処理するように、アダプタを設定することはできません。
- 着信ファイル アダプタの起動時に、ディレクトリにある複数のファイルの中で最も古いファイルが処理されます。
- 指定されたディレクトリに対して、ファイルを処理するように設定するエージェントは 1 つだけにする必要があります。
- 指定するディレクトリ (場所) は、Service Catalog がインストールされているか、またはアプリケーション サーバからアクセスできるアプリケーション サーバのファイル システム上にする必要があります。Service Link の処理が進行すると、1 つのディレクトリから別のディレクトリにファイルが移動されるため、すべてのディレクトリが同じ物理デバイス上に存在している必要があります。

ファイル アダプタの受信プロパティ

ファイル アダプタのデフォルト値を持つプロパティは次のとおりです。

ファイル着信アダプタのプロパティ シートでは、次に示すプロパティ名に「FileInboundAdapter」というプレフィックスが付けられます。

表 5-19　ファイル アダプタの受信プロパティ

プロパティ	説明
BackupLocation	FinalResolution または OnError プロパティが「Preserve」の場合、ファイルが処理後にバックアップされる場所。
BackupSuffix	バックアップ ファイルのファイル拡張子。デフォルトは .bak。
FileLocation	読み込まれる着信ファイルがポーリングされる場所(ディレクトリ)。各着信ファイル アダプタに対して一意の場所を使用する必要があります。
FileNameDateFormat	ファイルの日付形式。デフォルトは .yyyyMMddHHmmssSSS。
FinalResolution	トランザクション完了後に、ファイルに対して実行されるアクション。オプションは次のとおりです。 • [Preserve]:ファイルをバックアップ場所に移動します • [Delete]:ファイルを削除します デフォルトは [Preserve] です。
OnError	エラー発生時に、ファイルに対して実行されるアクション。オプションは次のとおりです。 • [Preserve]:ファイルをバックアップ場所に移動します • [Delete]:ファイルを削除します デフォルトは [Preserve] です。
TempLocation	着信ファイルの処理に使用される一時フォルダ。

ファイル アダプタの送信プロパティ

送信アダプタは、指定されたファイルの場所で XML ファイルを生成します。ファイル名には、channel-id が含まれます。これは、エージェントが含まれ、メッセージが作成された外部タスクの固有識別子です。ファイル名は、送信プロパティとして指定された日付形式で終わります。

ファイル発信アダプタのプロパティ シートでは、次に示すプロパティ名に「FileOutboundAdapter」というプレフィックスが付けられます。

表 5-20　ファイル アダプタの送信プロパティ

プロパティ	説明
BackupLocation	発信ファイルのバックアップ場所。アプリケーション サーバからアクセス可能な、任意の有効なファイル システム ディレクトリなどです。
BackupSuffix	バックアップ ファイルのファイル拡張子。デフォルトは .bak。
ConflictResolution	送信ファイル名に矛盾がある場合に実行されるアクション。オプションは次のとおりです。 • [Preserve]:ファイルをバックアップ場所に移動します • [Delete]:ファイルを削除します デフォルトは [Rename] です。
FileLocation	ファイルが書き込まれる場所。アプリケーション サーバからアクセス可能な任意のディレクトリなどです。

表 5-20　ファイル アダプタの送信プロパティ（続き）

プロパティ	説明
FileNameDateFormat	ファイル名の日付形式。デフォルトは .yyyyMMddHHmmssSSS です。
OnError	エラー発生時に、ファイルに対して実行されるアクション。オプションは次のとおりです。 • [Preserve]：ファイルをバックアップ場所に移動します • [Delete]：ファイルを削除します デフォルトは [Preserve] です。
TempLocation	発信ファイルの処理に使用される一時フォルダ。

HTTP/WS アダプタ

HTTP/WS アダプタは、HTTP 要求または Web サービス要求および応答を送受信するために使用します。HTTPS もサポートされます。

Web サービスに接続するプロキシ サーバの使用はサポートされていません。

Web サービスを呼び出すために使用する場合は、同期コールだけが可能です。発信変換は、Web サービスの標準に準拠する外部メッセージを生成するように作成する必要があります。SOAP ベースの Web サービスの場合、適切な SOAP ヘッダーと SOAP 本文要素を含める必要があります。

送信プロパティ

HTTP/WS アダプタの送信プロパティは、送信アダプタの動作を指定します。

http/ws 発信アダプタの HTTP/WS アダプタの送信プロパティ表では、次に示すプロパティ名に「HttpOutboundAdapter」というプレフィックスが付けられます。

表 5-21　HTTP/WS アダプタの送信プロパティ

プロパティ	説明
WsdlURL	実行される操作が含まれた wsdl の URL。Integration ウィザードだけで使用されます。
WsdlOperation	Web サービスによって実行される動作。Integration ウィザードを使用する場合を除いて指定。指定された WSDL に含まれるすべての動作のドロップダウン リストが表示されます。
RoutingURL	ポストされるすべての発信メッセージをルーティングする URL。Web サービスのエンド ポイント。
AcceptUntrustedURL	外部システムからの信頼できない証明書を許容可能にするオプション。デフォルトは true です。
ContentType	コンテンツ タイプ。デフォルトは text/xml; charset=ISO- 8859-1 です。
TimeOut	http url 接続を取得するためのタイムアウト。デフォルトは 180,000 マイクロ秒です。
ProcessResponse	着信メッセージとしての SOAP メッセージのポストまたは応答の結果を処理するオプション。デフォルトは false です。

表 5-21　HTTP/WS アダプタの送信プロパティ（続き）

プロパティ	説明
RequestHeaders	HTTP 要求ヘッダーに含める必要があるカスタム ヘッダー パラメータ（通常は、アンパサンド（&）で区切られた名前と値のペアの形式）。たとえば、SOAPAction は次のような形式で入力します。 SOAPAction=Op SOAPAction やカスタム ヘッダーは「参照元」と呼ばれ、次のような形式で入力されます。 SOAPAction=Op&referrer=www.test.com
AuthenticationScheme	Web サービスを要求するか、URL にポストするために使用される認証のタイプ。オプションは basic、anonymous、digest、または NTLM です。これらのオプションについては、後述の説明を参照してください。
AuthenticationScopeHost	認証クレデンシャルが適用されるホスト。クレデンシャルを任意のホストに適用できる場合は、空白のままにしておくことができます。
AuthenticationScopePort	認証クレデンシャルが適用されるポート。クレデンシャルを任意のポートに適用できる場合は、空白のままにしておくことができます。
AuthenticationScopeRealm	認証クレデンシャルが適用されるレルム。クレデンシャルを任意のレルムに適用できる場合、空白にできます。
[ユーザ名（Username）]	ターゲット システムへの認証のユーザ名。
[パスワード（Password）]	ターゲット システムへの認証のパスワード。
ホスト	一部の認証スキーム（NTLM など）で必要になることがあるホスト クレデンシャル。
ドメイン	一部の認証スキームで必要になることがあるドメイン クレデンシャル。 NTLM では、レルムの概念を使用しません。認証ドメインは、「レルム」属性の値として指定する必要があります。クレデンシャルを任意のドメインに適用できる場合は、空白のままにしておくことができます。
SaveRefField	*ProcessResponse* が true の場合に使用する boolean。外部システムがこのタスクの固有識別子（または TopicID）として使用するフィールドが、応答に含まれていることを示します。詳細については、http/ws 要求への応答を参照してください。
RefFieldXPath	参照フィールド（TopicID）を識別する、応答の XPath 式。
RefFieldPattern	参照フィールドに適用される正規表現。
CancelIdentifierXPath	存在すると、進行中タスクをキャンセルする必要があることを指定する XPath 式。
method	メッセージの送信に適用される HTTP メソッド。サポートされる 4 つのメソッドは、POST、GET、DELETE、PUT です。SOAP ベースの Web サービスでは、必ず POST を使用してください。
AppendUsernamePassword	この設定は RESTful Web サービスのみに使用されます。これを true に設定すると、ユーザ名とパスワードが HTTP 認証のために要求ヘッダーに追加されます。この設定は、シングル サインオンが有効なときには使用しないでください。SSO の場合は、代わりに認証スキームとスコープ パラメータを設定します。

表 5-21　HTTP/WS アダプタの送信プロパティ（続き）

プロパティ	説明
UsernameAlias	ユーザ名に対して要求ヘッダーで使用する引数名。
PasswordAlias	パスワードに対して要求ヘッダーで使用する引数名。
PublicKey	外部システムのパブリック キーを入力します。Service Link を通じて送信される **encrypt** 属性は、エージェントで設定された外部システムのパブリック キーを使用して暗号化されます。 ドロップダウン リストの既存のパブリック キーを使用するか、または、[Service Link] > [統合の管理（Manage Integrations）] > [HTTP アダプタのエージェント（Agents for HTTP adapter）] でエージェントを設定中に HttpOutboundAdapter.PublicKey に新しい公開キーを作成します。 新しいパブリック キーを追加するときには、新しいパブリック キーの係数および指数を必ず入力します。 発信メッセージを配信する必要がある複数の PO インスタンスがあり、それぞれが異なるパブリック キーを持っている場合は、異なるエージェントは異なるセキュア キーで設定する必要があり、それらの各エージェントを異なる外部タスクにマップします。セキュア データ トランザクションの詳細については、機密データの保護を参照してください。

EncryptStringFormat	Prime Service Catalog を Process Orchestrator 以外の外部システムと統合する場合は、暗号化された文字列を JSON 形式または XML 形式で渡す必要があります。
	
	Intelligent Automation 形式を使用する場合、Prime Service Catalog は Process Orchestrator と自動的に統合されます。Intelligent Automation 形式に関する詳細情報については、『Cisco Intelligent Automation for Cloud Documentation』を参照してください。
	
	JSON 形式:
	
	`{"encryptStringFormat":{"initVector":"DCI4Tg==","saltHash":"LUNZWg==","payload":"UGF2YW4gQ29uZmlkZW50aWFsIQ=="}}`
	
	XML 形式:
	
	`<encryptStringFormat><initVector>DCI4Tg==</initVector><payload>UGF2YW4gQ29uZmlkZW50aWFsIQ==</payload><saltHash>LUNZWg==</saltHash></encryptStringFormat>`

認証方式

送信 http/ws アダプタの一部のプロパティは、ある特定の認証方式、おそらく認証がカスタマイズされた Web サーバに対してにのみ必要です。認証方式の表では、送信 http/ws アダプタでサポートされている認証方式を要約します。

表 5-22　認証方式

認証タイプ	説明
Anonymous	ユーザ クレデンシャルの提供を要求する必要はありません。通常は、Web サーバへのアクセスはサービス アカウントで行います。
基本	ユーザ名とパスワードが必要です。パスワードはクリア テキスト形式で送信されます。
ダイジェスト（Digest）	ユーザ名とパスワードが必要ですが、パスワードは MD5 ハッシュで送信されます。
NTLM	Windows 2003 の統合 Windows 認証。
NTLMv2	Windows 2008 以降の統合 Windows 認証。

http/ws 要求への応答

要求が Web サイトにポストされると、あるいはメッセージが Web サービスに送られると、対象サイトでは通常はメッセージ送信者に応答を返します。応答に Service Link にとって有用な情報が含まれている可能性が低い場合、*Process Response* プロパティを false に設定し、Service Link にこのようなメッセージを無視するように指定できます。ただし、このような応答に、外部システムのチケット番号や、Service Catalog で生成されたタスクに割り当てられたケース番号などの、追加情報が含まれていることがあります。この場合、Process Response と Save Ref Field の両方のプロパティを true に設定し、Service Link の Reference フィールドの xpath を指定して、Web サービスの応答から参照をキャプチャできます。さらに、変換を応答に適用し、サービス フォームを外部システムからの情報で更新するアクションを起動できます。

送信プロパティの値にエージェント パラメータを使用する

送信プロパティに、エージェント パラメータ名前空間を含めることができるようになりました。この場合、複数の操作が同じ外部システムにルーティングされ、ルーティング URL セグメントまたは要求ヘッダーの値だけが異なる場合に、単一のエージェントを複数の操作に使用できます。エージェント パラメータ名前空間の構文は $ParamName$ です。

参照フィールドと TopicID

通常、外部システムには独自の手段があり、インシデント、要求、またはその他のオブジェクトが、サードパーティ システムで開かれているか、製品のユーザ インターフェイスで維持されているかを識別します。指定された参照フィールド（TopicID）により、Service Catalog は、外部システムの一意の識別子と Service Catalog の channel-id の間の相互参照を維持することができます。TopicID が Web サービス要求への初回の応答で識別され、保存されると、それ以降に外部システムから Web サービスのリスナー アダプタ経由で受信されるメッセージは、その TopicID を Service Catalog の外部タスクを識別するのに使用できます。

受信プロパティ

- HTTP/WS 着信アダプタはリスナー アダプタで、呼び出しに基づくポーリングはサポートされません。
- 指定の URL に対して HTTP/WS 着信エージェントを 1 つだけ設定する必要があります。http プロトコルまたは https プロトコルのいずれかを使用できます。

受信 http アダプタに指定できるプロパティはありません。すべての http post を Integration Server の URL に転送する必要があります。

<ServerName>:<Port>/IntegrationServer/ishttplistener?channel-id=<channel-id>

説明:

- <ServerName> はサービス カタログ（Service Catalog）アプリケーション サーバです。
- <Port> はサービス カタログ（Service Catalog）が待ち受けるポートです。
- <channel-id> は、着信メッセージの影響を受けるタスクを一意に識別するチャネル ID です。channel-id が実行中のタスクに該当しない場合は、「Error 503（Application Error）」が他社製のシステムに返されます。

Web サービスの起動

Web サービスはそれほど単純ではない「HTTP による XML」です。送信アダプタの場合、XML メッセージは http（または https）によって Web サービスに送信されます。メッセージ（変換のアプリケーションによって作成された発信メッセージ）は、SOAP エンベロープで囲む必要があります。Web サービスに送られる XML メッセージの例は、次のようになります。

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
<soap:Header
  soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <AuthenticationInfo>
    <userName>ns28sbd</userName>
    <password>09rbc19</password>
  </AuthenticationInfo>
</soap:Header>
<soap:Body
  soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <Op>
    <Assigned_To_Group>CSCC</Assigned_To_Group>
    <Case_Type>Problem</Case_Type>
    <Category>Computer/Printer/Server</Category>

<!-- additional tags as required />

    <txt_internalticketid/>
    <txt_requestid >40</txt_requestid >
    <Type>New Hardware Request</Type>
  </Op>
</soap:Body>
</soap:Envelope>
```

JMS アダプタ

JMS 受信アダプタはリスナー アダプタであり、ポーリングをベースとする起動はサポートしていません。

JMS アダプタはキューとの間でメッセージを読み書きしたり、特定のトピックをパブリッシュおよびサブスクライブすることができます。1 つのキューに設定できる JMS 受信エージェントは 1 つだけです。同じエージェントを複数のトピックのサブスクライブに使用することはできません。トピックは、「topic.sample.exported」のように、完全に指定する必要があります。

受信アダプタのプロパティ

JMS 着信アダプタのプロパティ シートでは、次に示すプロパティ名に「JMSInboundAdapter」というプレフィックスが付けられます。

表 5-23　JMS アダプタの着信プロパティ

名前	説明
JndiProviderUrl	JMS で管理される着信エージェント用オブジェクト検索のための JNDI プロバイダーの URL。デフォルトは jnp://localhost:4099 です。
JndiFactory	着信エージェント用の JNDI ネーミング ファクトリ。デフォルトは org.jnp.interfaces.NamingContextFactory です。
JmsTopicFactory	着信エージェント用の JMS トピック接続を取得するための Topic 接続ファクトリ。使用されません。
JmsQueueFactory	着信エージェント用の JMS キュー接続を取得するためのキュー接続ファクトリ。デフォルトは ConnectionFactory です。
MessageMode	JMS の宛先がキューまたはトピックかを指定します。有効な値は Queue または Topic。デフォルトは Queue です。
JmsQueue	着信エージェント用のメッセージ モードがキューの場合の名前付き JMS キュー。
JmsTopic	着信エージェント用のメッセージ モードがトピックの場合の名前付き JMS トピック。
MessageType	受信エージェントのメッセージのタイプ。有効値は Text です。
Publisher.isAdapter	パブリッシャがアダプタの場合。デフォルトは True です。
Listener.UseCallback	コールバックを使用するかどうか。デフォルトは True です。
UserName	着信エージェント用の JNDI セキュリティ クレデンシャルのユーザ名。
[パスワード（Password）]	着信エージェント用の JNDI セキュリティ クレデンシャルのパスワード。

送信アダプタのプロパティ

JMS 発信アダプタのプロパティ シートでは、次に示すプロパティ名に「JMSOutboundAdapter」というプレフィックスが付けられます。

表 5-24　JMS アダプタの発信プロパティ

名前	説明
JndiProviderUrl	JMS で管理される発信エージェント用オブジェクト検索のための JNDI プロバイダーの URL。デフォルトは jnp://localhost:4099 です。
JndiFactory	発信エージェント用の JNDI ネーミング ファクトリ。デフォルトは org.jnp.interfaces.NamingContextFactory です。
JmsTopicFactory	発信エージェント用の JMS トピック接続を取得するための Topic 接続ファクトリ。未使用。
JmsQueueFactory	発信エージェント用の JMS キュー接続を取得するためのキュー接続ファクトリ。デフォルトは ConnectionFactory です。

表 5-24　JMS アダプタの発信プロパティ（続き）

名前	説明
MessageMode	JMS の宛先がキューまたはトピックかを指定します。有効な値は Queue または Topic です。
JmsQueue	発信エージェント用のメッセージ モードがキューの場合の名前付き JMS キュー。
JmsTopic	発信エージェント用のメッセージ モードがトピックの場合の名前付き JMS トピック。
MessageType	送信エージェントのメッセージのタイプ。有効な値は Text です。
Publisher.isAdapter	パブリッシャがアダプタの場合。デフォルトは True です。
UserName	発信エージェント用の JNDI セキュリティ クレデンシャルのユーザ名。
[パスワード（Password）]	送信エージェント用の JNDI セキュリティ クレデンシャルのパスワード。

MQ アダプタ

MQ 受信アダプタは、IBM WebSphere Message Queue（MQ）システムを使用するポーラー アダプタです。このアダプタでは、IBM MQ Series バージョン 5.x 以上をサポートしています。また、組み込み用に IBM MQ Series Java API を使用しています。IBM MQ ソフトウェアはサービス カタログ（Service Catalog）には付属していません。またライセンスは IBM から入手する必要があります。

受信プロパティ

MQ 着信アダプタのプロパティ シートでは、次に示すプロパティ名に「MQInboundAdapter」というプレフィックスが付けられます。

表 5-25　IBM MQ 受信アダプタのプロパティ

名前	説明
ManagerName	IBM MQ Manager の名前
HostName	IBM MQ Server のホスト名
[ポート（Port）]	受信用の IBM MQ Server のポート
UserName	認証用のユーザ名
[パスワード（Password）]	認証用のパスワード
ChannelName	受信メッセージ用の IBM MQ チャネル名
QueueName	受信メッセージ用のキュー名
MsgFormat	受信メッセージのメッセージ形式。デフォルトは Text

送信プロパティ

MQ 発信アダプタのプロパティ シートでは、次に示すプロパティ名に「MQOutboundAdapter」というプレフィックスが付けられます。

表 5-26　IBM MQ アダプタのプロパティ

名前	説明
ManagerName	送信メッセージ用の IBM MQ Manager の名前
HostName	IBM MQ Server のホスト名
[ポート(Port)]	IBM MQ Server のポート
UserName	認証用のユーザ名
[パスワード(Password)]	認証用のパスワード
ChannelName	送信メッセージ用の IBM MQ チャネル名
QueueName	送信メッセージ用のキュー名
MsgFormat	送信メッセージのメッセージ形式。デフォルト値は Text

サービス項目リスナー アダプタ

Web サービス リスナー アダプタ(Web サービス リスナー アダプタを参照)と同様、サービス項目リスナー アダプタは、外部システムで外部タスクへの更新の送信に使用される Web サービス(SOAP)エンド ポイントを提供します。タスクの更新に加えて、このアダプタでは Lifecycle Center サービス項目を着信 SOAP メッセージの一部として作成、更新、および削除できます。また、アダプタにより、サービス項目メタデータとサービス項目インスタンスのデータを取得できます。

外部システムが送信する SOAP メッセージは、「processMessage」処理で起動される必要があります。SOAP 本文内のメッセージ内容は Service Link が認識できるメッセージに変換されてから、操作のタイプに基づいて分離され、それぞれ Business Engine と Service Item Import processor に転送されます。最大 2 つのメッセージが着信 SOAP メッセージの [トランザクションの表示(View Transactions)] ページに表示されます。1 つはタスク更新操作(take-action、add-comments、send-parameters)用で、1 つはサービス項目操作用(create、update、delete)です。後者のメッセージタイプは「Service Item」です。

着信メッセージの認証は、管理モジュールのサイト設定「着信 HTTP 要求認証(Inbound HTTP Requests Authentication)」をオンにすることにより、任意で有効にできます。詳細については、Web サービス リスナー アダプタを参照してください。

受信プロパティ

サービス項目リスナー着信アダプタのプロパティ シートでは、次に示すプロパティ名に「ServiceItemListenerInboundAdapter」というプレフィックスが付けられます。

表 5-27　サービス項目リスナー着信アダプタのプロパティ

プロパティ	説明
Wsdl	現在のインストール済み環境用の Service Catalog 着信サービス項目を記述する wsdl の URL は次の形式です。 <Protocol>://<ServerName>:<Port>/IntegrationServer/webservices/wsdl/ServiceItemTaskService.wsdl 引数の説明 <Protocol> は http または https です。 <ServerName> は Service Link がインストールされているサーバです。 <Port> は Service Link 用に指定された通信ポートです。 次に例を示します。 http://ccp-prod.cisco.com:8089/IntegrationServer/webservices/wsdl/ServiceItemTaskService.wsdl このプロパティは読み取り専用です。これを有効にすると設計者が WSDL を参照できるようになり、サービスの理解や Web サービス クライアントの記述に便利です。
RoutingURL	SOAP メッセージの送信先となる WSDL は次の形式です。 <Protocol>://<ServerName>:<Port>/IntegrationServer/services/TaskService このプロパティは読み取り専用です。外部システム インテグレータが、SOAP メッセージをこの URL に登録するクライアントを作成できるよう、用意されています。

送信プロパティ

サービス項目リスナー アダプタは単方向（着信専用）です。したがって、発信プロパティはありません。

Web サービス リスナー アダプタ

Web サービス リスナー アダプタは、外部システムで外部タスクへの更新の送信に使用される Web サービス（SOAP）のエンド ポイントを提供します。外部システムが送信する SOAP メッセージは、「processMessage」処理で起動される必要があります。SOAP 本文内のメッセージ内容は、Service Link が理解できるメッセージに変換されてから、処理のため HTTP/WS 着信アダプタに転送されます。

Web サービス リスナー アダプタでは、基盤となっている Web サービス リスナーを使用します。着信メッセージの認証は、管理モジュールのサイト設定「着信 HTTP 要求認証（Inbound HTTP Requests Authentication）」をオンにすることにより、任意で有効にできます。有効にした場合、着信メッセージを処理するためには、有効なユーザ名および対応するパスワードを要求ヘッダーに渡す必要があります。必要に応じて [暗号化されたパスワードを受け入れる（Accept Encrypted Password）] 設定を有効にして、暗号化パスワードのみを使用するよう強制することができます。暗号化ユーティリティは、パスワードの暗号化された値を取得するサイト管理者ロールを持つユーザが使用できます。このユーティリティにアクセスするには、次のブラウザ ページを開きます。

http://<server>:<port>/RequestCenter/EncryptedPassword.jsp

受信プロパティ

Web サービス リスナー着信アダプタのプロパティ シートでは、次に示すプロパティ名に「WSListenerInboundAdapter」というプレフィックスが付けられます。

表 5-28　Web サービス リスナー受信アダプタのプロパティ

プロパティ	説明
WsdlURL	現在のインストール済み環境用の Service Catalog 着信 Web サービスを記述する wsdl の URL は次の形式です。 <Protocol>://<ServerName>:<Port>/IntegrationServer/webservices/wsdl/TaskService.wsdl 引数の説明 <Protocol> は http または https です。 <ServerName> は Service Link がインストールされているサーバです。 <Port> は Service Link 用に指定された通信ポートです。 次に例を示します。 http://ccp-prod.cisco.com:8089/IntegrationServer/webservices/wsdl/TaskService.wsdl このプロパティは読み取り専用です。これを有効にすると設計者が WSDL を参照できるようになり、サービスの理解や Web サービス クライアントの記述に便利です。
WsdlRoutingURL	SOAP メッセージの送信先となる WSDL は次の形式です。 <Protocol>://<ServerName>:<Port>/IntegrationServer/services/TaskService このプロパティは読み取り専用です。外部システム インテグレータが、SOAP メッセージをこの URL に登録するクライアントを作成できるよう、用意されています

送信プロパティ

Web サービス リスナー アダプタは単方向（着信専用）です。したがって、発信プロパティはありません。

機密データの保護

Prime Service Catalog には通常、製品内で表示/アクセスされるとき、および外部システムと交換されるときに保護しなければならないデータが含まれます。このため、セキュア データ トランザクション中には暗号化方式を使用できます。

Prime Service Catalog 内における暗号化ディクショナリ属性の詳細については、『Cisco Prime Service Catalog Designer Guide』の「Configuring Dictionaries」を参照してください。

HTTP/WS および AMQP タスク アダプタに基づき、Service Link エージェントを介して送信される暗号化属性は、エージェントで設定される外部システムの 1024/2048 ビット RSA パブリック キーを使用して保護され、暗号化されます。アダプタの Http/WS 送信プロパティ ページで外部システムの**パブリック キー**と **EncryptStringFormat** を設定してから、エージェントのプロパティを設定する必要があります。詳細については、アダプタの管理を参照してください。

たとえば、Process Orchestrator（PO）へ送信される発信メッセージの暗号化属性は、次の方式で変換されます。

- 4 バイトのマジック番号
- 1 バイトの初期化ベクトル（IV）長
- IV
- 2 バイトのソルト長
- PO のパブリック キーを使用した、ソルト保護
- 暗号化ペイロード長 4 バイト
- 暗号化ペイロード

外部システムでパブリック キーが更新された場合、それに従ってエージェントを更新する必要があります。たとえば PO でパブリック キーが更新された場合、Service Link の古いメッセージが引き続き通過できるように、PO は猶予期間として 3 日間、古いキー ペアを受け入れます。3 日の猶予期間の後、Prime Service Catalog のエージェントが更新されなかった場合、PO はエラーを返します。したがって、外部システムでキーの再設定が行われた場合、エージェントではパブリック キーを手動でもう一度再設定する必要があります。

（注）　設定された外部システムのパブリック キーが Service Link のエージェントにない場合、エージェントが HTTPs/SSL プロトコルを使用する場合に限り、保存および暗号化されるように定義されたディクショナリ フィールド属性が復号化され、これらの属性の平文値が発信メッセージを介して送信されます。SSL が設定されていない場合、ディクショナリ フィールド属性は、暗号化形式のみで送信されます。

（注）　Prime Service Catalog は AES アルゴリズムおよび 128/256 ビット対称キーをサポートしています。したがって、Service Catalog のインストール中に Java 開発キット（JDK）に「Java Unlimited Strength Crypto Policy」ファイルを手動で必ずインストールしてください。Java Unlimited Strength Crypto Policy ファイルにより、外部システムが 2048 ビット暗号化を実行できるようになります。Unlimited Strength Crypto Policy をインストールしない場合、インストーラにより、サーバの起動中にエラー メッセージが表示されます。Unlimited Strength Crypto Policy ファイルのインストールの説明については、Oracle の Web サイトを参照してください。Cisco Prime Service Catalog のインストールに関する情報については、『Cisco Prime Service Catalog Installation and Upgrade Guide』を参照してください。

クラウド リソース マネージャのアダプタ

クラウド リソース マネージャ アダプタは、Service Link から UCSD への送信通信をサポートします。クラウド リソース マネージャ アダプタは、着信ポーラーがアクティブ化される頻度を決定する、ポーリング間隔属性をサポートします。

（注）　クラウド リソース マネージャのアダプタは事前設定されて提供されており、デフォルト設定を使用してこのアダプタを導入することが推奨されます。

Service Designer での [統合（Integration）] ウィザードの使用

[統合（Integration）] ウィザードでは、Service Catalog と SOAP Web サービスとの統合の作成に関する手順の多くが自動化されます。Integration ウィザードは、Web サービスの統合によって起動される WSDL や処理を収集することによって動作します。[統合（Integration）] ウィザードでは、統合の定義に基づいて、統合のサポートに必要なすべてのコンポーネントを表示します。

- 外部タスクで統合の実行に使用できる Service Link エージェントが作成され、現在のサービスの配信計画で参照されます。
- Web サービスで必要な SOAP メッセージに nsXML を変換する変換が作成され、エージェントの発信アダプタで参照されます。
- 最初の Web サービス要求と応答の両方で必要な、すべてのデータのエージェント パラメータがエージェント定義に追加されます。
- エージェント パラメータ値を格納するフィールドが含まれたディクショナリが作成され、ディクショナリ フィールドが、対応するエージェント パラメータにマッピングされます。
- ディクショナリが含まれたアクティブ フォーム コンポーネントが作成され、現在のサービスに含まれます。

Integration ウィザードでは、認証方式や統合の動作の定義にいくつかのデフォルト オプションを使用します。これらの設定が適切ではない場合、または統合の作成後に変更する必要がある場合、Service Link の [Manage Integration] ページで使用できる高度なコンフィギュレーション オプションを使用してエージェント定義を編集できます。

Integration ウィザードは、Service Link エージェントと変換の作成が可能なロールを付与された、サービス設計者だけが使用できます。

Integration ウィザードからアクセスする wsdl は、Web Services Operability（WS-I）のベスト プラクティスに従う必要があります。

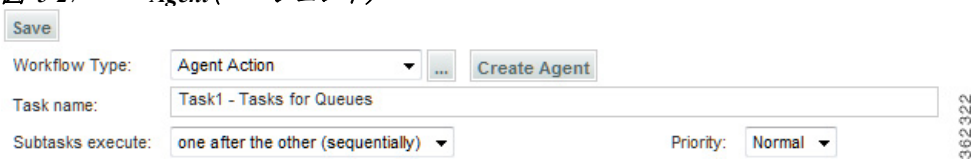
Integration ウィザードを使用する手順は、以下のとおりです。

手順 1　Service Designer でサービスを編集します。

手順 2　そのサービスの [計画（Plan）] タブにある [一般（General）] サブタブを開きます。タスクに関連する他のデータを必要に応じて入力します。

手順 3　[エージェント（Agent）] をクリックします。

図 5-27　Agent（エージェント）

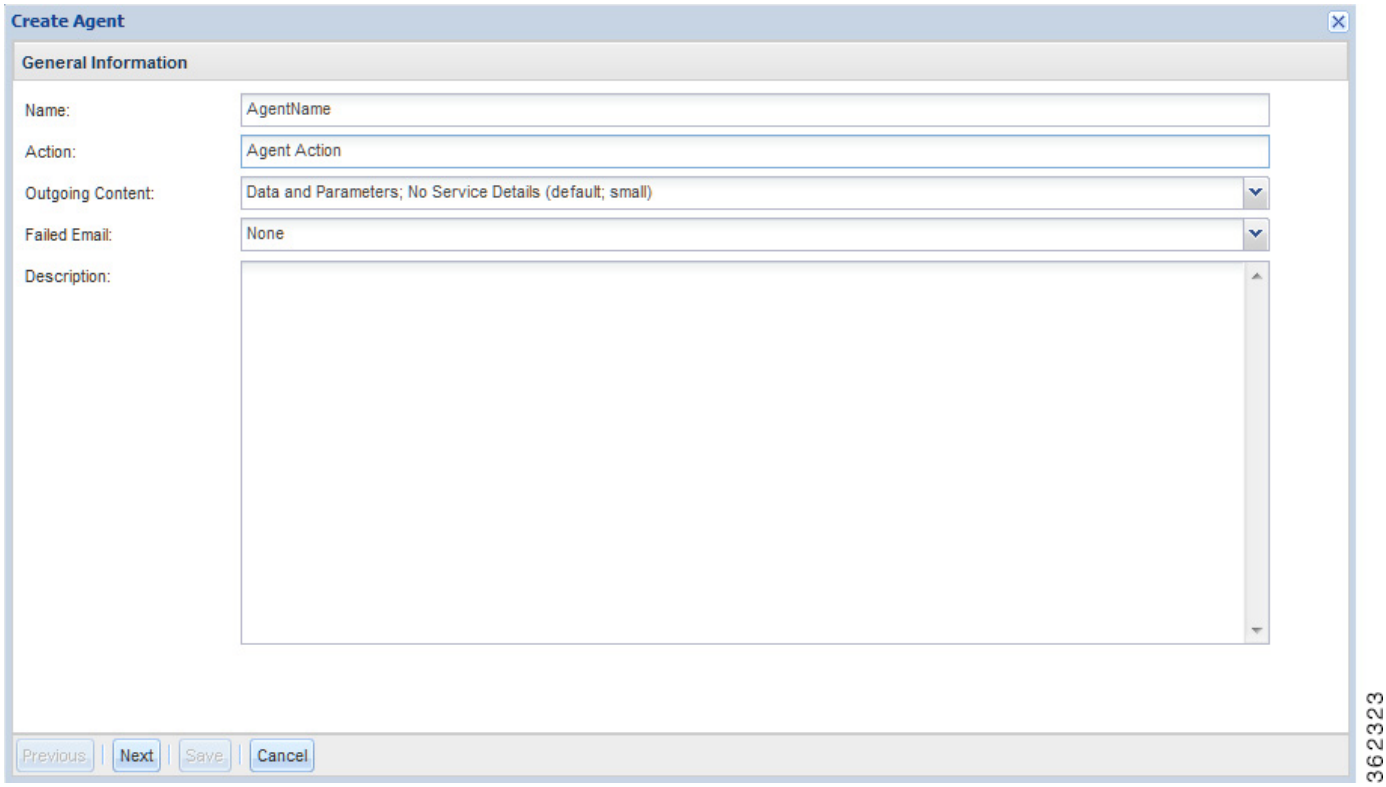


[統合（Integration）] ウィザードの最初のページが表示されます。ウィザードは、エージェントの設定に応じて、最大 8 枚のページで構成されます。各ページに入力したら、[Next] をクリックして次のページに進みます。また、[Previous] をクリックすると前のページに戻ることができます。入力が完了したら、[Save] をクリックしてエージェントの定義（およびその他の設計コンポーネント）を保存します。保存しないで作業を終了する場合は、[Cancel] をクリックします。

全般情報（General Information）

最初に、エージェントに関する一般的な情報を指定します。

図 5-28　Create Agent



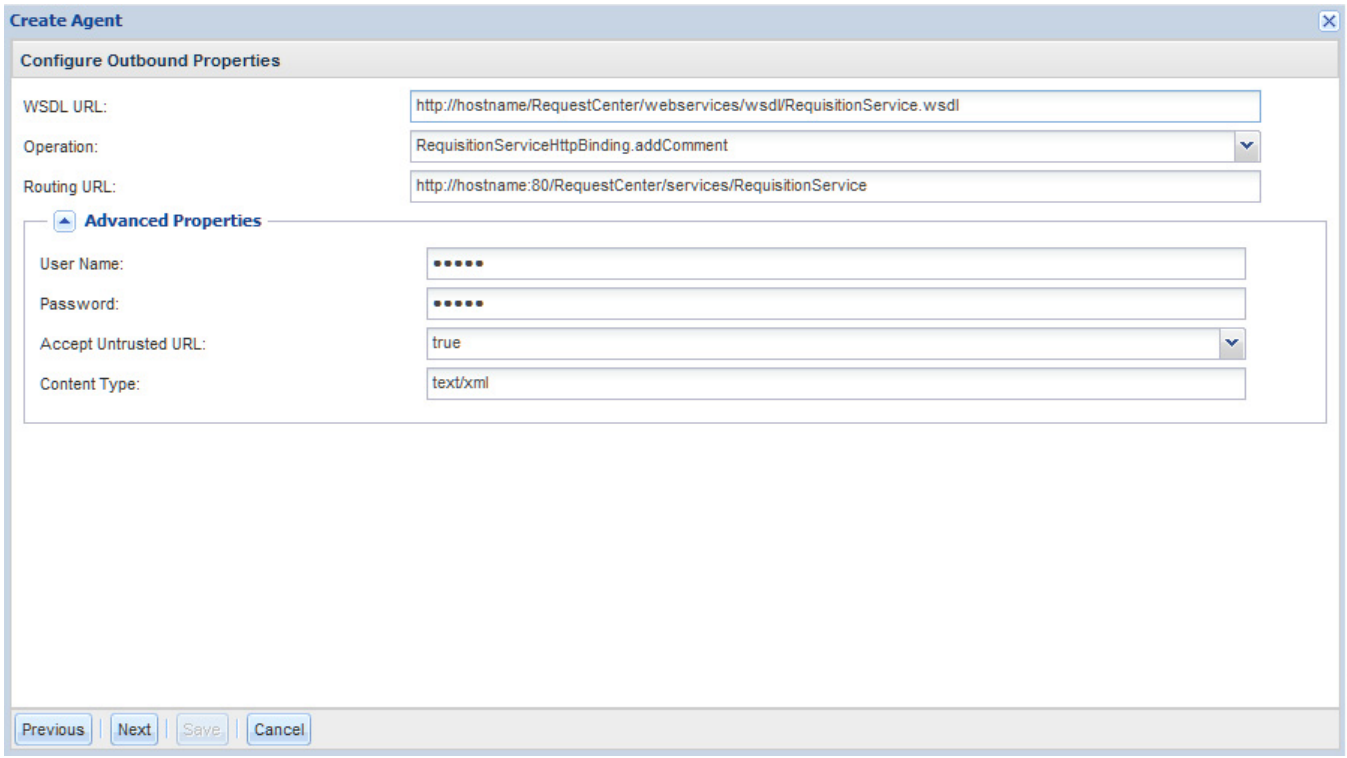
作成するディクショナリとアクティブ フォーム コンポーネントは、エージェントと同じ名前にします。ディクショナリに対する命名基準はエージェントに対する命名基準よりも厳しいため、エージェント名には英文字、数字、およびアンダースコアだけを使用します。また、数字から始まる名前は使用できません。

この画面の他のすべての設定は、Service Link の [Agents] ページで使用できる設定と対応しています。

[Next] をクリックして、ウィザードの次のページに進みます。

送信プロパティ

図 5-29　　エージェントの発信のプロパティの作成



統合で実行される処理が含まれる WSDL の場所を入力します。これは通常は WSDL がある場所の URL になります。

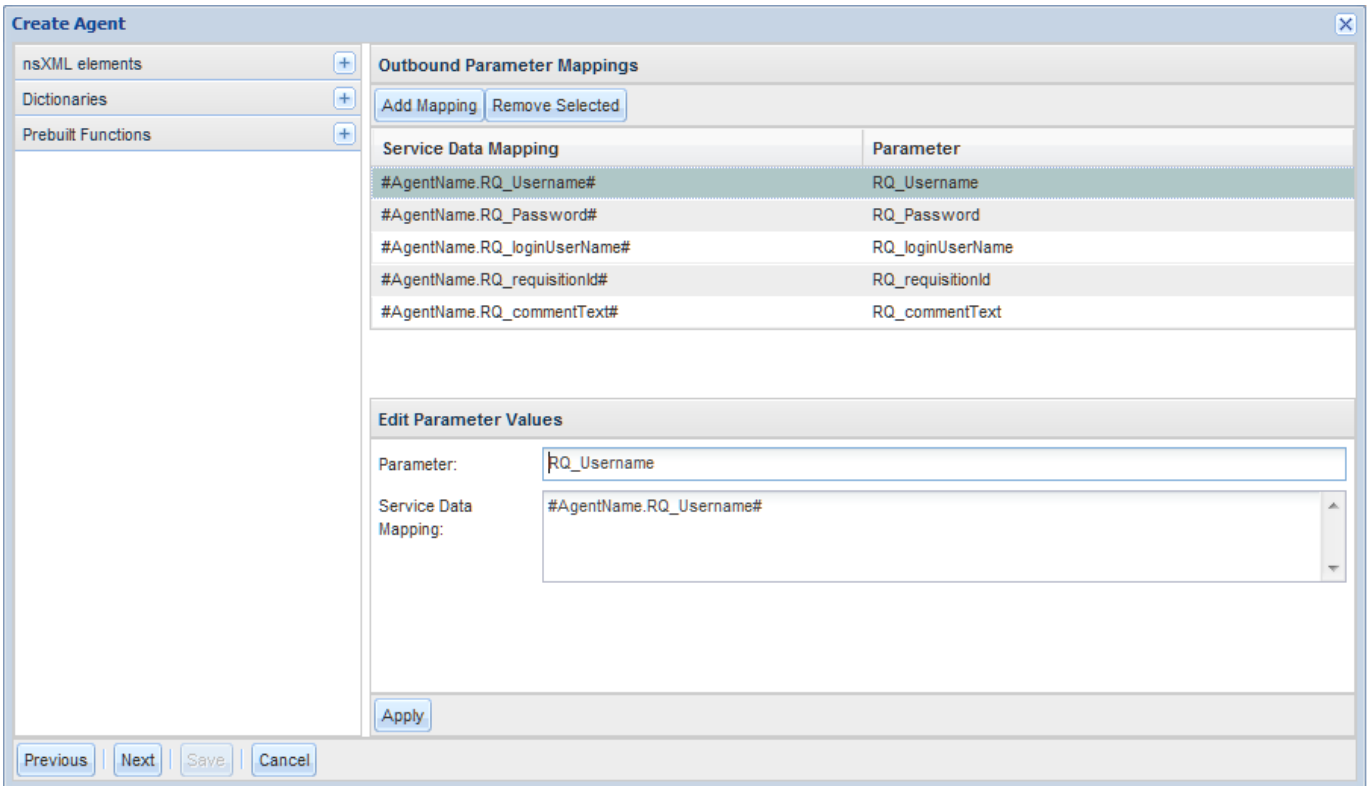
Integration ウィザードによって、ウィザードが読み込まれ、サポートされている処理のリストが表示されます。任意の処理を選択します。その処理に指定されている属性によって、このウィザードの後続のページに表示されるエージェント パラメータの定義が変わります。

WSDL にルーティング URL が含まれている場合は、これも表示されます。

必要に応じて、[詳細プロパティ(Advanced Properties)] ドロップダウン ボタンをクリックして、統合に関するその他の設定を表示します。これらはここで入力することも、Service Link から後で指定することもできます。ウィザードから指定できるのは、基本認証だけです。

送信要求パラメータのマッピング

図 5-30　送信要求パラメータのマッピング



以下の図で示すように、ウィザードによって WSDL が解析され、Web サービスの送信要求で使用する必要のある属性が 2 つ含まれることが判断されます。そのため、エージェント パラメータが 2 つ作成されます。これらの名前は WSDL 内の属性の名前と同じになります。

これらのエージェント パラメータはディクショナリ フィールドにマッピングされます。フィールド名は WSDL 内の属性の名前と同じになり、ディクショナリ名はエージェント名と同じになります。このディクショナリは、エージェントを保存したときに自動的に作成されます。

必要に応じて、以前に Service Designer で定義したディクショナリおよびフィールドを参照するように [Service Data Mapping] を変更します。この方法で、エージェント パラメータのマッピングを効率的に変更できます。しかし、このウィザードによって作成されるディクショナリには、元のフィールドも引き続き保存されています。これを削除するには、ディクショナリの定義を編集します。

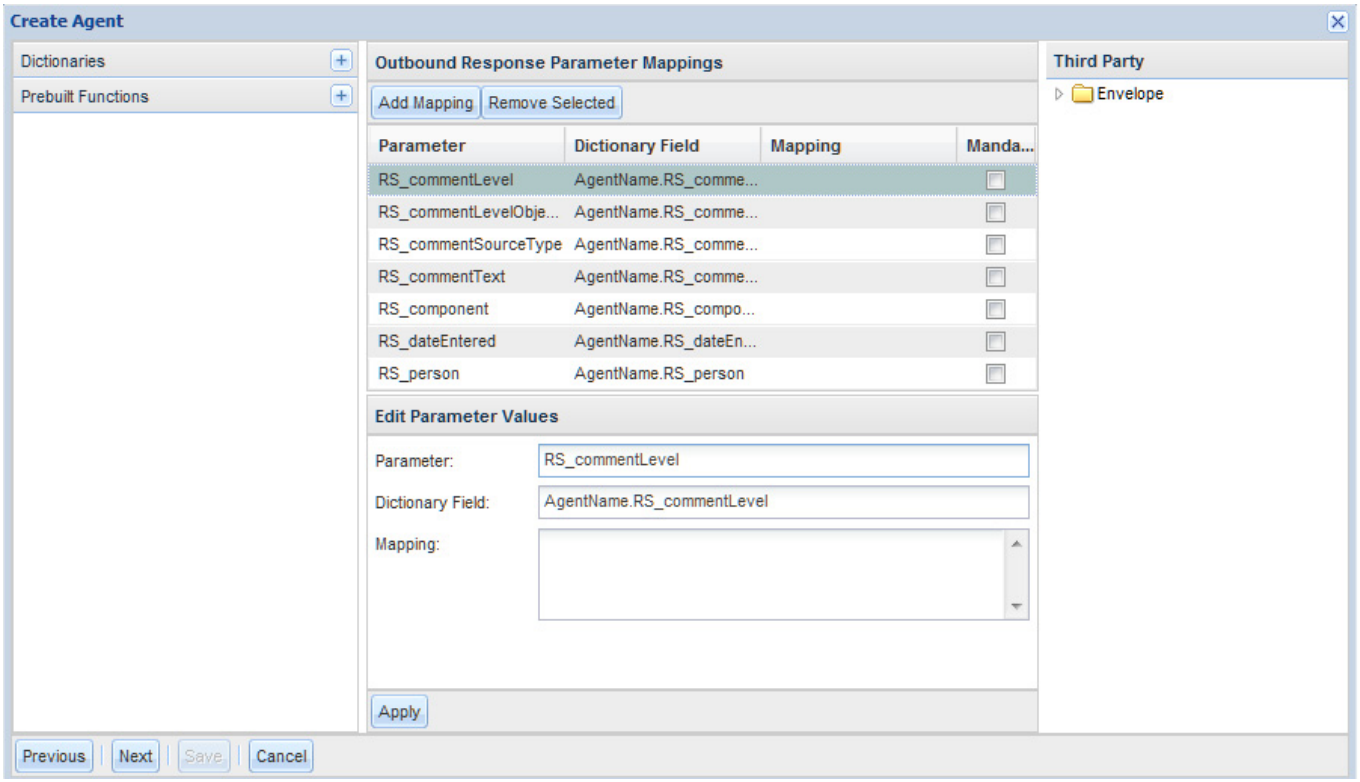
サービスにおけるディクショナリの構造や使用方法について、ここで簡単に説明します。Service Catalog ディクショナリの主な目的は、サービス フォームでユーザに表示するデータを構築することです。したがって、サービス設計者は通常はユーザ インターフェイスを念頭に置いてディクショナリを設定し、カスタマーとサービス チームのメンバーの両者にとって使いやすいようフィールドをグループ化および配置するようにします。

一般的には、送信メッセージには、多数のディクショナリのフィールドに入力された(あるいはデフォルトまたは計算されたもの)データを含めなければならない場合があります。ただし、これによってエージェント パラメータ マッピングの維持がより複雑になり、エラーが発生しやすくなります。統合の設計者はサービス フォームおよびディクショナリの設計についての詳しい知識を持っている必要があります。そのために、統合データを含める目的で、ディクショナリの作成を単独で練習することを推奨します。ディクショナリのフィールドの中には、サービス フォームに表示されるフィールドと重複するものがある場合があります。このような場合、サービス設計者は、表示されているディクショナリから統合ディクショナリにフィールドの値をコピーするための条件をルールとして設定する必要があります。また、統合ディクショナリはサービス フォームに表示されませんが(通常はアクティブ フォーム ルールによって非表示)、デバッグしやすくするために開発中は表示したままにすることができます。

送信応答パラメータのマッピング

デフォルトでは、Integration ウィザードでは、対象システムから受信した応答を処理すると仮定しています。応答で送信される属性には、対応するエージェント属性があり、これはディクショナリ フィールドにマッピングされています。

図 5-31　送信応答パラメータのマッピング



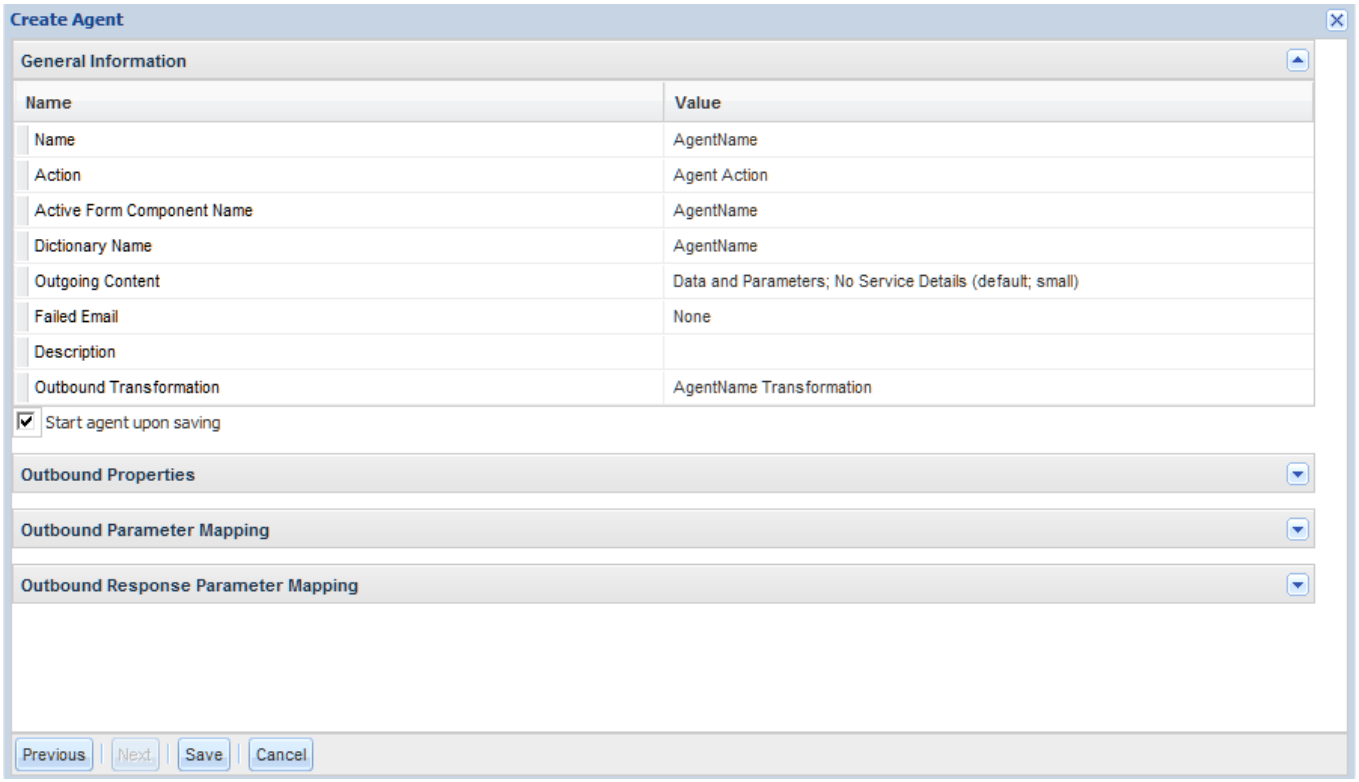
エージェントとフィールドの対応に加えて、マッピングには、ページの左側にある [Prebuilt Functions] ドロップダウンの矢印から指定できる単純な XSLT 処理も含めることができます。

送信パラメータについては、受信パラメータを代わりのディクショナリ フィールドにマッピングすることもできます。ページの左側にある [Dictionaries] ドロップダウン矢印からすべてのディクショナリを参照できます。

統合の要約

ウィザードの最後のページでは、定義した統合の要約が表示されます。定義を修正する場合は、前のページに戻ることができます。[保存(Save)] をクリックすると、作成したエージェントや他の統合コンポーネントを保存できます。デフォルトでは、統合を保存したときにエージェントが開始されます。この動作を変更するには、[保存時にエージェントを開始(Start agent upon saving)] チェック ボックスのチェックを解除します。

図 5-32　統合の要約



Service Link および Service Designer の画面からすべてのコンポーネントを編集できます。これらのコンポーネントを統合コンポーネントの表に示します。

表 5-29　統合コンポーネント

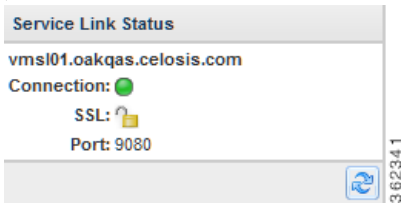
統合コンポーネント	説明
Agent(エージェント)	http/ws 発信アダプタを使用するエージェントと、エージェント パラメータおよび関連する変換。
トランスフォーメーション	発信 nsXML を、選択された WSDL 操作のために目的の形式に変換するために必要な変換。
ディクショナリ	すべての送信および受信パラメータに対応するフィールドが含まれるディクショナリ。ディクショナリが Integration グループに作成されます。必要に応じて移動できます。
Active Form Component	作成したディクショナリを含むアクティブ フォーム コンポーネント。フォーム コンポーネントが統合グループに作成されます。これは必要に応じて移動できます。

Service Link のトラブルシューティングと管理

Service Link のステータスの確認

Service Link の動作ステータスは、最初に [Service Link Status] 表示を確認します。Service Link のステータスは、常に Service Link のホーム ページの [Common Tasks] 領域の下に表示されます。

図 5-33　Service Link Status



この機能は、Service Link が、割り当てられたポートを介して Service Catalog サービスとの通信を行っていることを確認するために役立ちます。

[Service Link Status] は Service Link の接続ステータスが動作していることを示し、使用されるポートおよびプロトコルを示します。

エージェントの起動と停止

Service Link のエージェントを停止し、エージェントのプロパティを変更したら、これが機能するためにはエージェントを再起動する必要があります。エージェントは、[Control Agents] ページから個別に起動および停止できます。

（注）　Service Link サービスを停止して再起動すると、サービスが停止されたときに実行中だったすべてのエージェントが自動的に再起動されます。

ロギング

すべてのアダプタはそのアクティビティをサーバのログ ファイルに記録します。

さらに、Service Link\log ディレクトリに各標準アダプタ専用のログ ファイルがあります。ログに書き込む詳細度のレベルは設定できます。この方法は、アプリケーション サーバによって異なります。

サーバとアダプタ両方の固有のログ ファイルを管理する詳細については、『*Cisco Prime サービス カタログ（Service Catalog）管理および操作ガイド*』を参照してください。

JBoss のロギング

JBoss のインストールでは、「*<JBOSS_DIR>*\standalone\configuration」ディレクトリの logging.properties ファイルを変更することにより、Service Link アダプタのログをサーバ ログから別のファイルに分離できます。このような設定の例は、製品パッケージの「\preinstall\jboss\templates」ディレクトリにあるサンプル プロパティ ファイルにあります。

WebSphere のロギング

デフォルトでは、Service Link の WebSphere ロギングは、WebSphere アプリケーション サーバに含まれている log4j に基づいて行われます。WebSphere での log4j の実装は強力であり、管理コンソールや他のツールから設定できます。しかし、ログ ファイルを容易に分離することはできません。WebSphere でアダプタごとにログ ファイルを分離する場合は、次の手順に従ってください。

手順 1　「ISEE.war/WEB-INF/classes/config」で newscalelog.properties ファイルを見つけて、エディタで開きます。

手順 2　次の行をコメント解除します。

```
logger.class.name=com.newscale.bfw.logging.LogUtilCommonsImpl
```

手順 3　logger.directory の行を見つけます。たとえば、次のようにログ ディレクトリを指定します。

```
logger.directory=I:/logfiles/servicelinkserver2
```

これらのログ ファイルが存在する有効なディレクトリを入力すること、および IntegrationServer を実行するために使用されるユーザは、そのディレクトリに対する完全な書き込みアクセス権があることが非常に重要です。

手順 4　「ISEE.war/META INF/」ディレクトリで、「services」という名前のフォルダを手動で作成します。

手順 5　このサービス フォルダの下に、「org.apache.commons.logging.LogFactory」という名前のテキスト ファイルを手動で作成します。ファイル内で、次のように 1 行追加します。

```
org.apache.commons.logging.impl.Log4jFactory
```

メッセージの消去

Service Link メッセージでは大量のデータベース容量が使用される可能性があり、それらは一度要求が完了すると参照されなくなるため、データベース サイズを縮小するために定期的にデータベースからメッセージをパージすることにメリットがあります。パージ ユーティリティは、Administration モジュールの [ユーティリティ(Utilities)] タブでアクセスできます。ユーティリティは、メッセージ レコード エントリを実際に削除するわけではありません。代わりに、指定された保存期間よりも古いすべての完了または失敗したメッセージについて、メッセージの XML コンテンツを「Message has been purged」と置き換えます。詳細については、『Cisco Prime Service Catalog Administration and Operations』の「Purge Utilities」の項を参照してください。

アプリケーション サーバのコンフィギュレーション ファイル

トラブルシューティングを行う際には、以下のファイルを分析します。

- **rcjms.properties ファイル** - このファイルには、統合の発信 JMS キューに関する情報が含まれ、「/RequestCenter.war/WEB-INF/classes/config」ディレクトリにあります。ビジネス エンジンは、このファイルで指定されているキューにメッセージを書き込みます。次のプロパティの値は、ISEE.war ファイルの integrationserver.properties ファイルのプロパティの値と一致する必要があります。

  ISEEOutbound.JndiProviderUrl

  ISEEOutbound.JndiFactory

ISEEOutbound.JmsTopicFactory

ISEEOutbound.JmsQueueFactory

ISEEOutbound.JmsQueue

ISEEOutbound.JmsTopic

- **integrationserver.properties ファイル** – このファイルには発信および着信 JMS キューに関する情報が含まれ、「/ISEE.war/WEB- INF/classes/config」ディレクトリにあります。このフォルダで指定されている JMS プロパティを確認します。
- **newscale.properties** ファイル: このファイルには **isee.base.url** のプロパティが含まれています。それが、Service Link サーバの url を指していることを確認します。

オンライン エラー ログ

サーバのログ ファイルおよびアダプタに固有のログ ファイルに加えて、Service Link によって検出されたエラーをオンラインで参照することもできます。[Messages] ページに表示されるエラー メッセージのメッセージ テキストは、そのメッセージについての詳細なエラーの説明にハイパーリンクされています。

エラー メッセージは、サーバ ログに表示されるそのものであり、非常に技術的な内容である場合があります。エラー メッセージの例と説明を以下に示します。

```
com.newscale.is.core.RoutingException: Routing exception found: Reference Field not
retrieved from response
```

送信 Web サービスのメッセージは送信されたものの、指定された参照フィールドが応答メッセージにないため、受信応答を処理できませんでした。

```
com.newscale.is.core.TransformationException: javax.xml.transform.TransformerException:
javax.xml.transform.TransformerException: Tag is not allowed in this position in the
stylesheet!
```

変換によって、不正な XML メッセージが生成されました。

事前作成済み関数

事前作成済みの関数は、nsXML メッセージに含まれているエージェント パラメータの値を操作する機能を提供します。

事前作成済みの関数は、Visigoth Software Society によって開発されたオープン ソース ソフトウェアとして入手可能な FreeMarker テンプレート エンジン、バージョン 2.3.12 を使用して開発されています。シスコは以下で説明する関数だけを認定しており、エージェント パラメータの作成時にドロップダウン リストから使用できます。FreeMarker のフレームワークでサポートされている他の関数も使用できますが、広範囲に及ぶテストが必要です。

関数の使用法

基本的な関数の使用法は、式に関数を適用し、必要に応じて関数に引数リストを指定するというものです。一般的な表記は次のとおりです。

```
${Expression?function(argumentList)}
```

Service Link の場合、式は一般的に、Lightweight 名前空間構文によって指定されたディクショナリ フィールドまたは nsXML 要素になっています。次のように、引用符で囲みます。

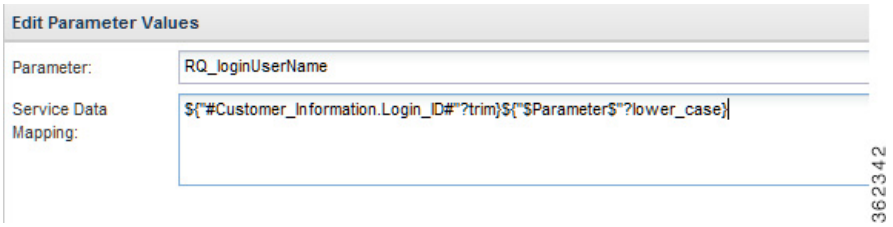
```
${"#Customer_Information.Login_ID#"?upper_case}
```

次の構文を使用して、同じ式に 2 つ以上の関数をチェーンにして適用することができます。

```
${Expression?function1(argumentList)}${"$Parameter$"?function2}
```

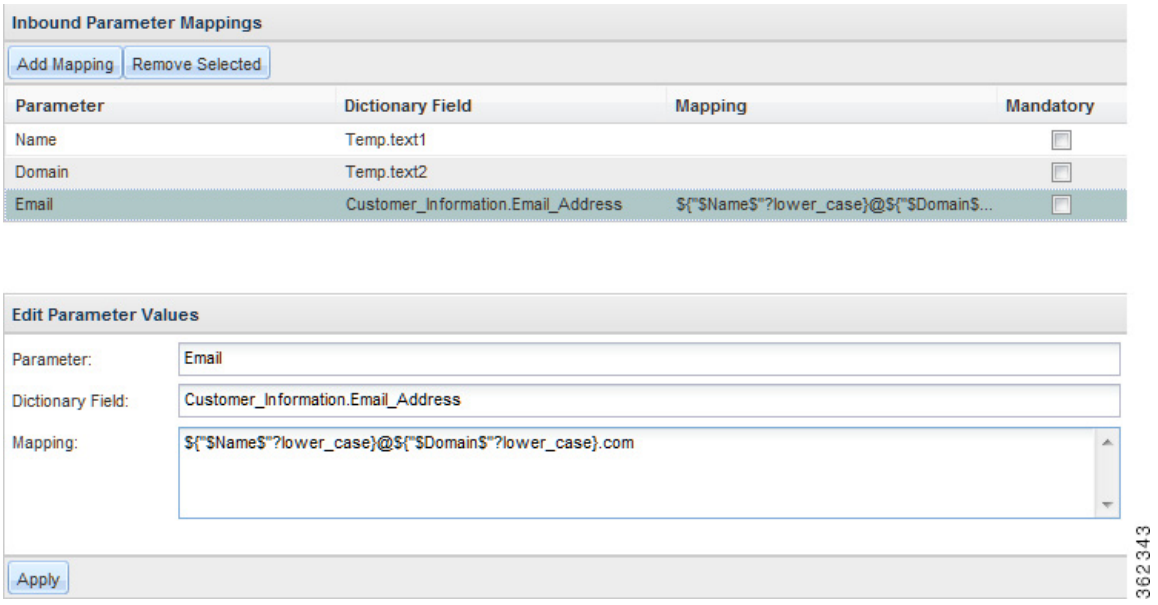
たとえば、以下のサービス データ マッピングでは、まず指定したディクショナリ フィールドから先頭と末尾のスペースを削除(trim)して、次にその結果を小文字に変換しています。

図 5-34　パラメータ値の編集



また、以下のように、複数の要素を 1 つのマッピングに結合することもできます。要素は自動的に連結されて、1 つの文字列を形成します。

図 5-35　着信パラメータのマッピング



このシナリオでは、「一時」フィールドを使用して入力値を格納し、マッピング式で使用できるようにする別のコーディング テクニックも示します。

関数の概要

substring

substring 関数の構文は、次のとおりです。

```
exp?substring(from, toExclusive), also callable as exp?substring(from)
```

文字列の中の部分文字列を取り出します。*from* は、先頭の文字の位置を表します。これは 0 以上、*toExclusive* 以下の数値である必要があります。これ以外の数値を指定すると、エラーによってテンプレートの処理が中断されます。*toExclusive* は、取り出す部分文字列の最後の文字の次の場所を表します。言い換えると、最後の文字より 1 つだけ大きい数値を指定します。この値は、0 以上、文字列の長さ以下の数値である必要があります。これ以外の数値を指定すると、エラーによってテンプレートの処理が中断されます。*toExclusive* を省略すると、デフォルトとして文字列の長さが使用されます。整数値でないパラメータを指定すると、その数値の整数部分だけが使用されます。

index_of

この文字列内の、指定された部分文字列の最初のインデックスを返します。たとえば、*"abcabc"?index_of("bc")* は 1 を返します（1 文字めのインデックスは 0 です）。また、検索を開始するインデックスを指定することもできます。*"abcabc"?index_of("bc", 2)* は 4 を返します。2 つ目のパラメータの数値に制約はありません。マイナスの値を指定した場合は、0 を指定した場合と同じ結果になります。また、この文字列の長さよりも大きな数値を指定した場合は、文字列の長さと同じ数値を指定した場合と同じになります。小数値は整数値に切り捨てられます。

1 つ目のパラメータがこの文字列の部分文字列として見つからない場合は（2 つ目のパラメータを指定した場合は、その位置以降で見つからない場合）、-1 が返されます。

last_index_of

この文字列内の、指定された部分文字列の最後（最も右側）のインデックスを返します。返される位置は、部分文字列の先頭（左側）の位置です。たとえば、*"abcabc"?last_index_of("ab")* は 3 を返します。また、検索を開始する位置を指定することもできます。次に例を示します。

```
"abcabc"?last_index_of("ab", 2)
```

0 が返されます。2 つ目のパラメータは、部分文字列が始まる最も大きなインデックスを表すことに注意してください。2 つ目のパラメータの数値に制約はありません。マイナスの値を指定した場合は、0 を指定した場合と同じ結果になります。また、この文字列の長さよりも大きな数値を指定した場合は、文字列の長さと同じ数値を指定した場合と同じになります。小数値は整数値に切り捨てられます。

1 つ目のパラメータがこの文字列の部分文字列として見つからない場合は（2 つ目のパラメータを指定した場合は、その位置よりも前で見つからない場合）、-1 が返されます。

length

文字列内の文字数を返します。

lower_case

文字列を小文字に変換します。たとえば、「*GrEeN MoUsE*」は「*green mouse*」になります。

replace

元の文字列内のあるパターンを別の文字列で置き換えるために使用します。単語の境界は考慮されません。次に例を示します。

```
${"this is a car acarus"?replace("car", "bulldozer")}
```

次のように置き換えられます。

```
 this is a bulldozer abulldozerus
```

置き換えは、左から右に向かって実行されます。つまり、次のように指定した場合は、

```
${"aaaaa"?replace("aaa", "X")}
```

次のように置き換えられます。

```
    Xaa
```

最初のパラメータが空の文字列の場合、空の文字列がすべて置き換えられます。たとえば、*"foo"?replace("","|")* は "|f|o|o|" になります。

replace は、3 つ目のパラメータとして、**フラグ パラメータ**も指定できます。

upper_case

文字列を大文字に変換します。たとえば、「*GrEeN MoUsE*」は「*GREEN MOUSE*」になります。

Adapter Development Kit による統合の設計

Service Link アダプタを開発するために、Service Link Adapter Development Kit（ADK）を使用できます。ADK は Service Catalog の Service Link サブシステム用のアダプタを作成できるコンポーネント セットです。Service Link は、Service Catalog に外部との通信を提供し、ワークフローを外部に公開して他のシステムと連携することができます。

この通信を実現するために、Service Link はインストール可能なアダプタをサポートしています。標準的なアダプタが Service Link に付属していますが、開発者が別途作成することも可能です。この章では、アダプタを作成するプロセスについて説明します。

この章の対象者は次のとおりです。

- **管理者。**管理者は、製品パッケージを使用できるため、サービス カタログ（Service Catalog）製品をカスタマー システムにインストールできます。
- **アダプタ開発者。**アダプタ開発者とは、ANT を含む Java テクノロジーに精通した人であり、要求されている統合についての専門家です。

カスタム アダプタの作成に関するサポートについては、*Cisco TAC* に連絡してください。

使用する前に

ここでは、ADK のインストール、ADK の構造、アダプタのコンパイル、およびアダプタの導入について説明します。

JDK のインストール

Sun または IBM の説明に従って、Java Development Kit をインストールします。サービス カタログ（Service Catalog）で動作が保証されているのは、JBoss 7.1.1 のインストール環境では Sun JDK 6、WebSphere 7.0.0.17 のインストール環境では IBM Java 1.6 です。

ADK のインストール

ADK のインストール方法は次のとおりです。

1. **管理者**：製品パッケージのコンテキストを展開し、image/isee/dist フォルダにある adk.zip を探し、その場所をアダプタ開発者に知らせます。
2. **アダプタ開発者**：ファイル adk.zip（または adk.tar.gz）を管理者から入手します。
3. **アダプタ開発者**：ADK パッケージをローカル マシン（C:\ADK）で展開します。C ドライブにする必要も、ADK ディレクトリにする必要もありません。ただし、この章の例では C:\ADK です。

ADK 構造

ADK をインストールすると、次のサブディレクトリが作成されます。

表 6-1　ADK のサブディレクトリ

ディレクトリ	説明
<root>	ビルド プロシージャ ファイルが格納されます。
ant	ANT ビルド システム一式が格納されます。この ANT は標準の Apache ANT ビルド システムであり、追加の拡張がいくつか含まれています。
doc	java doc サブディレクトリが格納されます。ADK ドキュメントをここに配置できます。
doc\javadoc	javadoc 形式の ADK のヘルプ。
example	アダプタのサンプルが格納されます。
example\src	サンプル アダプタのソース ファイル Java ファイルが格納されます。
example\deploy	そのアダプタについて記述された adapter.xml が格納されます。
lib	コンパイルに必要な ADK ライブラリが格納されます。

アダプタは ADK 主構造のサブディレクトリです。インストール後の **example** も、そのようなアダプタの 1 つです。アダプタのコードは、次のような構造にする必要があります。

表 6-2　アダプタ コード構造表

ディレクトリ	説明
<adapter>	アダプタの短縮名。サンプル アダプタの場合は **example** です。
<adapter>\src	必須。ソース Java ファイルのルート。
<adapter>\deploy	必須。導入ディレクトリ。ここには、**adapter.xml** というファイルが格納されている必要があります。
<adapter>\lib	これはオプションです。**ISEE.war** の lib ディレクトリに追加する、追加ライブラリ。
<adapter>\config	これはオプションです。**ISEE.war** のクラス ディレクトリにコピーされる追加ファイル。

この例では、**src** および **deploy** だけです。

ファイルをコンパイルした後、ステージング ディレクトリを作成します(アダプタの作成方法については、次の項を参照してください)。ステージング ディレクトリは、作成手順を使用して後から削除および再作成できます。

表 6-3　　ディレクトリの説明表

ディレクトリ	説明
staging	実稼働ディレクトリのルート。
staging\classes	アダプタのコンパイルされた Java クラス。
staging\adapters	各アダプタのビルド済み jar が格納されます。アダプタの名前は、adapter_<adaptershortname>.jar となります。
staging\config	各アダプタの config サブディレクトリにあったファイル。
staging\deploy	それぞれの deploy サブディレクトリのファイル名は、<adaptershortname>.xml に変更されます。
staging\lib	各アダプタの lib サブディレクトリにあったファイル。
staging\dist	最後の isee.adapters の展開可能なファイル。

アダプタのソース構造の作成

新しいアダプタを作成するには:

手順 1　前述のようなディレクトリ構造を作成します。

手順 2　ソースを作成し、構造内に配置します。

手順 3　adapter.xml を作成し、deploy ディレクトリに配置します。詳細については、adapter.xml 記述子についてを参照してください。

手順 4　オプションで、追加のライブラリおよび追加のコンフィギュレーション ファイルを追加します。

手順 5　build.properties を変更し、使用するアダプタをアダプタの行に追加します。これにより、作成したディレクトリを検索するよう ANT が設定されます。

コンパイル手順では、このビルドをバージョン管理に追加することができ、後でインストール前にコンパイルできます。

アダプタのコンパイル

アダプタを作成した後、次を実行することによってアダプタをビルドします。

build.cmd(UNIX システムの場合は **./build.sh**)

最終的な生成物が、**staging/dist/isee.adapters** に生成されます。導入するときは、このファイルを管理者に渡してください。

アダプタの導入

アダプタを導入するには、管理者が次の手順を実行します。

手順 1　Service Link のカスタム アダプタ パッケージを取得します。これは、zip ファイル形式でなければなりません。

手順 2　一時ディレクトリ（c:\temp\adapter など）にアダプタを解凍します。このディレクトリを、今後 <AH> と呼びます。

手順 3　展開された「ISEE.war/WEB-INF/lib」ディレクトリに <AH>/adapters/<ADAPTER_NAME>.jar をコピーします。

手順 4　展開された「ISEE.war/WEB-INF/lib」ディレクトリに <AH>/lib/* ファイル（存在する場合）をコピーします。

手順 5　展開された「ISEE.war/WEB-INF/classes」ディレクトリに <AH>/config/* ファイル（存在する場合）をコピーします。

手順 6　展開された「ISEE.war/WEB-INF/classes」ディレクトリに <AH>/udk/* ファイル（存在する場合）をコピーします。

手順 7　カスタム アダプタが Adapter Development Kit を使用して内部的に作成されていない場合は、「<ServicePortal_Software_Dir>/adk」フォルダから adk.zip を入手します。ここで、<ServicePortal_Software_Dir> はサービス カタログ（Service Catalog）アプリケーションの解凍されたソフトウェア イメージです。adk.zip を c:\adk（Windows の場合）または /opt/adk（UNIX/Linux の場合）に解凍します。このディレクトリを、今後 <ADK> と呼びます。

手順 8　JAVA_HOME 環境変数が環境にまだ設定されていない場合は、これを設定します。

手順 9　コマンド ウィンドウを開き、<ADK>/lib フォルダにディレクトリを変更します。環境に応じて adapter_dbinstaller.sh または adapter_dbinstaller.cmd を実行します。--help または -? をアダプタ インストーラの引数として使用し、必要な入力引数のリストを表示します。Adapter Deployment Descriptor ファイルの指定を求められたら、<AH>/deploy ディレクトリの下に xml ファイル名をフル パスで入力します（/opt/<AH>/deploy/custom_adapter.xml など）。

手順 10　手順 6 でインストールされた各 udk ファイルについて、integrationserver.properties ファイル内の「UDConfig」プロパティにファイル名を追加します。UDConfig プロパティはすべての udconfig ファイルのカンマ区切りリストです。アダプタの udconfig ファイルをこのリストに追加します。

手順 11　Service Catalog および Service Link サーバを起動します。新しいアダプタがあることを確認します。

アダプタの実装について

アダプタは Service Link が外部システム（通常はサードパーティ システムと呼ばれます）に接続する手段です。アダプタは、3 つの部分で構成されます。

- 着信部分。**着信アダプタ**と呼ばれます。
- 発信部分。**発信アダプタ**と呼ばれます。
- エラーハンドラ

着信アダプタは、Service Catalog への着信通信を管理します。これは、システムに届く XML メッセージを処理します。受信アダプタには、ポーラーとリスナーの 2 種類があります。ポーラーは定期的に起動して、着信メッセージを探すスレッドです。これに対し、リスナーは待機し、外部イベントによって起動されます。ポーラーの例としては、メッセージを定期的にチェックする必要がある受信ファイル アダプタがあります。リスナーの例としては、HTTP XML イベントがポストされるまで待機する HTTP アダプタがあります。

発信アダプタは Service Catalog から発信される XML メッセージを管理します。発信アダプタのタイプは 1 つだけです。

エージェントは論理要素であり、これによりサービス設計者は、複雑なアダプタおよび接続プロパティをすべて把握する必要がなくなります。また、エージェントは着信アダプタと発信アダプタを定義します。着信アダプタは任意選択で、「自動完了（Auto complete）」と指定できます。「自動完了（Auto complete）」はワークフローの進行にサードパーティ システムからの応答を必要としないモードで、ほとんどの場合、電子メール ベース システムのような、信頼できない、または一度確立したら後は調整されないプロトコルと関連づけられます。管理者は、サービス設計者が使用できるようにエージェントおよびエージェントとアダプタとの関連付けを設定します。

また、メッセージがサードパーティ システムに送られる前、またはサードパーティ システムから受信され、Service Link に配信された後で、XML 変換をメッセージに適用できます。

メッセージ システムは nsXML と呼ばれる一般的な XML 方言を使用します。これは、Service Link が処理または作成できる有効な XML を定義するスキーマです。nsXML は、現在 6 つの操作で構成されています。

- task-started：発信
- task-cancelled：発信
- take-action：着信
- send-parameters：着信
- add-comment：着信

発信の場合、Service Link はこれらの操作を宛先に変換できます。同じことが着信メッセージでも行われ、XSL 変換によって外部形式を nsXML の方言に変換できます。

nsXML 操作の詳細については、通信メッセージのコンテンツと構造の理解を参照してください。

アダプタの種類

アダプタには次の 2 つのタイプがあります。

- 転送アダプタ

  転送アダプタは HTTP、ファイル、JMS、または独自のネットワーク ソケット実装など、特定の転送に固有のものです。

- アプリケーション アダプタ

  アプリケーション アダプタには転送要素がありますが、Remedy や Siebel などの特定のサードパーティ アプリケーションを使用したほうがよく理解できます。多くの場合、jar によりネイティブ API が提供されます。このバージョンの Service Link では、転送アダプタを拡張してアプリケーション アダプタを作成することはまだ不可能です。

エージェントは着信メッセージと発信メッセージに異なるアダプタを使用することができます。

アダプタのコンポーネント

java コードの他には、次のものでアダプタは構成されています。

- Jar ライブラリ（Remedy java API など）
- 静的コンフィギュレーション ファイル。一度導入されたテキスト ファイルの変更は推奨されません。
- 展開記述子。アダプタを記述する XML ファイル。

接続プロパティ

アダプタは、Service Link モジュールが管理者に公開している接続プロパティを、サードパーティ システムに接続するために公開することができます。これらは XML 展開記述子内に記述されており、その値は java コードによって取得し、完成度の高い API に渡すことができます。

例: ファイル アダプタの実装

ここでは、簡単なアダプタを実装する方法を示します。サンプル アダプタは、外部環境との通信を行うファイル アダプタです。

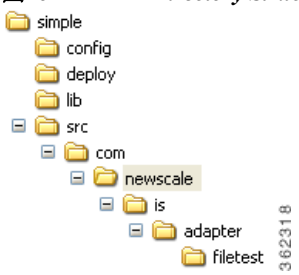
この単純なファイル アダプタには次のものが含まれています。

- ファイルを作成する発信アダプタ。ファイル名はアダプタのプロパティで指定されます。
- ファイルを読み取る着信アダプタ。ファイル名はアダプタのプロパティで指定されます。
- 単純な例外ハンドラ。

ディレクトリ構造の作成

最初に、アダプタのディレクトリ構造を作成します。ADK ディレクトリ構造内に、**simple** という名前のディレクトリを作成し、その下に次のディレクトリ構造を作成します。

図 6-1　Directory Structure



src の下が、java パッケージ **com.newscale.is.adapter.filetest** を表すソース パッケージになっていることに注目してください。他のパッケージも使用できますが、この例ではこれを使用します。

発信アダプタ クラスの作成

2 番目に、発信アダプタ クラスを作成します。名前は **FileOutboundAdapter** で、このクラス ファイルは、手順 1 で説明したパッケージに配置する必要があります。その枠組みを次に示します（メソッドの実装は含まれていません）。

```
import com.newscale.is.adk.AdapterContext;
import com.newscale.is.adk.base.OutboundAdapter;
import com.newscale.is.adk.exceptions.AdapterException;
public class  FileOutboundAdapter extends OutboundAdapter {
  public FileOutboundAdapter (AdapterContext context) {
    super(context);
  }
  public void initiate (AdapterContext context) throws AdapterException {
  }
  public void processMessage (String message, String channelId)throws AdapterException {
  }
  public void terminate () throws AdapterException {
  }
  public void commit() throws AdapterException {
  }
  public void rollback() throws AdapterException {
  }
}
```

発信アダプタを作成するには、上記のようにクラス **com.newscale.is.adk.base.OutboundAdapter** をこのクラスで拡張する必要があります。

パラメータとして **com.newscale.is.adk.AdapterContext** を受け取るコンストラクタを実装します。このコンストラクタの実装に関して推奨される方法についても、上に示しています(スーパー コンストラクタを呼び出しています)。

上記のように initiate メソッドを実装します。このメソッドは、アダプタを使用するエージェントが起動されると呼び出されます。このメソッドが空の場合は省略可能です。

processMessage メソッドを実装します。このメソッドは、メッセージの送信準備が完了したときに呼び出されます。トランスフォーマがエージェントで指定されている場合、そのトランスフォーマによってメッセージが変換されています。

terminate メソッドを実装します。エージェントの停止時にこのメソッドを呼び出します。このメソッドが空の場合は省略可能です。

commit メソッドを実装します。エージェントがトランザクションを完了しようとしているときに、このメソッドを呼び出します。このメソッドが空の場合は省略可能です。このメソッドを使用すると、トランザクションを **processMessage** で開始し、後で完了できます。

rollback メソッドを実装します。このメソッドは、エージェントがトランザクションをロールバックしようとしているときに呼び出されます。このメソッドを空のままにする場合は省略可能です。このメソッドを使用すると、トランザクションを **processMessage** で開始し、後で再呼び出しできます。

トランザクション サポートの詳細については、トランザクション通知の設定を参照してください。

この例では、ファイル発信クラスは xml コンテンツを持つファイルを書き込みます。このために、ファイルが保存されファイル名を保持する変数を最初に設定します。これには、エージェントのプロパティを使用します。

```
String path = null;
 public void initiate (AdapterContext context)
   throws AdapterException {
   Properties properties = context.getProperties();
   this.path = properties.getProperty("OB_FILE_DIR") + "/" +
       properties.getProperty("OB_FILE_NAME");
 }
```

文字列が受信される場合、それをファイルに書き込むことは自明です。

```
  public void processMessage (String message, String channelId)
    throws AdapterException {
    try {
      Writer w = new FileWriter(path);
      w.write(message);
      w.close();
    } catch (Exception e) {
      e.printStackTrace();
      throw new AdapterException(1, "Problem while writing to a file: " +
          e.getMessage());
    }
}
```

もちろん、このコードは説明のために大幅に簡素化しています。

ポーラー着信アダプタ クラスの作成

着信アダプタの枠組みは次のようになります。

```
public class FileInboundAdapter extends InboundAdapter {
  public FileInboundAdapter (AdapterContext context) {
    super(context);
  }
  public void initiate (AdapterContext context) throws AdapterException {
  }
  public String receiveMessage () throws AdapterException {
      return null;
    }
  }
  public void terminate () throws AdapterException {
  }
  public void commit()
    throws AdapterException {
  }
  public void rollback()
    throws AdapterException {
  }
}
```

メソッドのセマンティクスは発信アダプタのものとほとんど同じです。唯一の例外は **receiveMessage** メソッドです。**receiveMessage** メソッドは、ポーラー アダプタの場合は定期的に呼び出されます。データが検出されると、このメソッドは有効な xml をサードパーティ形式で返します。データが検出されなかった場合は、ヌルが返されます。着信アダプタのコードは次のとおりです(発信アダプタと同様、正しいパラメータを使用して初期化が行われます)。

```
  String path = null;
  public void initiate (AdapterContext context)
    throws AdapterException {
    Properties properties = context.getProperties();
    this.path = properties.getProperty("IB_FILE_DIR") + "/" +
        properties.getProperty("IB_FILE_NAME");
  }
```

ファイルの処理は次のように行われます。

```
 public String receiveMessage () throws AdapterException {
    String receivedMessage = "";
    char data[] = {};
    try {
      StringBuffer buffer = new StringBuffer();
      FileInputStream fis = new FileInputStream(path);
      InputStreamReader isr = new InputStreamReader(fis, "UTF8");
```

```
    Reader in = new BufferedReader(isr);
    int ch;
    while ((ch = in.read()) > -1) {
      buffer.append((char) ch);
    }
    in.close();
    String requestString = buffer.toString();
    boolean success = (new File(path)).delete();
    return requestString;
  } catch (Exception e) {
    return null;
  }
}
```

リスナー着信アダプタの作成

リスナー アダプタは、Ad-Hoc プロセスを利用して作成されます。着信アダプタ クラスと、servlet のような実際のレシーバ クラスという 2 つのクラスが動作します。レシーバ クラスは、チャネル ID を取得する必要があります。レシーバ クラスは、次のようにして InboundAdapter クラスを発見します。

```
ChannelInfoVO chVo = AgentDAO.getInstance().getChannelInfo(channelId);
 if(chVo != null){
   Adapter adapter =
       AgentManager.getInstance().getAdapter(chVo.getAgentId());
   ((InboundAdapter).receiverProcess(xml);
 }
```

着信アダプタには、メッセージにより呼び出す必要がある **receiverProcess** メソッドか、または、**toString()** メソッドがメッセージのテキストを返すオブジェクトがあります。サンプルには、リスナー着信アダプタは用意されていません。

例外ハンドラの実装

2 つのアダプタが完成したら、例外ハンドラを実装する必要があります。ここでは非常に単純なクラスを使用しているため、必要な処理はエラーの出力のみです。完全なクラスを以下に示します。

```
public class FileExceptionHandler implements ITransportExceptionHandler {
  public FileExceptionHandler () {
  }
  public void handleException (Map props, String message) {
    System.out.println("Outbound Message Failed to deliver: " + message);
  }
}
```

トランザクション通知の設定

アダプタにはトランザクション サポートが提供されているため、アダプタのトランザクションを取り消す前にはエージェントへの通知が行われます。この目的のために、メソッド **commit** および **rollback** が追加されています。

(注)　システムでトランザクションのコミットまたはロールバックを実行中に、これらのメソッドに論理コードを追加すべきではありません。これらのメソッドでロールバックまたはコミットするのは、自身のリソースのみにする必要があります。

次の一般的な手法をアダプタで使用すると、コミットまたはロールバックされたリソースをトラッキングすることができます。

静的マップを作成します。処理メソッド（processMessage または receiveMessage）が呼び出されると、そのメソッドは次のものを追加できます。

```
private static Map resources = new HashMap();
public void processMessage (String message, String channelId)
  throws AdapterException {
  Connection con = … // obtain a connection to external resource
  Map.put(Thread.getCurrentThread(), con);
}

public void commit() throws AdapterException {
  con = (Connection) map.get(Thread.getCurrentThread());
  map.remove(Thread.getCurrentThread());
  con.commit();
}
```

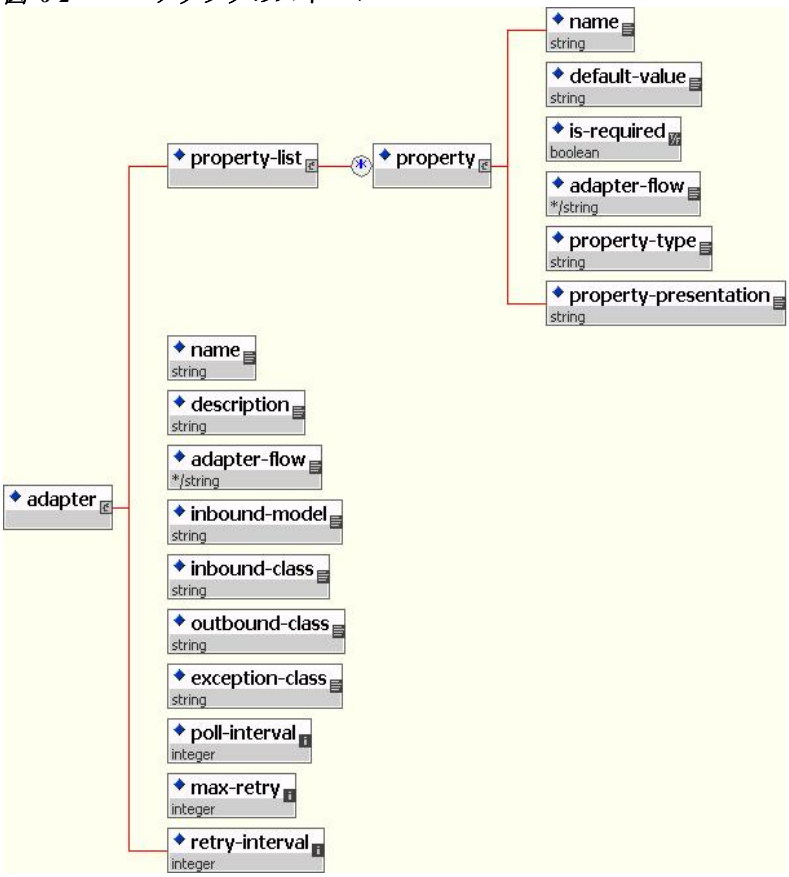
adapter.xml 記述子について

アダプタ記述子には、アダプタの導入とプロパティに関する情報が格納されます。

アダプタのスキーマ

アダプタのスキーマを次に示します。

図 6-2　　アダプタのスキーマ



「adapter」要素のフィールドの説明

name：アダプタの名前

description：アダプタの説明

adapter-flow：

有効な値は次のとおりです。

- 「inbound」
- 「outbound」

inbound-model：

有効な値は次のとおりです。

- 「listener」
- 「poller」
- 「extendedpoller」

inbound-class：着信アダプタの絶対クラス名

outbound-class:発信アダプタの絶対クラス名

exception-class：このアダプタの例外ハンドラの絶対クラス名

poll-interval：ミリ秒単位のポール間隔（「poller」タイプのアダプタに適用される）

max-retry：メッセージが失敗した場合に行われる再試行の最大回数

retry-interval：再試行間隔（ミリ秒）

「property」（アダプタ プロパティ）要素のフィールドの説明。

name：アダプタ プロパティの名前

default-value：プロパティのデフォルト値

is-required：必須プロパティとオプション プロパティのいずれであるか。有効な値は「true」または「false」

property-type：プロパティの種類。現在のところ、有効な値は「string」です。

property-presentation：有効な値は「text」および「password」

adapter-flow：

有効な値は次のとおりです。

- 「inbound」
- 「outbound」

Adapter.xml の例

```
<?xml version="1.0" encoding="UTF-8"?>
<adapter>
<property-list>
<property>
<name>InboundFinalResolution</name>
<default-value>Preserve</default-value>
<is-required>true</is-required>
<adapter-flow>inbound</adapter-flow>
<property-type>string </property-type>
```

```
<property-presentation>text</property- presentation>
</property>
<property>
<name>InboundFileLocation</name>
<default-value>C://SL2//InboundFiles</default-value>
<is-required>true</is-required>
<adapter-flow>inbound</adapter-flow>
<property-type>string </property-type>
<property-presentation>text</property- presentation>
</property>
<property>
<name>OnError</name>
<default-value>Preserve</default-value>
<is-required>true</is-required>
<adapter-flow>inbound</adapter-flow>
<property-type>string </property-type>
<property-presentation>text</property- presentation>
</property>
<property>
<name>InboundBackupLocation</name>
<default-value>c://SL2//InboundBackup</default-value>
<is-required>true</is-required>
<adapter-flow>inbound</adapter-flow>
<property-type>string </property-type>
<property-presentation>text</property- presentation>
</property>
<property>
<name>BackupSuffix</name>
<default-value>.bak</default-value>
<is-required>true</is-required>
<adapter-flow>inbound</adapter-flow>
<property-type>string </property-type>
<property-presentation>text</property- presentation>
</property>
<property>
<name>FileNameDateFormat</name>
<default-value>.yyyyMMddHHmmssSSS</default-value>
<is-required>true</is-required>
<adapter-flow>inbound</adapter-flow>
<property-type>string </property-type>
<property-presentation>text</property- presentation>
</property>
<property>
<name>InboundTempLocation</name>
<default-value>C://SL2//InboundTemp</default-value>
<is-required>true</is-required>
<adapter-flow>inbound</adapter-flow>
<property-type>string </property-type>
<property-presentation>text</property- presentation>
</property>
<property>
<name>OutboundConflictResolution</name>
<default-value>Rename</default-value>
<is-required>true</is-required>
<adapter-flow>outbound</adapter-flow>
<property-type>string </property-type>
<property-presentation>text</property- presentation>
</property>
<property>
<name>OutboundFileLocation</name>
<default-value>C://SL2//InboundFiles</default-value>
<is-required>true</is-required>
<adapter-flow>outbound</adapter-flow>
<property-type>string </property-type>
```

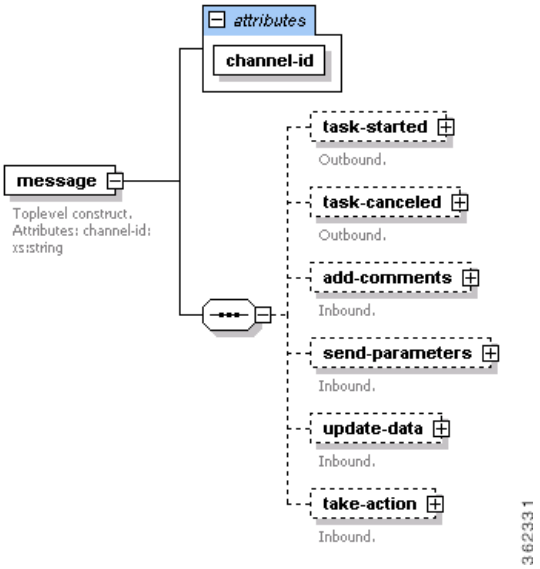
```
<property-presentation>text</property- presentation>
</property>
<property>
<name>OutboundBackupLocation</name>
<default-value>c://SL2//InboundBackup</default-value>
<is-required>true</is-required>
<adapter-flow>outbound</adapter-flow>
<property-type>string </property-type>
<property-presentation>text</property- presentation>
</property>
<property>
<name>OutboundTempLocation</name>
<default-value>C://SL2//InboundTemp</default-value>
<is-required>true</is-required>
<adapter-flow>outbound</adapter-flow>
<property-type>string </property-type>
<property-presentation>text</property- presentation>
</property>
</property-list>
<name>File Adapter</name>
<description>Read/write the external data from/to external file system</description>
<adapter-flow>inbound</adapter-flow>
<inbound-model>poller</inbound-model>
<inbound-class>com.newscale.is.adapter.file.FileInboundAdapter</inbound-class>
<outbound-class>com.newscale.is.adapter.file.FileOutboundAdapter</outbound-class>
<exception-class>com.newscale.is.adapter.file.FileExceptionHandler</exception-class>
<poll-interval>10000</poll-interval>
<max-retry>0</max-retry>
<retry-interval>0</retry-interval>
</adapter>
```

通信メッセージのコンテンツと構造の理解

ここでは、通信メッセージの内容と構造について詳しく説明します。メッセージの内容は、Service Catalog とサードパーティ システムの間で、HTTP、SOAP、JMS などの多様なキャリア プロトコルによって送信される XML 文書内にカプセル化されます。メッセージの構造およびサブ構造を理解しやすいよう、図に示します。

メッセージ(Message)

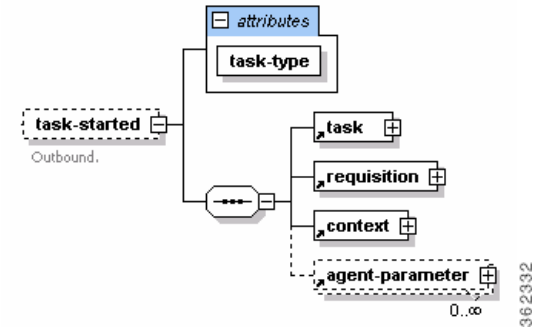
図 6-3　　メッセージ(Message)



発信文書のトップ レベル要素 message には、「task-started」または「task-canceled」要素が格納されます。着信文書のトップ レベル要素 message には、「add-comments」、「send-parameters」、または「take-action」要素が 1 つ以上格納されます。message タグには、channel-id という文字列タイプの必須属性があります。これは、作成された発信メッセージに対して Service Link が生成する一意の文字列値です。サード パーティ システムは、メッセージに対応する channel-id で応答する必要があります。この ID は、サービス カタログ(Service Catalog)とサード パーティ システムの両方で引き継がれる必要があります。

Task Started または Task Cancelled

図 6-4　　Task Started または Task Cancelled



Task started は、サードパーティ システムで外部アクティビティを開始します。この操作に関する設計上のポイントは、サードパーティ システムでタスクを実行するために必要となる可能性がある情報をすべて組み込むことです。この要素には、タスクおよびそのタスクが属している要求に関する、必要な詳細情報がすべて保持されます。また、オプションのエージェント パラメータも 1 つ以上含めることができます。context 要素は、サービス配信計画のコンテキスト内のタスクについて記述したものです。このノードに、要求レベルの確認および許可に関する値は格納されません。

タスク

図 6-5　　タスク

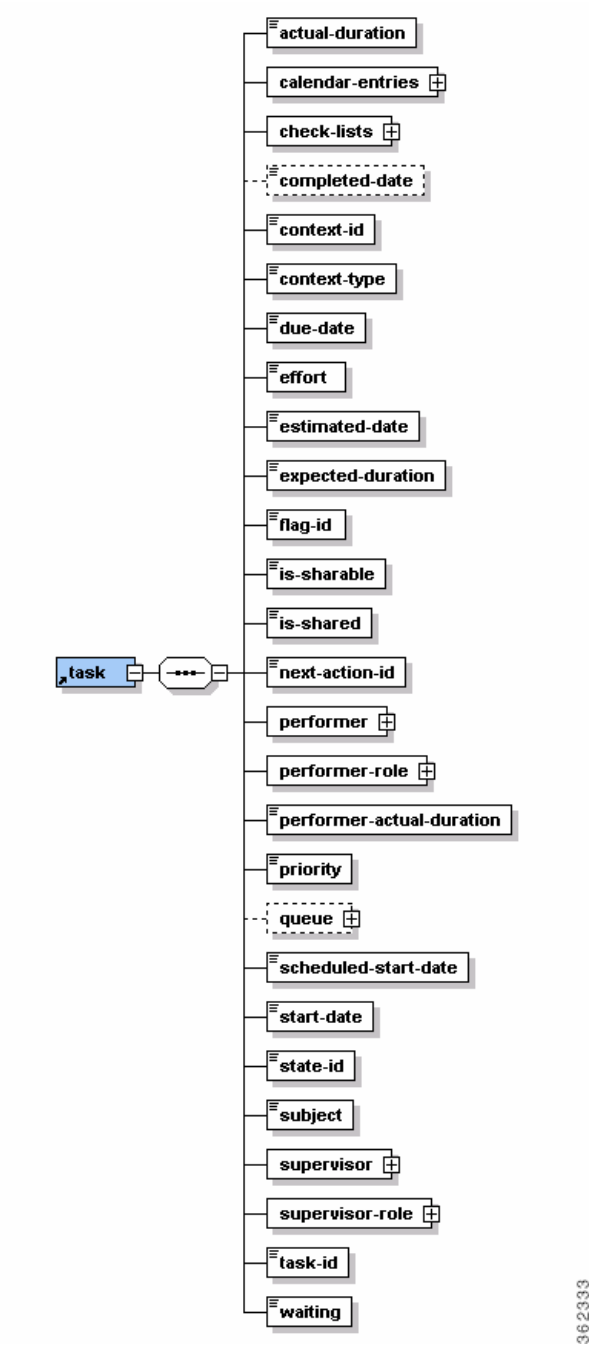
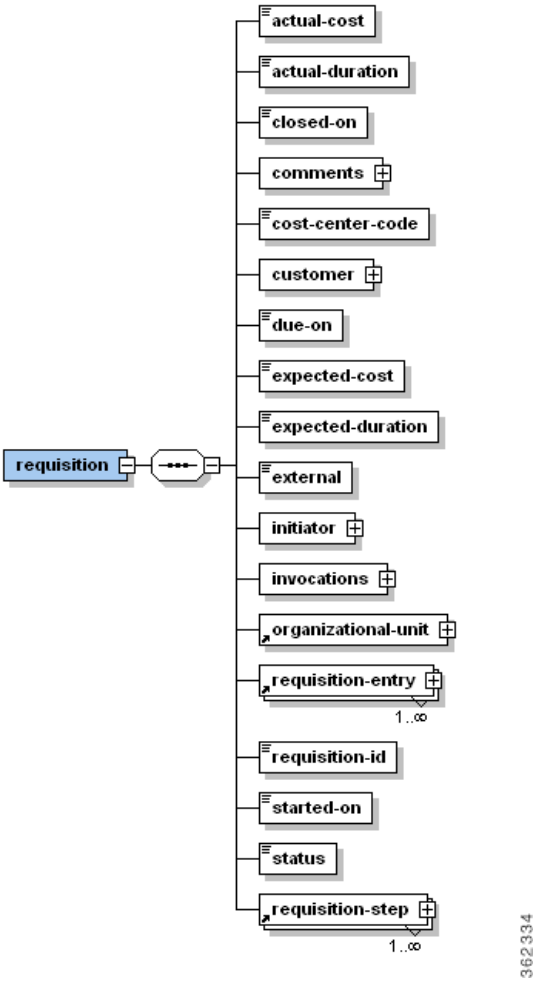

task
actual-duration
calendar-entries
check-lists
completed-date
context-id
context-type
due-date
effort
estimated-date
expected-duration
flag-id
is-sharable
is-shared
next-action-id
performer
performer-role
performer-actual-duration
priority
queue
scheduled-start-date
start-date
state-id
subject
supervisor
supervisor-role
task-id
waiting
362333

表 6-4　タスク要素と説明

要素	説明
actual-duration	タスクは実行中になったばかりであるため、空です。
calendar-entries	サービスを要求した個人のカレンダ エントリ。
check-lists	タスクのチェック リスト。
completed-date	タスクはまだ完了していないため、空です。
context-id	タスクが開始されたコンテキスト オブジェクトのオブジェクト ID。
context-type	オブジェクトのコンテキスト(例:要求エントリ)。
due-date	タスクの期日。
実行	予想されるタスクの時間単位の工数(1 人でタスクを完了するために必要な作業時間の予想)。
estimated-date	予測されるタスクの完了日時。
expected-duration	予想されるタスクの時間単位の所要期間(タスクの開始から完了までに要する作業時間の予想)。
flag-id	UI のカラー インジケータ。
is-sharable	タスクが共有可能かどうかを示すブール値。
is-shared	タスクが共有されているかどうかを示すブール値。
next-action-id	タスクに関して次に実行可能なアクションは何か。
performer	タスクの実行者。performer 要素には、関連付けられた個人オブジェクトがあります。
performer-actual-duration	タスクはまだ完了されていないため、空です。
performer-role	実行者が果たしている処理ロールは何か。
[プライオリティ(priority)]	タスクの優先順位:1、2、または 3(それぞれ高、中、または低を表す)。
キュー	タスクが割り当てられているキューの説明。
scheduled-start-date	タスクの開始が予定された日付。
start-date	タスクが開始された日付。
state-id	タスクの状態。
subject	タスクの件名。件名は、サービス定義で変更されます。要求エントリでは変更されません。
スーパーバイザ	タスクの上司。supervisor 要素には、関連付けられた個人オブジェクトがあります。
supervisor-role	上司が果たしている処理ロール。
task-id	このタスク インスタンスを一意に識別するために使用される整数。要求エントリのタスクごとに新しい番号が生成されます。
待機中	サブタスクおよびサードパーティ システムを含む、このタスクの依存関係を表すインジケータ。

要求

図 6-6　　要求



requisition 要素は、すべての要求と要求エントリ データをカプセル化したものであり、外部アクティビティを実行するときに、統合の目的でこのデータを使用できます。

表 6-5　　要求要素の説明表

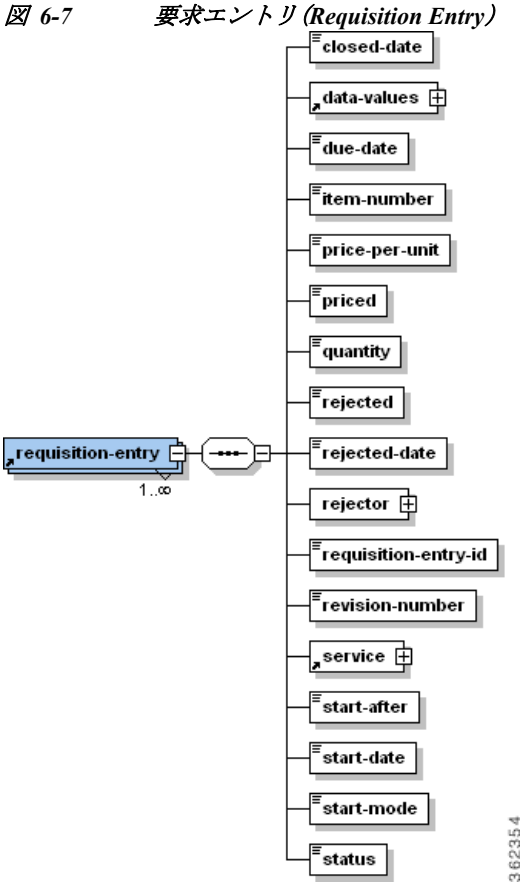
要素	説明
サービス	要求されるサービス数(または要求エントリ数)。
actual-cost	要求の実際のコスト。
actual-duration	要求の実際の所要期間。
closed-on	要求はまだ完了していないため、空です。
コメント	要求に関するコメント。
cost-center-code	未使用。
インタビューの	要求が注文された目的となっている個人。個人のオブジェクト データが保持されます。
due-on	要求の配信期限を表す日時。

表 6-5　要求要素の説明表(続き)

要素	説明
expected-cost	予想される要求のコスト。
expected-duration	要求全体の処理にかかると予想される所要期間(時間単位)。
外部	要求が外部システムから開始されたかどうかを示すブール値。
initiator	この要求をオーダーした個人。個人のオブジェクト データが保持されます。
呼び出し	RAPI(要求 API)からセットアップされた属性。
organizational-unit	要求者(発信者)の組織単位。
requisition-entry	要求エントリのデータ。
requisition-id	送信された要求の整数 ID。要求を手動で送信した後に My Services および Service Manager に表示される ID と同じです。
requisition-step	要求の承認または配信のステップとステータス。
started-on	要求が開始された日付。
status	要求の現在の状態。外部アクティビティの実行中は Ongoing となります。

要求エントリ

図 6-7　要求エントリ(Requisition Entry)



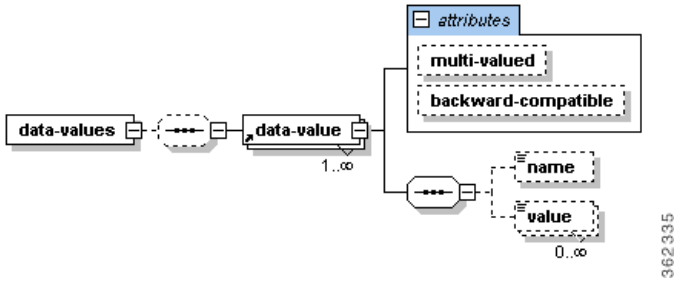
このタグは、1 つの要求エントリのデータをすべてカプセル化したものであり、統合のために使用できます。

表 6-6　要求エントリ要素の説明表

要素	説明
closed-date	要求エントリのステータスが ongoing から completed に変化した日時。タスクが ongoing である間は要求がクローズされないため、空になります。
data-values	要求エントリのデータ値。
due-date	要求が終了する予定の日付。
item-number	要求内の要求エントリの項目番号。
price-per-unit	要求されたサービスの単価。
priced	要求の価格が設定されている場合は true、設定されていない場合は false。
quantity	注文された数量。
rejected	要求エントリが承認されたか、拒否されたかを示します。
rejected-date	拒否された場合は、その日付。
rejector	要求を拒否した個人を示します。
requisition-entry-id	エントリ ID。
revision-number	リビジョンが作成されている場合は、そのリビジョン番号を示します。
service	要求エントリが属しているサービスに関連する要素。
start-after	遅れて開始される日付。
start-date	要求エントリが開始された日付。
start-mode	要求エントリがすぐに開始されるか、遅延されるかを指定します。
status	要求エントリのステータス：closed または ongoing。

データ値

図 6-8　データ値



data-values 要素は、ディクショナリ データで構成されるデータ値を 1 つ以上持つことができます。data-value の name は「Dictionaryname.FieldName」を指し、value はサービスの注文時にユーザが入力した値です。その値が複数選択ドロップダウン リストになっている場合は、1 つの data-value 要素が複数の値を持つことができます。

サービス

図 6-9　　サービス

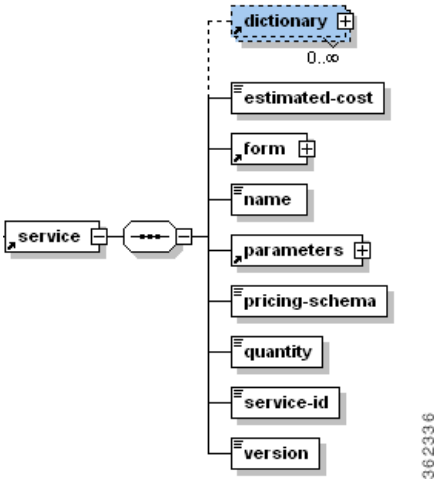


表 6-7　　サービス要素の説明表

要素	説明
ディクショナリ	service 要素は、ゼロ個以上のディクショナリを持つことができます。
estimated-cost	予測されるサービスのコスト。
form	サービス フォームのすべてのフィールド要素を保持する要素。
名前	サービスの名前。
パラメータ	このサービスに定義されているパラメータ。
pricing-schema	サービスが、入札で価格設定を行うタスクであるか、固定価格であるかを指定します。
quantity	注文された数量。
service-id	Service Catalog のサービスの ID。
version	サービスの最終更新バージョン番号。
standard-duration	サービスの標準的な所要期間。

辞書（Dictionary）

図 6-10　　ディクショナリ

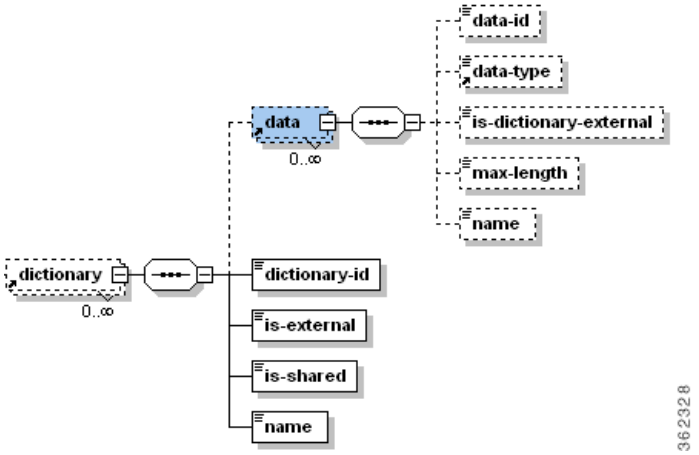


表 6-8　　ディクショナリ要素の説明表

要素	説明
caption	ディクショナリ内のキャプション データが格納された文字列値。
データ	ディクショナリ内のデータ要素。データ要素には、データ要素の名前、最大長、データ型、およびその他のメタデータの値が保持されます。
dictionary-id	Service Catalog 内のディクショナリのディクショナリ ID。
dictionary-template-type-id	ディクショナリの作成に使用されたテンプレート（たとえば、個人ベースのディクショナリの場合は 2）。
classification-id	ディクショナリの分類（サービス項目ディクショナリの場合のみ）。
mdr-data-type-id	ディクショナリのサービス項目タイプ（サービス項目ディクショナリの場合のみ）。
display-order	ディクショナリの表示順序が入った整数値。
is-external	ディクショナリが内部 Service Catalog ディクショナリであるか、外部ディクショナリであるかを示すブール値。
is-reportable	ディクショナリが、Advanced Reporting モジュールの Ad-Hoc レポート機能での使用に関して、レポート可能としてマークされているかどうかを示すブール値。
is-shared	ディクショナリが共有ディクショナリであるかどうかを示すブール値。
is-template	ディクショナリがテンプレートであるかどうかを示すブール値。この値は必ず false になります。
logic-name	ディクショナリの内部名（予約ディクショナリの場合のみ）。
名前	ディクショナリの名前。

フォーム

図 6-11　　フォーム（Form）

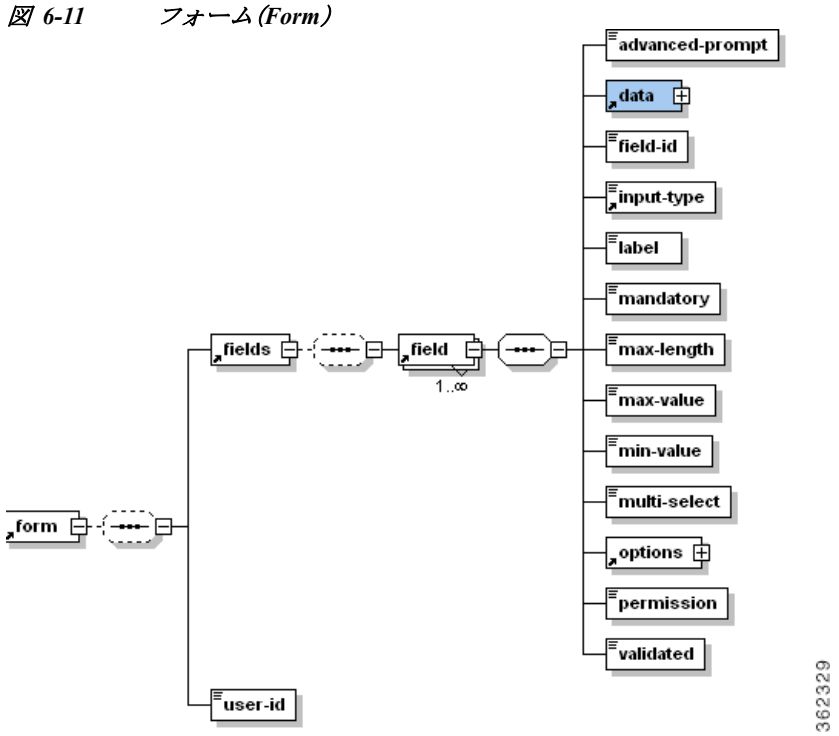
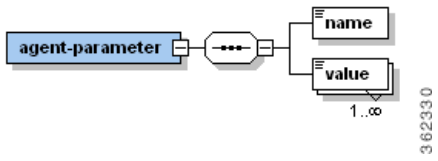


表 6-9　フォーム要素の説明表

要素	説明
フィールド	フィールドは、要求フォームの内部に 1 つまたは複数の field 要素を持ちます。 advanced-prompt：リッチ HTML プロンプト。 data：データ型、名前、長さなどのフィールドのデータが保持されます。 dictionary-display-order：dictionary-display-order の値は、フィールドに関連付けられた DataElement に関連付けられている Dictionary の DefObjectDictionaries.DisplayOrder の値です。 display-order：**display-order** の値は、フィールドの DefObjectDataHTML.DisplayOrder の値です。 field-id：Service Catalog データベース内でのフィールド ID。 input-type：フィールドの入力の種類（例：text、option など）。 label：フィールドに指定されているラベル。 mandatory：フィールドのデータは、サービスで必須です。 max-length：フィールドに指定されている最大長。 max-value：数値の場合は範囲が指定されます。 min-value multi-select：入力の種類が複数選択ボックスかどうか。 options：このデータ フィールドに使用可能なオプション リスト。 permission： validated：検証が必要かどうか。
user-id	

エージェント パラメータ

図 6-12　エージェント パラメータ



エージェント パラメータは、エージェントに対して指定されている外部パラメータを表します。これには multi-valued というブール属性があり、ユーザが複数の値を選択できるパラメータかどうかに基づいて true または false のいずれかになります。name はエージェント パラメータの名前を表し、value はその値を表します。

着信および発信ドキュメントの例

task-started または task-canceled（発信）

```
<?xml version="1.0" encoding="UTF-8"?>
<message channel-id="18071221:1124919814742:-32752"
xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <task-started task-type="task">
        <task>
            <actual-duration>0.0</actual-duration>
            <calendar-entries>
                <calendar-entry>
                    <calendar-entry-id>2</calendar-entry-id>
                    <date>Thu Aug 25 17:00:00 PDT 2005</date>
                    <end-time>Fri Aug 26 21:40:37 PDT 2005</end-time>
                    <is-blocked>false</is-blocked>
                    <is-break>false</is-break>
                    <is-read>false</is-read>
                    <person>
                        <company-address/>
                        <email>admin@company.com</email>
                        <fax/>
                        <first-name>admin</first-name>
                        <home-ou>
                            <name>&lt;s ID=&quot;847&quot;/&gt;</name>
                            <organizational-unit-id>1</organizational-unit-id>
                        </home-ou>
                        <home-phone/>
                        <last-name/>
                        <login-name>admin</login-name>
                        <person-id>1</person-id>
                        <personal-address/>
                        <supervisor-name/>
                        <timezone>Pacific Standard Time</timezone>
                        <work-phone/>
                    </person>
                    <sequence>0</sequence>
                    <start-time>Thu Aug 25 21:40:37 PDT 2005</start-time>
                    <subject>External Task</subject>
                </calendar-entry>
            </calendar-entries>
            <check-lists>
                <check-list-entry>
                    <display-order>1</display-order>
                    <is-mandatory>true</is-mandatory>
                    <last-date/>
                    <last-person/>
                    <name>Make sure you wake up</name>
                    <status>false</status>
                </check-list-entry>
                <check-list-entry>
                    <display-order>2</display-order>
                    <is-mandatory>true</is-mandatory>
                    <last-date/>
                    <last-person/>
                    <name>Make sure you take a shower</name>
                    <status>false</status>
                </check-list-entry>
                <check-list-entry>
                    <display-order>3</display-order>
```

```
                        <is-mandatory>true</is-mandatory>
                        <last-date/>
                        <last-person/>
                        <name>Make sure you have breakfast</name>
                        <status>false</status>
                    </check-list-entry>
                </check-lists>
                <completed-date/>
                <context-id>1</context-id>
                <context-type>Requisition Entry</context-type>
                <due-date>Fri Aug 26 21:40:37 PDT 2005</due-date>
                <effort>10.0</effort>
                <estimated-date/>
                <expected-duration>10.0</expected-duration>
                <flag-id>0</flag-id>
                <is-sharable>false</is-sharable>
                <is-shared>false</is-shared>
                <next-action-id>2</next-action-id>
                <performer>
                    <company-address/>
                    <email>admin@company.com</email>
                    <fax/>
                    <first-name>admin</first-name>
                    <home-ou>
                        <name>&lt;s ID=&quot;847&quot;/&gt;</name>
                        <organizational-unit-id>1</organizational-unit-id>
                    </home-ou>
                    <home-phone/>
                    <last-name/>
                    <login-name>admin</login-name>
                    <person-id>1</person-id>
                    <personal-address/>
                    <supervisor-name/>
                    <timezone>Pacific Standard Time</timezone>
                    <work-phone/>
                </performer>
                <performer-actual-duration>0.0</performer-actual-duration>
                <priority>2</priority>
                <scheduled-start-date>Thu Aug 25 21:40:37 PDT 2005</scheduled-start-date>
                <start-date>Wed Aug 24 21:42:15 PDT 2005</start-date>
                <state-id>2</state-id>
                <subject>External Task</subject>
                <supervisor>
                    <company-address>Foo Bar 25 Suite 300 Foo City CA  94404
USA</company-address>
                    <email>internal@company.com</email>
                    <fax/>
                    <first-name>Monkey</first-name>
                    <home-ou>
                        <name>&lt;s ID=&quot;847&quot;/&gt;</name>
                        <organizational-unit-id>1</organizational-unit-id>
                    </home-ou>
                    <home-phone/>
                    <last-name>McBride</last-name>
                    <login-name>monkey</login-name>
                    <person-id>3</person-id>
                    <personal-address>Fuchi Caca 16 Apartment C Fuchi Minn OR  78787
USA</personal-address>
                    <supervisor-name/>
                    <timezone>Pacific Standard Time</timezone>
                    <work-phone/>
                </supervisor>
                <task-id>3</task-id>
                <waiting>1</waiting>
```

```
            </task>
            <requisition>
                <actual-cost>0.0</actual-cost>
                <actual-duration>0.0</actual-duration>
                <closed-on/>
                <comments>
                    <comment>
                        <comment-date>Wed Aug 24 21:42:06 PDT 2005</comment-date>
                        <comment-id>1</comment-id>
                        <comment-text>I am adding a comment and I cannot think of a better
comment</comment-text>
                        <component-id>3</component-id>
                        <component-name>Request Center Component</component-name>
                        <person>
                            <company-address/>
                            <email>admin@company.com</email>
                            <fax/>
                            <first-name>admin</first-name>
                            <home-ou>
                                <name>&lt;s ID=&quot;847&quot;/&gt;</name>
                                <organizational-unit-id>1</organizational-unit-id>
                            </home-ou>
                            <home-phone/>
                            <last-name/>
                            <login-name>admin</login-name>
                            <person-id>1</person-id>
                            <personal-address/>
                            <supervisor-name/>
                            <timezone>Pacific Standard Time</timezone>
                            <work-phone/>
                        </person>
                        <source-object-id>2</source-object-id>
                        <source-object-inst-id>1</source-object-inst-id>
                    </comment>
                </comments>
                <cost-center-code/>
                <customer>
                    <company-address/>
                    <email>admin@company.com</email>
                    <fax/>
                    <first-name>admin</first-name>
                    <home-ou>
                        <name>&lt;s ID=&quot;847&quot;/&gt;</name>
                        <organizational-unit-id>1</organizational-unit-id>
                    </home-ou>
                    <home-phone/>
                    <last-name/>
                    <login-name>admin</login-name>
                    <person-id>1</person-id>
                    <personal-address/>
                    <supervisor-name/>
                    <timezone>Pacific Standard Time</timezone>
                    <work-phone/>
                </customer>
                <due-on>Fri Aug 26 21:40:37 PDT 2005</due-on>
                <expected-cost>0.0</expected-cost>
                <expected-duration>0.0</expected-duration>
                <external>false</external>
                <initiator>
                    <company-address/>
                    <email>admin@company.com</email>
                    <fax/>
                    <first-name>admin</first-name>
                    <home-ou>
```

```
            <name>&lt;s ID=&quot;847&quot;/&gt;</name>
            <organizational-unit-id>1</organizational-unit-id>
        </home-ou>
        <home-phone/>
        <last-name/>
        <login-name>admin</login-name>
        <person-id>1</person-id>
        <personal-address/>
        <supervisor-name/>
        <timezone>Pacific Standard Time</timezone>
        <work-phone/>
    </initiator>
    <invocations/>
    <organizational-unit>
        <name>&lt;s ID=&quot;847&quot;/&gt;</name>
        <organizational-unit-id>1</organizational-unit-id>
    </organizational-unit>
    <requisition-entry>
        <closed-date/>
        <data-values>
            <data-value>
                <name>Requester</name>
                <value>John McGarzafi</value>
            </data-value>
            <data-value>
                <name>RemedyStuff.TicketID</name>
                <value>None yet</value>
            </data-value>
            <data-value>
                <name>RemedyStuff.AssetNumber</name>
                <value>123456789</value>
            </data-value>
        </data-values>
        <due-date>Fri Aug 26 21:40:37 PDT 2005</due-date>
        <item-number>1</item-number>
        <price-per-unit>0.0</price-per-unit>
        <priced>true</priced>
        <quantity>1</quantity>
        <rejected>false</rejected>
        <rejected-date/>
        <rejector>
            <company-address/>
            <email/>
            <fax/>
            <first-name/>
            <home-phone/>
            <last-name/>
            <login-name/>
            <person-id>0</person-id>
            <personal-address/>
            <supervisor-name/>
            <timezone/>
            <work-phone/>
        </rejector>
        <requisition-entry-id>1</requisition-entry-id>
        <revision-number>5</revision-number>
        <service>
            <dictionary>
                <data>
                    <data-id>3</data-id>
                    <data-type>Person</data-type>
                    <is-dictionary-external>false</is-dictionary-external>
                    <max-length>100</max-length>
                    <name>Requester</name>
```

```
    </data>
    <dictionary-id>1</dictionary-id>
    <is-external>false</is-external>
    <is-shared>false</is-shared>
    <name>Monkey Service (private)</name>
</dictionary>
<dictionary>
    <data>
        <data-id>1</data-id>
        <data-type>Text</data-type>
        <is-dictionary-external>false</is-dictionary-external>
        <max-length>50</max-length>
        <name>TicketID</name>
    </data>
    <data>
        <data-id>2</data-id>
        <data-type>Text</data-type>
        <is-dictionary-external>false</is-dictionary-external>
        <max-length>50</max-length>
        <name>AssetNumber</name>
    </data>
    <dictionary-id>2</dictionary-id>
    <is-external>false</is-external>
    <is-shared>true</is-shared>
    <name>RemedyStuff</name>
</dictionary>
<estimated-cost>0.0</estimated-cost>
<form>
    <fields>
        <field>
            <advanced-prompt/>
            <data>
                <data-id>2</data-id>
                <data-type>Text</data-type>
                <is-dictionary-external>false</is-dictionary-external>
                <max-length>50</max-length>
                <name>AssetNumber</name>
            </data>
            <field-id>2</field-id>
            <input-type>text</input-type>
            <label>AssetNumber</label>
            <mandatory>false</mandatory>
            <max-length>50</max-length>
            <max-value>0.0</max-value>
            <min-value>0.0</min-value>
            <multi-select>false</multi-select>
            <options>
                <available-keys/>
                <available-labels/>
                <current-values/>
                <multivalued>false</multivalued>
            </options>
            <permission>4</permission>
            <validated>true</validated>
        </field>
        <field>
            <advanced-prompt/>
            <data>
                <data-id>1</data-id>
                <data-type>Text</data-type>
                <is-dictionary-external>false</is-dictionary-external>
                <max-length>50</max-length>
                <name>TicketID</name>
            </data>
```

```
                    <field-id>1</field-id>
                    <input-type>text</input-type>
                    <label>TicketID</label>
                    <mandatory>false</mandatory>
                    <max-length>50</max-length>
                    <max-value>0.0</max-value>
                    <min-value>0.0</min-value>
                    <multi-select>false</multi-select>
                    <options>
                        <available-keys/>
                        <available-labels/>
                        <current-values/>
                        <multivalued>false</multivalued>
                    </options>
                    <permission>4</permission>
                    <validated>true</validated>
                </field>
                <field>
                    <advanced-prompt>Give the name!</advanced-prompt>
                    <data>
                        <data-id>3</data-id>
                        <data-type>Person</data-type>
                        <is-dictionary-external>false</is-dictionary-external>
                        <max-length>100</max-length>
                        <name>Requester</name>
                    </data>
                    <field-id>3</field-id>
                    <input-type>text</input-type>
                    <label>Requester Name</label>
                    <mandatory>false</mandatory>
                    <max-length>100</max-length>
                    <max-value>0.0</max-value>
                    <min-value>0.0</min-value>
                    <multi-select>false</multi-select>
                    <options>
                        <available-keys/>
                        <available-labels/>
                        <current-values>
                            <string>John McGarzafi</string>
                        </current-values>
                        <multivalued>false</multivalued>
                    </options>
                    <permission>4</permission>
                    <validated>true</validated>
                </field>
            </fields>
            <user-id>0</user-id>
        </form>
        <name>Monkey Service</name>
        <parameters>
            <default-duration>0.0</default-duration>
            <priority>2</priority>
            <start-date/>
            <start-mode>0</start-mode>
        </parameters>
        <pricing-schema>0</pricing-schema>
        <quantity>1</quantity>
        <service-id>1</service-id>
        <version>5</version>
    </service>
    <start-after/>
    <start-date>Wed Aug 24 21:40:50 PDT 2005</start-date>
    <start-mode>0</start-mode>
    <status>1</status>
```

```
            </requisition-entry>
            <requisition-id>1</requisition-id>
            <started-on>Wed Aug 24 21:40:32 PDT 2005</started-on>
            <status>1</status>
            <requisition-step>
                <completed-on/>
                <due-on>Fri Aug 26 21:40:37 PDT 2005</due-on>
                <estimated-on>Fri Aug 26 21:40:37 PDT 2005</estimated-on>
                <name>Delivery project for Monkey Service</name>
                <status>2</status>
            </requisition-step>
        </requisition>
        <agent-parameter multi-valued="false">
            <name>Ticket</name>
            <value>None yet</value>
        </agent-parameter>
        <agent-parameter multi-valued="false">
            <name>Asset</name>
            <value>123456789</value>
        </agent-parameter>
    </task-started>
</message>
```

take-action（着信）

```
<?xml version="1.0" encoding="UTF-8"?>
<message  channel-id="18071221:1124919814742:-32752">
            <take-action action="done"/>
</message>
```

send-parameters（着信）

```
<?xml version="1.0" encoding="UTF-8"?>
<message  channel-id="18071221:1116468068789:-32360">
    <send-parameters>
      <agent-parameter>
         <name>Param1</name>
              <value>cat</value>
      </agent-parameter>
      <agent-parameter>
         <name>Param2</name>
              <value>catlitter</value>
       </agent-parameter>
     </send-parameters>
</message>
```

add-comments（着信）

```
<?xml version="1.0" encoding="UTF-8"?>
<message  channel-id="32580443:1116276793649:-32629">
            <add-comments>
               <comment>Test Comment</comment>
            </add-comments>
</message>
```

PART 3

外部システムとの統合

JSR ポートレットを使用した外部システムとの統合の開発

サービス カタログ（Service Catalog）の Portal Designer ソリューションは、JSR ポートレットを通じて外部アプリケーションと統合するための高機能なプラットフォームです。ポータルのフロントエンドでは、フレームワークに Apache Pluto 1.1 ライブラリを使用します。Java Portlet Specification（JSR168、JSR286）規格を満たす API を使用して作成されたポートレットは、サービス カタログ（Service Catalog）と共に導入できます。導入すると、Portal Designer に [サードパーティのポートレット（Third-Party Portlets）] と表示され、ポータル ページに追加できるようになります。JSR ポートレットその他のコンテンツを Portal Manager ソリューションでメンテナンスする方法の詳細については、『*Cisco Prime サービス カタログ（Service Catalog）Designer Guide*』を参照してください。

この章では、Portal Designer ソリューション用に JSR ポートレットを作成および導入する際のガイドラインをいくつか示します。章全体を通じて、「MyJSR」というサンプル ポートレットを使用して説明します。ポートレットは、Spring 3.0 Annotation ベースのコントローラおよび Sencha の Ext JS（ポータル フロントエンドの JavaScript フレームワーク）を使用して作成します。

ポートレットの構造とパッケージ化

ポートレット ファイルは JSR 168 または 286 仕様に従って、使用するアプリケーション サーバに適した Web アプリケーション（war）ファイルの形式でパッケージ化する必要があります。一般的なポートレット war ファイルには、サーブレット、リソース バンドル、イメージ、html、jsp、css ファイルなどが含まれます。

JBoss アプリケーション サーバ

「MyJSR.war」という単純なポートレットの構造を示します。

```
1. css
    MyJSR.css
2. images
    <Custom Images that the Portlet needs can be placed here>
3. js
    MyJSRCreatePersonView.js
    MyJSREdit.js
    MyJSRHelp.js
    MyJSRView.js
4. WEB-INF
    classes
```

```
    com
        myjsr
            MyJSRController.class
    config
        spring
            MyJSRApplicationContext.xml
    jsrportlet.properties
    log4j.properties
jsp
    MyJSREdit.jsp
    MyJSRHelp.jsp
    MyJSRView_listperson.jsp
    MyJSRView_updateperson.jsp
lib
    newscale_appclient.jar
    newscale_core.jar
    cxf-2.2.12.jar
    cxf-rt-transports-http-2.2.12.jar
    pluto-portal-driver-2. 0.2.jar
    org.springframework.aop-3.1.0.RELEASE.jar
    org.springframework.asm-3.1.0.RELEASE.jar
    org.springframework.aspects-3.1.0.RELEASE.jar
    org.springframework.beans-3.1.0.RELEASE.jar
    org.springframework.context-3.1.0.RELEASE.jar
    org.springframework.context.support-3.1.0.RELEASE.jar
    org.springframework.core-3.1.0.RELEASE.jar
    org.springframework.expression-3.1.0.RELEASE.jar
    org.springframework.instrument-3.1.0.RELEASE.jar
    org.springframework.instrument.tomcat-3.1.0.RELEASE.jar
    org.springframework.jdbc-3.1.0.RELEASE.jar
    org.springframework.jms-3.1.0.RELEASE.jar
    org.springframework.orm-3.1.0.RELEASE.jar
    org.springframework.oxm-3.1.0.RELEASE.jar
    org.springframework.test-3.1.0.RELEASE.jar
    org.springframework.transaction-3.1.0.RELEASE.jar
    org.springframework.web-3.1.0.RELEASE.jar
    org.springframework.web.portlet-3.1.0.RELEASE.jar
    org.springframework.web.servlet-3.1.0.RELEASE.jar
    org.springframework.web.struts-3.1.0.RELEASE.jar
tld
    c.tld
    pluto.tld
    portlet.tld
    portlet_2_0.tld
    portlet-el.tld
    portlet-el_2_0.tld
portlet.xml
web.xml
jboss-deployment-structure.xml
```

このサンプル ポートレットでは、ポートレットが REST API を呼び出してサービス カタログ（Service Catalog）からデータを取得するため、lib フォルダに nsAPI java クライアント（newscale_appclient.jar）が含まれています。nsAPI java クライアントが使用する pluto ライブラリおよびその他のライブラリが lib フォルダに含まれている必要があります。さらに JBoss モジュールの依存関係を記述するために、jboss-deployment-structure.xml が含まれます。

ポートレット関連の設定を指定するために、追加の記述子（portlet.xml）が必要です。

依存ライブラリ

JSR ポートレットの war ファイルに含める必要があるライブラリ セットは、アプリケーション サーバの RequestCenter 配置済みアプリケーションまたはサービス カタログ（Service Catalog）インストーラ イメージで入手できます。

-pluto-taglib-2.0.2.jar：製品イメージの「preinstall」フォルダ下

-他のすべてのファイル：RequestCenter.war/WEB-INF/lib

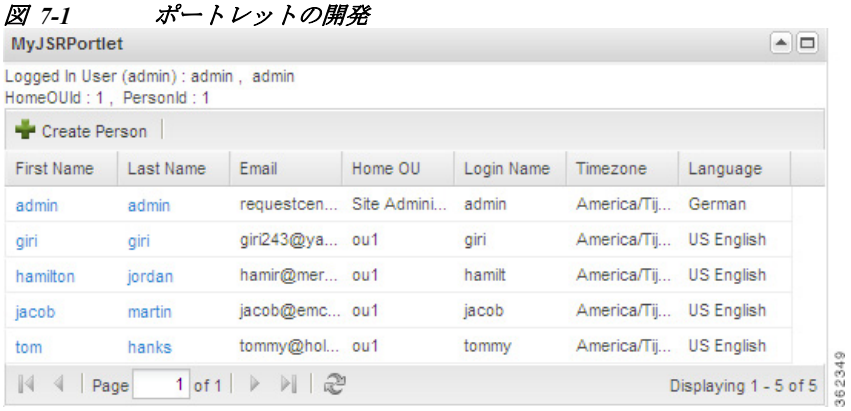
ポートレットの開発

一般的な JSR ポートレットでは、表示（View）、編集（Edit）、ヘルプ（Help）の 3 つのレンダリング モードをサポートする必要があります。さらに、通常（Normal）、最小化（Minimized）、最大化（Maximized）のウィンドウ状態をサポートする必要があります。

以下に示す MyJSR ポータルの例では、2 つの機能をサポートするユーザ インターフェイスが提供されます。

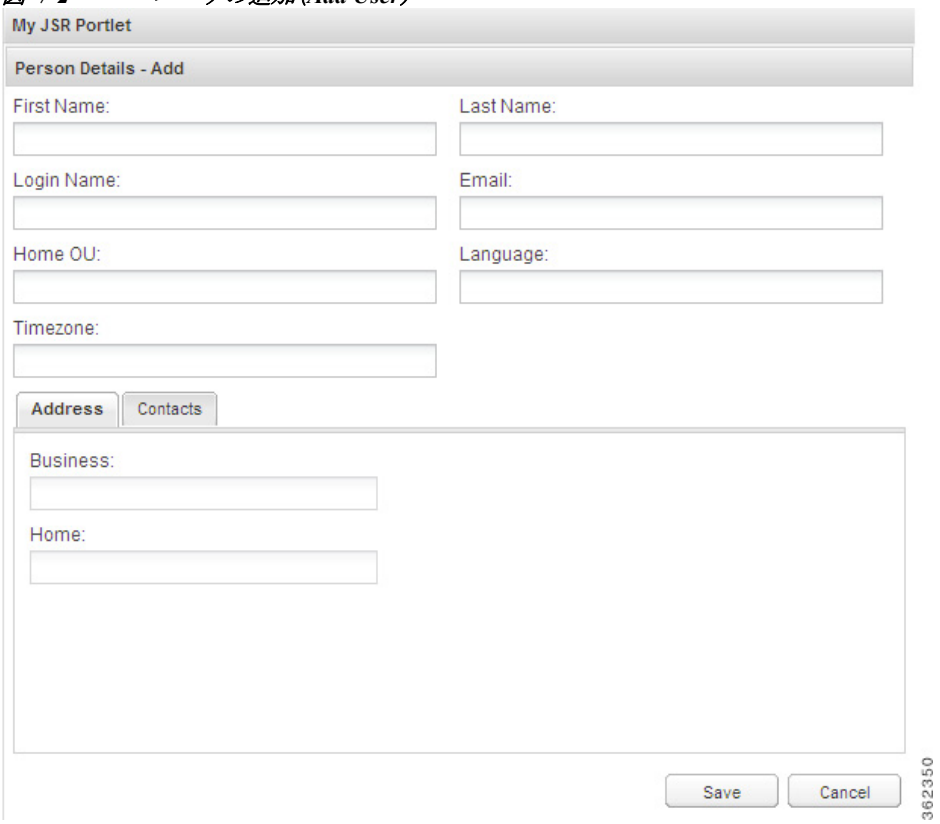
1. サービス カタログ（Service Catalog）ユーザをグリッドで一覧表示します。

図 7-1　　ポートレットの開発



2. ユーザを追加または更新できます。

図 7-2　　ユーザの追加（Add User）



この要件を満たすために、次に示すサンプル コードには、次の高水準操作が含まれています。

- nsAPI java クライアントを使用して、サービス カタログ（Service Catalog）ユーザを取得する
- ユーザの詳細を JSON 形式でユーザ インターフェイスに返す
- Ext JS グリッドで、ブラウザにユーザのリストをレンダリングする
- Ext JS で設計した形式で、ユーザの詳細を表示または入力する
- nsAPI java クライアントを使用して、サービス カタログ（Service Catalog）リポジトリのユーザの詳細を追加または更新する

MyJSR.css

ポートレットのコード カスタム スタイルをここで設計できます。

以降で、MyJSR.war 内のコンポーネントそれぞれの内容を見ていきます。

MyJSRCreatePersonView.js

Ext JS を使用して、ユーザを作成するためのフォームを表示するコード例。

```
/* Code custom JavaScript for the portlet here */

Ext.onReady(function() {
var tab2 = new Ext.FormPanel({
```

```
id : 'personEditForm',
labelAlign : 'top',
title : 'Person Details - Add',
bodyStyle : 'padding:5px',
width : 600,
renderTo : MyJSREditDiv,
items : [{
layout : 'column',
border : false,
items : [{
columnWidth : .5,
layout : 'form',
border : false,
items : [{
xtype : 'textfield',
fieldLabel : 'First Name',
               value : personListObj.firstName,
name : 'firstName',
anchor : '95%'
                                 }, {
xtype : 'textfield',
fieldLabel : 'Login Name',
               value : personListObj.login,
name : 'login',
anchor : '95%'
                                 }, {
xtype : 'textfield',
fieldLabel : 'Home OU',
name : 'homeOrganizationalUnitName',
               value : personListObj.homeOrganizationalUnitName,
anchor : '95%'
                                 }, {
xtype : 'textfield',
fieldLabel : 'Timezone',
name : 'timeZoneName',
               value : personListObj.timeZoneName,
anchor : '95%'
                                 }]
                        }, {
columnWidth : .5,
layout : 'form',
border : false,
items : [{
xtype : 'textfield',
fieldLabel : 'Last Name',
name : 'lastName',
               value : personListObj.lastName,
anchor : '95%'
                                 },{
xtype : 'textfield',
fieldLabel : 'Email',
name : 'email',
               value : personListObj.email,
vtype : 'email',
anchor : '95%'
                                 }, {
xtype : 'textfield',
fieldLabel : 'Language',
name : 'languageName',
               value : personListObj.languageName,
anchor : '95%'
                                 }]
                        }]
            }, {
```

```
xtype : 'tabpanel',
plain : true,
activeTab : 0,
height : 235,
defaults : {
bodyStyle : 'padding:10px'
                },
items : [{
title : 'Address',
layout : 'form',
defaults : {
width : 230
                        },
defaultType : 'textfield',
items : [{
fieldLabel : 'Business',
name : 'businessAddress',
disabled : true
                                }, {
fieldLabel : 'Home',
name : 'homeAddress',
disabled : true
                                }]
                    }, {
title : 'Contacts',
layout : 'form',
defaults : {
width : 230
                        },
defaultType : 'textfield',
items : [{
fieldLabel : 'Business',
name : 'businessPhone',
disabled : true
                                }, {
fieldLabel : 'Home',
name : 'homePhone',
disabled : true
                                }, {
fieldLabel : 'Mobile',
name : 'mobilePhone',
disabled : true
                                }, {
fieldLabel : 'Fax',
name : 'faxNumber',
disabled : true
                                }]
                    }]
            }],

buttons : [{
text : 'Save',
handler : function() {
Ext.getCmp("personEditForm").getForm().submit({
url : addPersonActionUrl,
params : {},
success : function(form, action) {
var responseObj = Ext.util.JSON.decode(action.response.responseText);
if(responseObj.success == "true")
Ext.Msg.alert('Success', responseObj.successMsg);
else
Ext.Msg.alert('Error', responseObj.errorMsg);
                    }
                   });
```

```
                }
            },{
text : 'Cancel',
handler : function() {
window.location=viewPersonUrl;
                }
                }]
   });
});
```

MyJSREdit.js

この JavaScript を使用すると、ポートレット編集モード用のカスタム コードを追加できます。

```
/* Code custom JavaScript for the Portlet here */

Ext.onReady(function() {

});
```

MyJSRHelp.js

この JavaScript を使用すると、ポートレット編集モード用のカスタム コードを追加できます。

```
/* Code custom JavaScript for the Portlet here */

Ext.onReady(function() {

});
```

MyJSRView.js

Ext JS グリッドにユーザを表示する JavaScript の例。

```
/* Code custom JavaScript for the Portlet here */
Ext.onReady(function() {
// Demonstrates how to getUser info from Java Script and set it to div
varLogin=document.getElementById('MyJSRLoginNameDiv');
  Login.innerHTML=nsAPP_CurrentUserLoginName;
varFirstName=document.getElementById('MyJSRFirstNameDiv');
  FirstName.innerHTML=nsAPP_CurrentUserFirstName;
varLastName=document.getElementById('MyJSRLastNameDiv');
  LastName.innerHTML=nsAPP_CurrentUserLastName;
varHomeOU=document.getElementById('MyJSRHomeOUDiv');
  HomeOU.innerHTML=nsAPP_CurrentUserHomeOuId;
var PersonID=document.getElementById('MyJSRPersonIDDiv');
  PersonID.innerHTML=nsAPP_CurrentUserId;

var pid = portletId.substr(pidPrefix.length);
if (Ext.getCmp(pid).height && Ext.getCmp(pid).height >= 29) {
var gridHeight = Ext.getCmp(pid).height - 29;
  }

var gridStore = new Ext.data.JsonStore({
proxy : new Ext.data.HttpProxy({
```

```
url : pagingUrl,
timeout : connectionTimeOut
            }),
autoLoad: {params:{start: 0, limit: defaultRecordSize}},
root: 'rows',
totalProperty: 'results',
fields : [{
name : 'firstName',
type : 'string'
                }, {
name : 'lastName',
type : 'string'
                }, {
name : 'email',
type : 'string'
                }, {
name : 'homeOrganizationalUnitName',
type : 'string'
                }, {
name : 'login',
type : 'string'
                }, {
name : 'timeZoneName',
type : 'string'
                }, {
name : 'languageName',
type : 'string'
                }, {
name : 'businessPhone',
type : 'string'
                }, {
name : 'homePhone',
type : 'string'
                }, {
name : 'mobilePhone',
type : 'string'
                }, {
name : 'faxNumber',
type : 'string'
                }, {
name : 'businessAddress',
type : 'string'
                }, {
name : 'homeAddress',
type : 'string'
                }]
        });
gridStore.load();

var expander = new Ext.ux.grid.RowExpander({
tpl : new Ext.Template(
'<h2 class="title">Address</h2><table>',
'<tr><td width=400><b>Business</b> {businessAddress}</td>',
'<td width=400><b>Home</b>{homeAddress}</td></tr></table>',
'<h2 class="title">Contact</h2><table>',
'<tr><td width=400><b>Business</b> {businessPhone}</td>',
'<td width=400><b>Home</b> {homePhone}</td></tr>',
'<tr><td width=400><b>Mobile</b> {mobilePhone}</td>',
'<td width=400><b>Fax</b> {faxNumber}</td></tr></table>')
        });

var gridColModel = new Ext.grid.ColumnModel({
defaults : {
sortable : true,
```

```
autoWidth : true
            },
columns : [{
header : "First Name",
dataIndex : 'firstName'
                  }, {
header : "Last Name",
dataIndex : 'lastName'
                  }, {
header : "Email",
dataIndex : 'email'
                  }, {
header : "Home OU",
dataIndex : 'homeOrganizationalUnitName'
                  }, {
header : "Login Name",
dataIndex : 'login'
                  }, {
header : "Timezone",
dataIndex : 'timeZoneName'
                  }, {
header : "Language",
dataIndex : 'languageName'
                  }]
         });

var gridConfig = {
renderTo :  MyJSREditDiv,
width : "100%",
layout : 'fit',
store : gridStore,
cm : gridColModel,
loadMask: true,
autoWidth : true,
plugins : expander,
tbar : [{
text : 'Create Person',
iconCls : 'add',
handler : function() {
window.location=createNewPersonActionUrl;
                }
                },'-'],
bbar : new Ext.PagingToolbar({
pageSize : defaultRecordSize,
store : gridStore,
displayInfo : true,
params:{
start: 0,
limit: defaultRecordSize
                           }
                  })
         };

if ('maximized' == portletWindowState) {
gridConfig.height = document.documentElement.clientHeight - 188;
   } elseif ('normal' == portletWindowState) {
var viewConfig = {
forceFit : true
    };
gridConfig.viewConfig = viewConfig;

if (gridHeight && gridHeight > -1) {
gridConfig.height = gridHeight;
    } else {
```

```
gridConfig.autoHeight = true;
    }
   }

var grid = new Ext.grid.GridPanel(gridConfig);
});
```

portlet.xml

ポートレット仕様の例。portlet-class は、init-params のペア contextConfigLocation と nsContentPortlet とともに設定する必要があります（nsContentPortlet は、常に「false」に設定します）。

```
<?xmlversion="1.0"encoding="UTF-8"?>
<!--
Licensed to the Apache Software Foundation (ASF) under one or more
contributor license agreements. See the NOTICE file distributed with
this work for additional information regarding copyright ownership.
The ASF licenses this file to You under the Apache License, Version 2.0
(the "License"); you may not use this file except in compliance with
the License.  You may obtain a copy of the License at

http:www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed  under the  License is distributed on an "AS IS" BASIS,
WITHOUT  WARRANTIES OR CONDITIONS  OF ANY KIND, either  express  or
implied.

See the License for the specific language governing permissions and
limitations under the License.
-->
<portlet-app
xmlns="http:java.sun.com/xml/ns/portlet/portlet-app_2_0.xsd"
version="1.0"
xmlns:xsi="http:www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http:java.sun.com/xml/ns/portlet/portlet-app_1_0.xsd
                    http:java.sun.com/xml/ns/portlet/portlet-app_1_0.xsd">
<portlet>
<description>MyJSR Description</description>
<portlet-name>nsMyJSR</portlet-name>
<display-name>My JSR Portlet</display-name>
<portlet-class>org.springframework.web.portlet.DispatcherPortlet</portlet-class>
<init-param>
<name>contextConfigLocation</name>
<value>/WEB-INF/classes/config/spring/MyJSRApplicationContext.xml</value>
</init-param>
<init-param>
<name>nsContentPortlet</name>
<value>false</value>
</init-param>
<expiration-cache>-1</expiration-cache>
<supports>
<mime-type>text/html</mime-type>
<portlet-mode>VIEW</portlet-mode>
<portlet-mode>EDIT</portlet-mode>
<portlet-mode>HELP</portlet-mode>
</supports>
<portlet-info>
<title>My JSR Portlet</title>
</portlet-info>
</portlet>
</portlet-app>
```

web.xml

サーブレットによる展開記述子の例。ポータル サーバ（この例では Apache Pluto）でサーブレットのマッピングが必要です。

```
<?xmlversion="1.0"encoding="UTF-8"?>
<!DOCTYPEweb-appPUBLIC"-Sun Microsystems, Inc. DTD Web Application
2.3EN""http:java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
<display-name>My JSR Portlet Application</display-name>
<description>My JSR Portlet</description>

<!-- Resources bundle base class -->
<context-param>
<param-name>contextConfigLocation</param-name>
<param-value>
         /WEB-INF/classes/config/spring/MyJSRApplicationContext.xml
</param-value>
</context-param>

<context-param>
<param-name>parameter-name</param-name>
<param-value>parameter-value</param-value>
</context-param>

<servlet>
<servlet-name>ViewRendererServlet</servlet-name>
<servlet-class>
org.springframework.web.servlet.ViewRendererServlet
</servlet-class>
</servlet>

<servlet>
<servlet-name>MyJSR</servlet-name>
<servlet-class>org.apache.pluto.container.driver.PortletServlet</servlet-class>
<init-param>
<param-name>portlet-name</param-name>
<param-value>MyJSR</param-value>
</init-param>
<load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
<servlet-name>ViewRendererServlet</servlet-name>
<url-pattern>/WEB-INF/servlet/view</url-pattern>
</servlet-mapping>

<servlet-mapping>
<servlet-name>MyJSR</servlet-name>
<url-pattern>/PlutoInvoker/nsMyJSR</url-pattern>
</servlet-mapping>

<!- Declare Tag libraries that are used in which are going to use in JSP pages-->
<taglib>
<taglib-uri>http://portals.apache.org/pluto</taglib-uri>
<taglib-location>/WEB-INF/tld/pluto.tld</taglib-location>
</taglib>

<taglib>
<taglib-uri>http://java.sun.com/portlet_2_0</taglib-uri>
<taglib-location>/WEB-INF/tld/portlet_2_0.tld</taglib-location>
</taglib>

<taglib>
<taglib-uri>/WEB-INF/tld/c.tld</taglib-uri>
```

```
<taglib-location>/WEB-INF/tld/c.tld</taglib-location>
</taglib>

<taglib>
<taglib-uri>http://java.sun.com/portlet</taglib-uri>
<taglib-location>/WEB-INF/tld/portlet.tld</taglib-location>
</taglib>

<taglib>
<taglib-uri>http://portals.apache.org/pluto/portlet-el</taglib-uri>
<taglib-location>/WEB-INF/tld/portlet-el.tld</taglib-location>
</taglib>

<taglib>
<taglib-uri>http://portals.apache.org/pluto/portlet-el_2_0</taglib-uri>
<taglib-location>/WEB-INF/tld/portlet-el_2_0.tld</taglib-location>
</taglib>

</web-app>
```

MyJSREdit.jsp

ポートレット編集モード用の JSP。

```
<%
/**
 * Copyright (c) 2012, Cisco Systems, Inc. All rights reserved.
 */
%>

<%@tagliburi="http:java.sun.com/portlet"prefix="portlet"%>
<%@taglibprefix="portlet2"uri="http:java.sun.com/portlet_2_0"%>
<%@taglibprefix="c"uri="/WEB-INF/tld/c.tld"%>

<% String contextPath = request.getContextPath(); %>

<!-- This is for IE -->
<scripttype="text/javascript">
if(document.createStyleSheet) {
document.createStyleSheet('<%= response.encodeURL(contextPath + "/css/MyJSR.css") %>');
}
else {
var styles = "@import url('<%= response.encodeURL(contextPath + "/css/MyJSR.css") %>');";
var newSS=document.createElement('link');
newSS.rel='stylesheet';
newSS.href='data:text/css,'+escape(styles);
document.getElementsByTagName("head")[0].appendChild(newSS);
}
</script>

<!-- This is foFirefox -->
<linkrel="stylesheet"type="text/css"href="<%= response.encodeURL(contextPath +
"/css/MyJSR.css") %>"></link>

<script>
var head = document.getElementsByTagName('head')[0];
var script = document.createElement('script');
script.type = 'text/javascript';
script.src = '<%= response.encodeURL(contextPath + "/js/MyJSREdit.js") %>';
head.appendChild(script);
</script>
```

```
<!-- Write your JSP Code for Portlet Edit here -->
<c:iftest="${portletWindowState == 'NORMAL' or portletWindowState == 'normal'}">
Portlet Mode = <c:outvalue='${portletMode}'/>
Portlet Window State = <c:outvalue='${portletWindowState}'/>
</c:if>

<c:iftest="${portletWindowState == 'MINIMIZED' or portletWindowState == 'minimized'}">
Portlet Mode = <c:outvalue='${portletMode}'/>
Portlet Window State = <c:outvalue='${portletWindowState}'/>
</c:if>

<c:iftest="${portletWindowState == 'MAXIMIZED' or portletWindowState == 'maximized'}">
Portlet Mode = <c:outvalue='${portletMode}'/>
Portlet Window State = <c:outvalue='${portletWindowState}'/>
</c:if>

<divid="MyJSREditDiv-<portlet:namespace/>"class="x-grid-mso"></div>

<script>
var MyJSREditDiv = 'MyJSREditDiv-<portlet:namespace/>';
var addPersonActionUrl = '<portlet2:resourceURL id="addPersonData" escapeXml="false" />';
var personListObj = Ext.util.JSON.decode('<c:out value="${PersonData}"
escapeXml="false"/>');
</script>
```

MyJSRHelp.jsp

ポートレット ヘルプ モード用の JSP。

```
<%
/**
 * Copyright (c) 2012, Cisco Systems, Inc. All rights reserved.
 */
%>

<%@tagliburi="http:java.sun.com/portlet"prefix="portlet"%>
<%@taglibprefix="c"uri="/WEB-INF/tld/c.tld"%>

<% String contextPath = request.getContextPath(); %>

<!-- This is for IE -->
<scripttype="text/javascript">
if(document.createStyleSheet) {
document.createStyleSheet("<%= response.encodeURL(contextPath + "/css/MyJSR.css") %>");
}
else {
var styles = "@import url('<%= response.encodeURL(contextPath + "/css/MyJSR.css") %>');";
var newSS=document.createElement('link');
newSS.rel='stylesheet';
newSS.href='data:text/css,'+escape(styles);
document.getElementsByTagName("head")[0].appendChild(newSS);
}
</script>

<!-- This is foFirefox -->
<linkrel="stylesheet"type="text/css"href="<%= response.encodeURL(contextPath +
"/css/MyJSR.css") %>"></link>

<script>
var head = document.getElementsByTagName('head')[0];
```

```
var script = document.createElement('script');
script.type = 'text/javascript';
script.src = '<%= response.encodeURL(contextPath + "/js/MyJSRHelp.js") %>';
head.appendChild(script);
</script>

<!-- Write your JSP Code for Portlet Help here -->
<c:iftest="${portletWindowState == 'NORMAL' or portletWindowState == 'normal'}">
Portlet Mode = <c:outvalue='${portletMode}'/>
Portlet Window State = <c:outvalue='${portletWindowState}'/>
</c:if>

<c:iftest="${portletWindowState == 'MINIMIZED' or portletWindowState == 'minimized'}">
Portlet Mode = <c:outvalue='${portletMode}'/>
Portlet Window State = <c:outvalue='${portletWindowState}'/>
</c:if>

<c:iftest="${portletWindowState == 'MAXIMIZED' or portletWindowState == 'maximized'}">
Portlet Mode = <c:outvalue='${portletMode}'/>
Portlet Window State = <c:outvalue='${portletWindowState}'/>
</c:if>

<divid="MyJSRHelpDiv-<portlet:namespace/>"class="x-grid-mso"></div>

<script>
var MyJSRHelpDiv = 'MyJSRHelpDiv-<portlet:namespace/>';
</script>
```

MyJSRView_listperson.jsp

ポートレット表示モードの JSP コード。

```
<%
/**
 * Copyright (c) 2012, Cisco Systems, Inc. All rights reserved.
 */
%>

<%@tagliburi="http:java.sun.com/portlet"prefix="portlet"%>
<%@taglibprefix="portlet2"uri="http:java.sun.com/portlet_2_0"%>
<%@taglibprefix="c"uri="/WEB-INF/tld/c.tld"%>

<% String contextPath = request.getContextPath(); %>

<!-- This is for IE -->
<scripttype="text/javascript">
if(document.createStyleSheet) {
document.createStyleSheet('<%= response.encodeURL(contextPath + "/css/MyJSR.css") %>');
}
else {
var styles = "@import url('<%= response.encodeURL(contextPath + "/css/MyJSR.css") %>');";
var newSS=document.createElement('link');
newSS.rel='stylesheet';
newSS.href='data:text/css,'+escape(styles);
document.getElementsByTagName("head")[0].appendChild(newSS);
}
var portletId = 'portlet-container-<c:out value="${portlet}"/>';
var pidPrefix = "portlet-container-";
var portletWindowState = "<c:out value='${portletWindowState}'/>";
var portletMode = "<c:out value='${portletMode}'/>";
var defaultRecordSize = <c:out value='${defaultRecordSize}'/>;
```

```
var nsappPersonViewDiv = 'nsappPersonViewDiv-<portlet:namespace/>';
var pagingUrl = '<portlet2:resourceURL escapeXml="false" />';
var connectionTimeOut = <c:out value='${connectionTimeOut}'/>;
</script>

<!-- This is foFirefox -->
<linkrel="stylesheet"type="text/css"href="<%= response.encodeURL(contextPath +
"/css/MyJSR.css") %>"></link>

<script>
var head = document.getElementsByTagName('head')[0];
var script = document.createElement('script');
script.type = 'text/javascript';
script.src = '<%= response.encodeURL(contextPath + "/js/MyJSRView.js") %>';
head.appendChild(script);
</script>

<!-- Write your JSP Code for Portlet Edit here -->
<c:iftest="${portletWindowState == 'NORMAL' or portletWindowState == 'normal'}">
<!--PortletMode = <c:out value='${portletMode}'/>
Portlet Window State = <c:out value='${portletWindowState}'/> -->
</c:if>

<c:iftest="${portletWindowState == 'MINIMIZED' or portletWindowState == 'minimized'}">
<!--Portlet Mode = <c:out value='${portletMode}'/>
Portlet Window State = <c:out value='${portletWindowState}'/> -->
</c:if>

<c:iftest="${portletWindowState == 'MAXIMIZED' or portletWindowState == 'maximized'}">
<!--Portlet Mode = <c:out value='${portletMode}'/>
Portlet Window State = <c:out value='${portletWindowState}'/>-->
</c:if>

<!-Un-comment this to get User Info from the Portlet Controller that sets user info in
model
Logged In User (<c:out value="${userName}" escapeXml="false"/>): <c:out
value="${firstName}" escapeXml="false"/> ,  <c:out value="${lastName}"
escapeXml="false"/> , <c:out value="${PersonId}"
escapeXml="false"/> , <c:out value="${HomeOUId}" escapeXml="false"/>
-->

<!-Write declare divs to display user info -->
<div>Logged In User (<div id="MyJSRLoginNameDiv"
style="text-align:right;display:inline;width:100%;"><span
style="padding-left:20px;"></span></div>) : <div id="MyJSRFirstNameDiv"
style="text-align:right;display:inline;width:100%;"><span></span></div> , 
<divid="MyJSRLastNameDiv" style="text-align:right;display:inline;width:100%;"><span
style="padding-left:20px;"></span></div>
</div>
<div>HomeOUId : <div id="MyJSRHomeOUDiv"
style="text-align:right;display:inline;width:100%;"><span
style="padding-left:20px;"></span></div> , 
PersonId : <div id="MyJSRPortletPersonIDDiv"
style="text-align:right;display:inline;width:100%;"><span
style="padding-left:20px;"></span>
</div>

<divid="MyJSREditDiv-<portlet:namespace/>"class="x-grid-mso"></div>

<script>
var MyJSREditDiv = 'MyJSREditDiv-<portlet:namespace/>';
var createNewPersonActionUrl = '<portlet:renderURL><portlet:param name="formAction"
value="createNewPerson" /></portlet:renderURL>';
var addPersonActionUrl = '<portlet2:resourceURL id="addPersonData" escapeXml="false" />';
```

```
var personListObj = Ext.util.JSON.decode('<c:out value="${PersonData}"
escapeXml="false"/>');
</script>
```

MyJSRView_updateperson.jsp

ユーザ更新操作のデモ用 JSP コードの例。

```
<%
/**
 * Copyright (c) 2012, Cisco Systems, Inc. All rights reserved.
 */
%>

<%@tagliburi="http:java.sun.com/portlet"prefix="portlet"%>
<%@taglibprefix="portlet2"uri="http:java.sun.com/portlet_2_0"%>
<%@taglibprefix="c"uri="/WEB-INF/tld/c.tld"%>

<%String contextPath = request.getContextPath(); %>

<!-- This is foFirefox -->
<linkrel="stylesheet"type="text/css"href="<%= response.encodeURL(contextPath +
"/css/MyJSR.css") %>"></link>

<!-- This is for IE -->
<scripttype="text/javascript">
if(document.createStyleSheet) {
document.createStyleSheet("<%= response.encodeURL(contextPath + "/css/MyJSR.css") %>");
   }
else {
var styles = "@import url('<%= response.encodeURL(contextPath + "/css/MyJSR.css") %>');";
var newSS=document.createElement('link');
newSS.rel='stylesheet';
newSS.href='data:text/css,'+escape(styles);
document.getElementsByTagName("head")[0].appendChild(newSS);
   }
</script>

<script>
var head = document.getElementsByTagName('head')[0];
var script = document.createElement('script');
script.type = 'text/javascript';
script.src = '<%= response.encodeURL(contextPath + "/js/MyJSRCreatePersonView.js") %>';
head.appendChild(script);
</script>

<!-- Write your JSP Code for Portlet View here -->
<c:iftest="${portletWindowState == 'NORMAL' or portletWindowState == 'normal'}">
<!--Portlet Mode = <c:out value='${portletMode}'/>
Portlet Window State = <c:out value='${portletWindowState}'/> -->
</c:if>

<c:iftest="${portletWindowState == 'MINIMIZED' or portletWindowState == 'minimized'}">
<!--Portlet Mode = <c:out value='${portletMode}'/>
Portlet Window State = <c:out value='${portletWindowState}'/> -->
</c:if>

<c:iftest="${portletWindowState == 'MAXIMIZED' or portletWindowState == 'maximized'}">
<!--Portlet Mode = <c:out value='${portletMode}'/>
Portlet Window State = <c:out value='${portletWindowState}'/>-->
```

```
</c:if>

<divid="MyJSREditDiv-<portlet:namespace/>"class="x-grid-mso"></div>
<script>
var MyJSREditDiv = 'MyJSREditDiv-<portlet:namespace/>';
var addPersonActionUrl = '<portlet2:resourceURL id="addPersonData" escapeXml="false" />';
var viewPersonUrl = '<portlet:renderURL></portlet:renderURL>';
var personListObj = Ext.util.JSON.decode('<c:out value="${PersonData}"
escapeXml="false"/>');
</script>
```

MyJSRController.java

java ポートレット コントローラを作成する一般的な手順は、次のとおりです。

手順 1　3 つのポートレット モード（表示（View）、編集（Edit）、ヘルプ（Help））のハンドラ コードを記述します。

手順 2　3 つのポートレット ビュー（通常（Normal）、最小化（Minimized）、最大化（Maximized））のハンドラ コードを記述します。

手順 3　サービス カタログ（Service Catalog）エンティティを処理/表示する JSR ポートレットでは、nsAPI クライアントを使用してポートレット コントローラで関連する REST API を呼び出すことができます。

a. nsAPI クライアント API の参照を取得します。

b. nsAPI クライアントを呼び出して、必要なサービス カタログ（Service Catalog）エンティティのインスタンスのリストを取得します。

c. オプションで、現在ログインしているユーザの詳細（ユーザ ID、名、姓など）を取得します。

手順 4　インスタンスをグリッドまたはその他の形式でレンダリングします（この例では、Ext JS グリッドの nsAPI でページングをする方法も示します）。

```
package com.myjsr;

importjava.io.IOException;
importjava.util.ArrayList;
importjava.util.HashMap;
importjava.util.Iterator;
importjava.util.List;
import java.util.Map;
importjava.util.Properties;

importjavax.portlet.RenderRequest;
importjavax.portlet.RenderResponse;
importjavax.portlet.ResourceRequest;
importjavax.portlet.ResourceResponse;

importnet.sf.json.JSON;
importnet.sf.json.JSONSerializer;
importjavax.portlet.ActionRequest;
importjavax.servlet.http.HttpSession;
importjavax.portlet.PortletURL;
importorg.apache.commons.collections.map.MultiValueMap;
importorg.apache.commons.lang.StringEscapeUtils;
importorg.apache.commons.lang.StringUtils;
```

```
importorg.springframework.ui.Model;
importorg.springframework.web.portlet.ModelAndView;
importorg.springframework.web.bind.annotation.ModelAttribute;
importorg.springframework.web.bind.annotation.RequestMapping;
importorg.springframework.web.portlet.bind.annotation.ResourceMapping;
importorg.springframework.web.bind.annotation.RequestParam;

importcom.newscale.comps.conf.domain.AppParamUtil;
importcom.newscale.nsapi.directory.person.Person;
importcom.newscale.nsapi.directory.person.PersonList;
importcom.newscale.nsapiclient.NSApiClient;
importcom.newscale.nsapiclient.NSApiClientConstants;
importcom.newscale.nsapiclient.NSApiClientFactory;
importcom.newscale.portlets.GenericNewScaleSpringPortletBase;


/**
*MyJSRController
*/
publicclass MyJSRController extends GenericNewScaleSpringPortletBase  {
privatestaticfinal String configPropsFile = "jsrportlet.properties";
privatestaticfinal String viewPageList = "MyJSRView_listperson";
privatestaticfinal String viewPageUpdate = "MyJSRView_updateperson";
privatestaticfinal String editPage = "MyJSREdit";
privatestaticfinal String helpPage = "MyJSRHelp";
private NSApiClient nsApiClient = getNSApiClient();

public NSApiClient getNsApiClient() {
return nsApiClient;
  }

public void setNsApiClient(NSApiClient nsApiClient) {
this.nsApiClient = nsApiClient;
  }

public String viewNormal(RenderRequest request, RenderResponse response, Model model) {
try {
super.viewNormal(request, response, model);
    getLoginUsername(request , model);
    } catch(Exception e){
e.printStackTrace();
    }

return doView(request, response, model);
  }

public String viewMinimized(RenderRequest request, RenderResponse response, Model model) {
try {
super.viewMinimized(request, response, model);
   } catch(Exception e){
e.printStackTrace();
   }
return viewPageList;
  }

public String viewMaximized(RenderRequest request, RenderResponse response, Model model) {
try {
super.viewMaximized(request, response, model);
    getLoginUsername(request , model);
   } catch(Exception e){
e.printStackTrace();
   }
```

```
return doView(request, response, model);
  }

    private void getLoginUsername(RenderRequest request , Model model){
        Properties properties = getConfigProperties(configPropsFile);
    nsApiClient.login(properties.getProperty("BASE_URL"),
request.getPortletSession().getId());
             // Get Currently Logged-in user from nsAPI client
         Person persons  = nsApiClient.getDirectory().getCurrentUser();

// Set user info into model so that JSP can access it
model.addAttribute("PersonID",  persons.getPersonId());
    model.addAttribute("HomeOUId",  persons.getHomeOrganizationalUnitId());
    model.addAttribute("firstName",  persons.getFirstName());
    model.addAttribute("lastName",  persons.getLastName());
    model.addAttribute("userName",  persons.getLogin());
  }

private String doView(RenderRequest request, RenderResponse response, Model model) {
try {
    Properties properties = getConfigProperties(configPropsFile);
nsApiClient.login(properties.getProperty("BASE_URL"),
request.getPortletSession().getId());

int defaultRecordSize = AppParamUtil.getInstance().getMaxMaxPagingSizeInNSApi();
model.addAttribute("defaultRecordSize", "" + defaultRecordSize);
int connectionTimeOut = 0;
if (AppParamUtil.getInstance().isParamExists((AppParamUtil.SESSION_TIMEOUT))) {
connectionTimeOut =
AppParamUtil.getInstance().getIntegerParam(AppParamUtil.SESSION_TIMEOUT);
    }
if (connectionTimeOut < 1) {
connectionTimeOut = 20;
    }
model.addAttribute("connectionTimeOut", "" + connectionTimeOut * 1000 * 60);

    String formAction = request.getParameter("formAction");
    String personIdStr = request.getParameter("personId");

if (null != formAction && formAction.equals("createNewPerson") ) {
    showAddPersonPage(request, response, model);
    return viewPageUpdate;
    } elseif(null != personIdStr){
    editPerson(request, response, model, new Integer(personIdStr).intValue());
    return viewPageUpdate;
    } else {
    return viewPageList;
    }
     } catch (Exception e){
        e.printStackTrace();
     }
    returnnull;
  }

@RequestMapping("VIEW")
  @ResourceMapping
public ModelAndView doPagingOrSavePerson(ResourceRequest request, ResourceResponse
response, Person person)
throws Exception {
    String instanceName= getInstanceName(request.getWindowID());
 Save Button is clicked while adding person
if ("createNewPerson".equals(request.getParameter("formAction")) || null !=
request.getParameter("personId")) {
addPersonData(person ,request, response );
```

```
    } elseif (null != request.getParameter("start") &&null !=
request.getParameter("limit")) {  Paging
int startInt = Integer.parseInt(request.getParameter("start")) + 1; extjs sends 1 less
than what nsAPI wants
int limit = Integer.parseInt(request.getParameter("limit")) + 1; extjs sends 1 less than
what nsAPI wants
try {
if (request.getWindowState().equals(request.getWindowState().NORMAL)) {
doPagingInternal(request, response, instanceName, 1, startInt, limit);
        }
if (request.getWindowState().equals(request.getWindowState().MAXIMIZED)) {
doPagingInternal(request, response, instanceName, 2, startInt, limit);
        }
      } catch (Exception e) {
e.printStackTrace();
      }
    }
returnnull;
  }

private void doPagingInternal(ResourceRequest request, ResourceResponse response, String
portletInst,
int windowStateInt, int start, int limit) throws Exception {
    Map<String, Object> jsonMap = new HashMap<String, Object>();
    List recordList = newArrayList();
int totalCount = 1;
    String editPersonUrl = request.getParameter("editPersonUrl");
 MultiValueMap paramsmap = new MultiValueMap();
    paramsmap.put(NSApiClientConstants.QUERYPARAM_START_ROW, "" + start);
    paramsmap.put(NSApiClientConstants.QUERYPARAM_RECORD_SIZE, "" + limit);
    PersonList personList = nsApiClient.getDirectory().getPeople(paramsmap);
if(personList.getPeople() != null) {
for(Iterator iterator = personList.getPeople().iterator(); iterator.hasNext();) {
        Person portalPerson = (Person) iterator.next();
portalPerson.setPersonURL(StringEscapeUtils.escapeXml(portalPerson.getPersonURL()));
        PortletURL editPersonURL = response.createRenderURL();
editPersonURL.setParameter("personId", "" + portalPerson.getPersonId());
        String firstNameUrl = "<a href='" + editPersonURL.toString() + "'>" +
portalPerson.getFirstName() + "</a>";
        String lastNameUrl = "<a href='" + editPersonURL.toString() + "'>" +
portalPerson.getLastName()  + "</a>";
portalPerson.setFirstName(firstNameUrl);
portalPerson.setLastName(lastNameUrl);
recordList.add(portalPerson);
      }
jsonMap.put("success", "true");
jsonMap.put("results", personList.getTotalCount());
jsonMap.put("rows", recordList);
      JSON json = (JSON) JSONSerializer.toJSON(jsonMap);
      String jsonStr = json.toString();

response.setContentType("text/plain");
response.getPortletOutputStream().write(jsonStr.getBytes());
response.getPortletOutputStream().flush();
    }
  }


private ModelAndView addPersonData(@ModelAttribute("personData") Person person,
      ResourceRequest request, ResourceResponse response) throws Exception {
 Add Person from Form Data in Request
    Map jsonMap = newHashMap();
try {
Person  Updateperson = nsApiClient.getDirectory().updatePerson(person);
```

```
jsonMap.put("success", "true");
jsonMap.put("successMsg", "Person Added/Updated Successfully");
jsonMap.put("rows", Updateperson);
      JSON json = (JSON) JSONSerializer.toJSON(jsonMap);
      String jsonStr = json.toString();

response.setContentType("text/plain");
response.getPortletOutputStream().write(jsonStr.getBytes());
response.getPortletOutputStream().flush();
    } catch (Exception e) {
e.printStackTrace();
jsonMap.put("success", "false");
jsonMap.put("errorMsg", "Person Add/Update Failed : " + e.getMessage());

      JSON json2 = (JSON) JSONSerializer.toJSON(jsonMap);
      String jsonStr2 = json2.toString();

response.setContentType("text/plain");
response.getPortletOutputStream().write(jsonStr2.getBytes());
response.getPortletOutputStream().flush();
    }

returnnull;
  }

private void editPerson(RenderRequest request, RenderResponse response, Model model, int
personId) {
int person = personId;
    Map<String, Object> jsonMap = new HashMap<String, Object>();
try {
Person  persons = nsApiClient.getDirectory().getPersonById(person);
persons.setPersonURL(StringEscapeUtils.escapeXml(persons.getPersonURL()));
     JSON json = (JSON) JSONSerializer.toJSON(persons);
     String jsonStr = json.toString();
model.addAttribute("PersonData", jsonStr);
    } catch (Exception e) {
e.printStackTrace();
    }
  }

private String showAddPersonPage(RenderRequest request, RenderResponse response, Model
model) {
    Person DummyPerson = newPerson();
DummyPerson.setPersonURL(StringEscapeUtils.escapeXml(DummyPerson.getPersonURL()));
    JSON json1 = (JSON) JSONSerializer.toJSON(DummyPerson);
    String jsonStr1 = json1.toString();
model.addAttribute("PersonData", jsonStr1);

return viewPageUpdate;
  }

public String editNormal(RenderRequest request, RenderResponse response, Model model) {
try {
super.editNormal(request, response, model);
    } catch(Exception e){
e.printStackTrace();
    }
return editPage;
  }

public String editMinimized(RenderRequest request, RenderResponse response, Model model) {
try {
super.editMinimized(request, response, model);
    } catch(Exception e){
```

```
e.printStackTrace();
    }
    return editPage;
      }

public String editMaximized(RenderRequest request, RenderResponse response, Model model) {
try {
super.editMaximized(request, response, model);
   } catch(Exception e){
e.printStackTrace();
   }
return editPage;
  }

public String helpNormal(RenderRequest request, RenderResponse response, Model model) {
try {
super.helpNormal(request, response, model);
    } catch(Exception e){
e.printStackTrace();
    }
return helpPage;
  }

public String helpMinimized(RenderRequest request, RenderResponse response, Model model) {
try {
super.helpMinimized(request, response, model);
    } catch(Exception e){
e.printStackTrace();
    }
return helpPage;
  }

public String helpMaximized(RenderRequest request, RenderResponse response, Model model) {
try {
super.helpMaximized(request, response, model);
    } catch(Exception e){
e.printStackTrace();
    }
return helpPage;
  }

private NSApiClient getNSApiClient() {
return NSApiClientFactory.getInstance();
  }
}
```

MyJSRApplicationContext.xml

ポートレット用の Spring アプリケーション コンテキスト XML。

```
<?xmlversion="1.0"encoding="UTF-8"?>
<beansxmlns="http:www.springframework.org/schema/beans"
xmlns:xsi="http:www.w3.org/2001/XMLSchema-instance"xmlns:p="http:www.springframework.org/s
chema/p"
xmlns:context="http:www.springframework.org/schema/context"
xsi:schemaLocation="
    http:www.springframework.org/schema/beans
http:www.springframework.org/schema/beans/spring-beans-2.5.xsd
    http:www.springframework.org/schema/context
http:www.springframework.org/schema/context/spring-context-2.5.xsd">
```

```
<beanid="viewResolver"
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
<propertyname="cache"value="true"/>
<propertyname="viewClass"
value="org.springframework.web.servlet.view.JstlView"/>
<propertyname="prefix"value="/WEB-INF/jsp/"/>
<propertyname="suffix"value=".jsp"/>
</bean>
<context:annotation-config/>
<bean
class="org.springframework.web.portlet.mvc.annotation.DefaultAnnotationHandlerMapping">
<propertyname="interceptors">
<bean
class="org.springframework.web.portlet.handler.ParameterMappingInterceptor"/>
</property>
</bean>
<beanid="MyJSRController"class="com.myjsr.MyJSRController">
</bean>
</beans>
```

jsrportlet.properties

nsAPI で使用する MyJSRerver の URL。

(注)　クラスタ環境でポートレットがサービス カタログ（Service Catalog）アプリケーション URL を参照する場合は、URL を「http://localhost:<port>/RequestCenter」として指定します（<port> はクラスタの各ノードで使用するポート番号）。つまり、URL は「http:<host_name>/RequestCenter」として指定しないでください（<host_name> は Web サーバまたはクラスタのいずれかのホストのコンピュータ名）。

```
#(Port number and host has to changed as per the application server).
BASE_URL=http:localhost:8088/RequestCenter
```

Log4j.properties

```
log4j.rootCategory=INFO,CONSOLE

log4j.appender.CONSOLE=org.apache.log4j.ConsoleAppender
log4j.appender.CONSOLE.layout=org.apache.log4j.PatternLayout
log4j.appender.CONSOLE.layout.ConversionPattern=%d{ABSOLUTE}%-5p[%c{1}:%L]%m%n
```

jboss-deployment-structure.xml

```
<jboss-deployment-structure>
  <deployment>
     <dependencies>
          <module name="javax.portlet" slot="main" export="true"/>
          <module name="org.apache.pluto.container.om" export="true"/>
          <module name="org.apache.pluto.container.driver" export="true"/>
          <module name="org.apache.pluto.tags" export="true"/>
      </dependencies>
  </deployment>
</jboss-deployment-structure>
```

JSR ポートレット コントローラのコンパイル

ポートレット コントローラをコンパイルするときは、従属ライブラリをクラスパスに含めます。すべてのライブラリのリストについては、「ポートレットの構造とパッケージ化」を参照してください。

ポートレットの導入

導入手順は、使用するアプリケーション サーバによって異なります。一般的な注意点として、JSR ポートレットは通常の Web アプリケーションのように、アプリケーション サーバの管理コンソールを使用して導入できます。

JSR ポートレットを導入する方法、および導入後にポータル ページでポートレットを使用する方法の詳細については、『*Cisco Prime サービス カタログ(Service Catalog)Designer Guide*』を参照してください。

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）

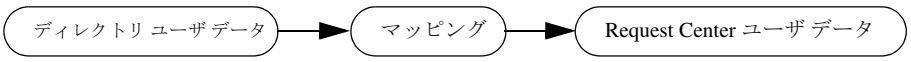
設定	値	説明
Protocol	• LDAP	現時点でサポートされているプロトコルは LDAP だけです。他のプロトコルを使用してディレクトリ情報を格納する場合は、この情報にアクセスするためのカスタム コードを作成する必要があります。
Server Product	• Sun™ ONE Directory • Microsoft® Active Directory® • IBM® Tivoli®	使用するディレクトリ サーバ製品を選択します。現時点でサーバがサポートされていない場合は、サーバにアクセスするためのカスタム コードを作成し、ディレクトリ情報を抽出する必要があります。
Authentication 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。

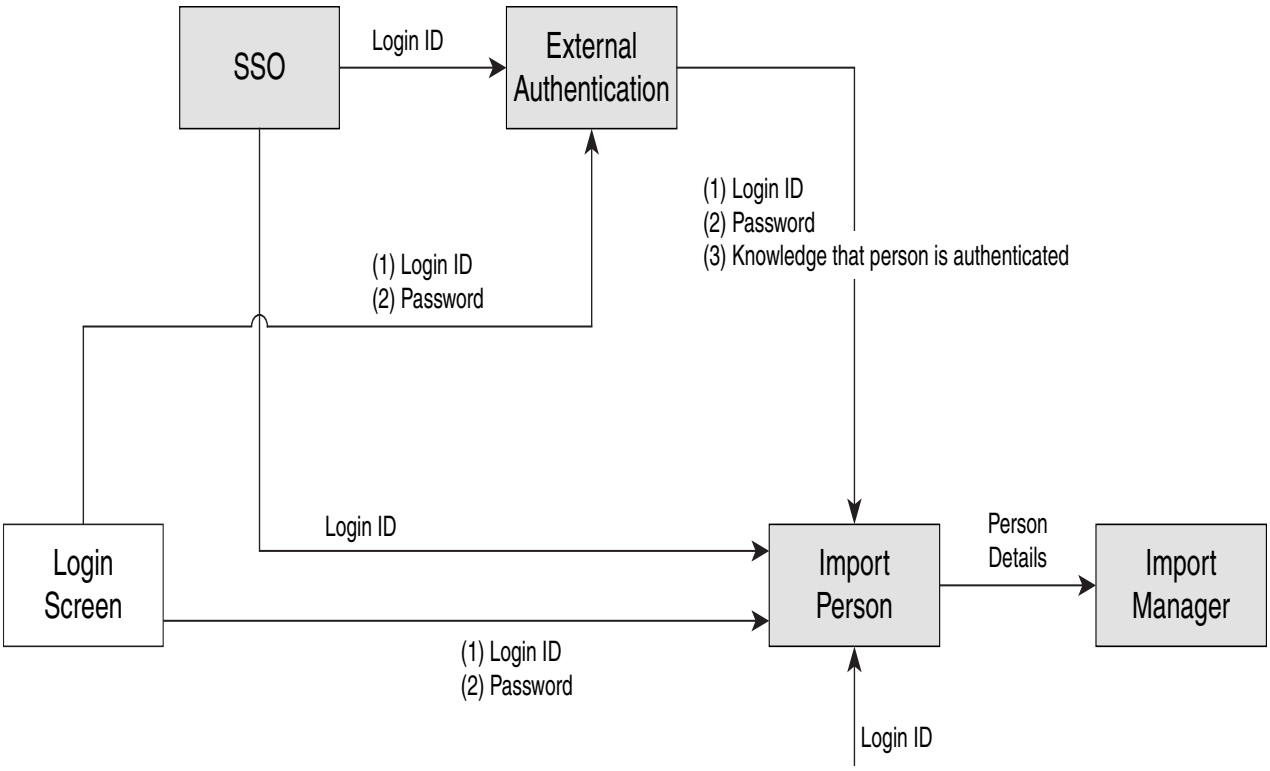
ディレクトリ フレームワークには次の操作が含まれます。

- **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

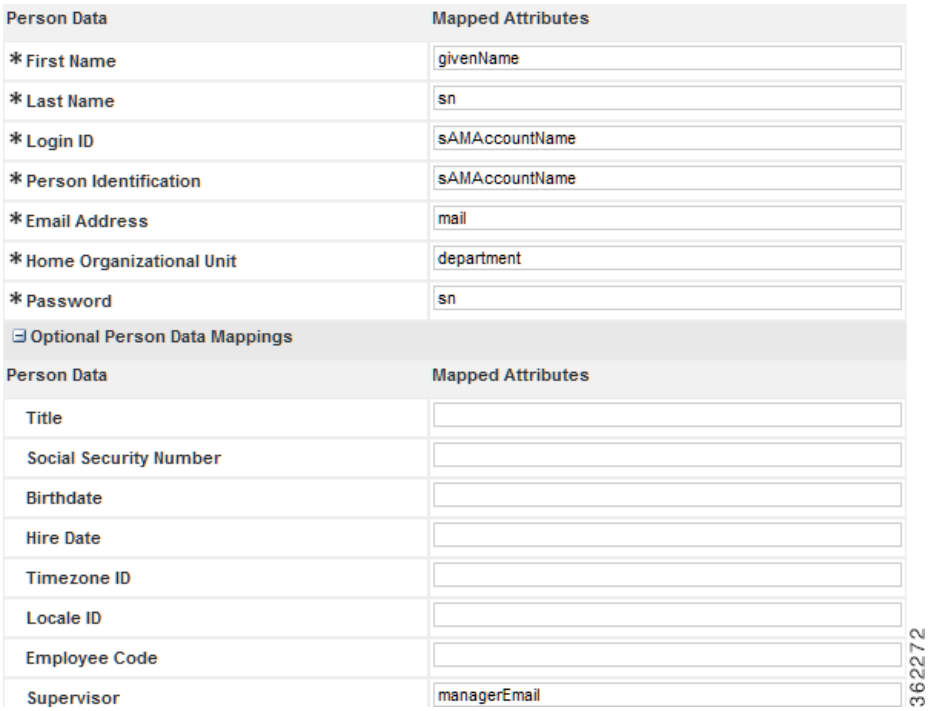
サービス カタログ（Service Catalog）では、承認と確認を動的に割り当てることができます。たとえば、要求のドル値が指定されたしきい値を超えている場合、特定の部署の部長による承認が必要な場合があります。別の要求では、要求者の直属の上司による確認が必要な場合があります。

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。

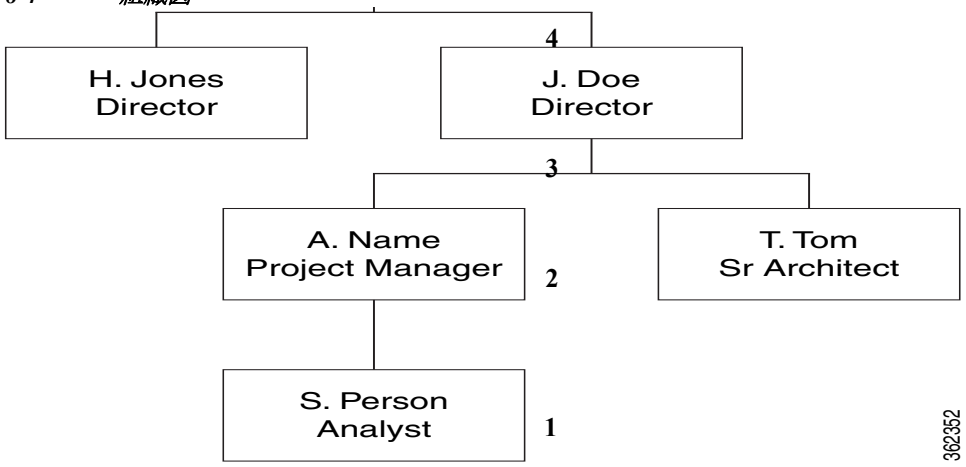
表 8-9　マネージャ ID のキー フィールド

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。

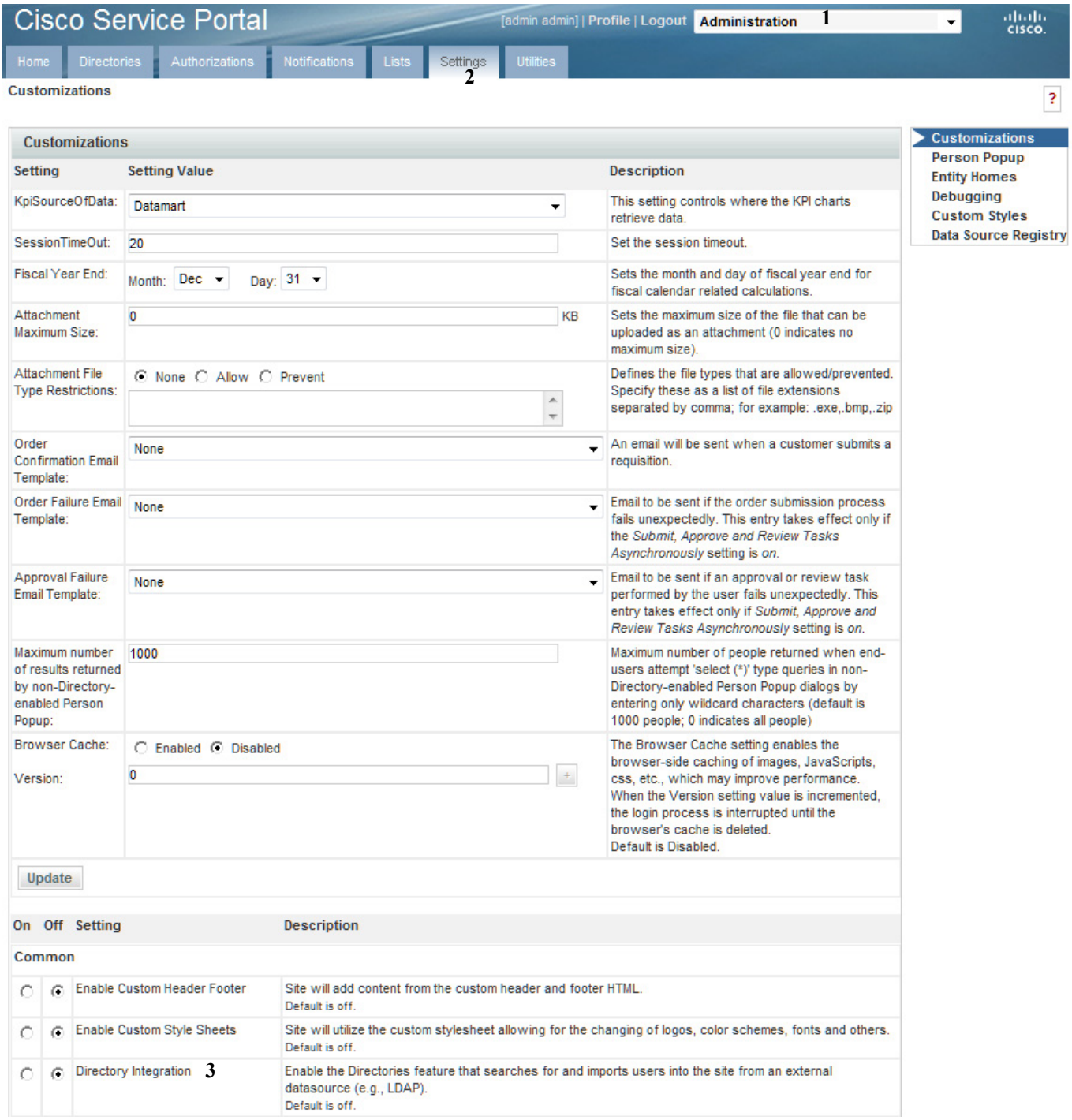
手順 2　[設定（Settings）] タブをクリックします。

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

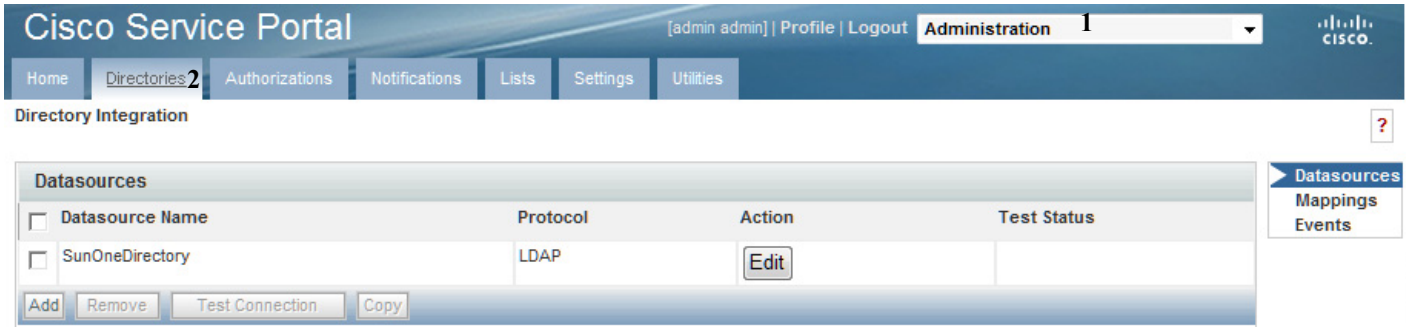


1	[Administration] モジュール	**3**	ディレクトリ統合の設定
2	[設定(Settings)] タブ		

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

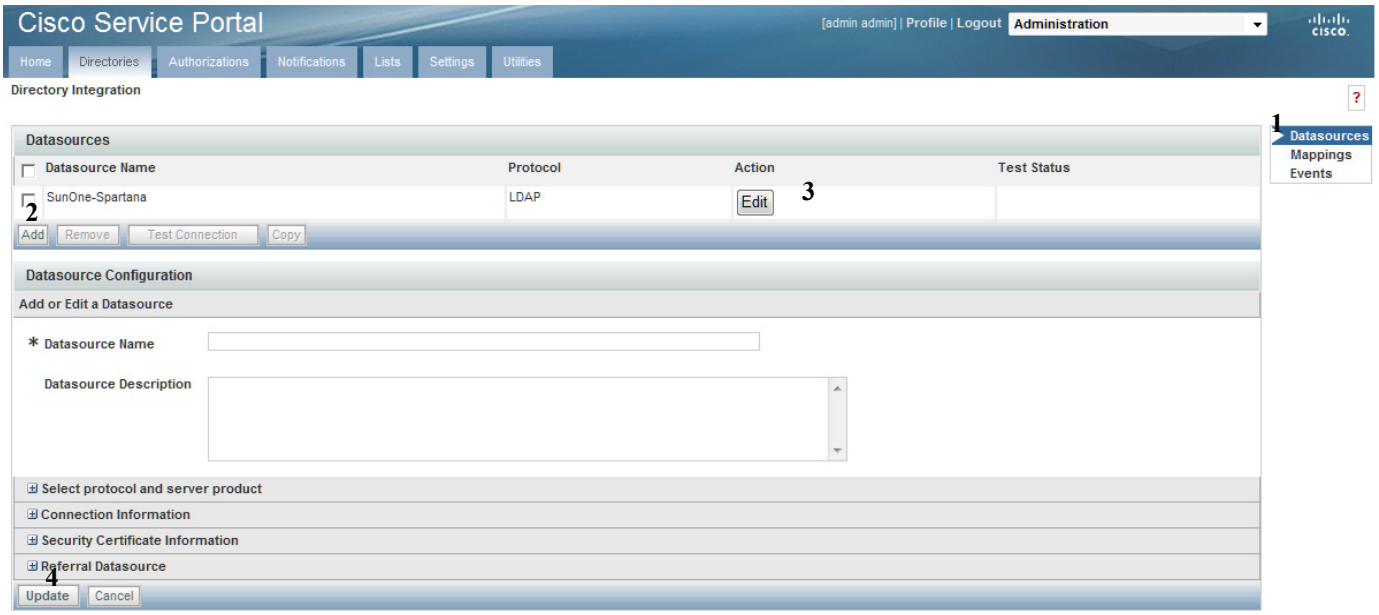
以降の項では、データソース特有の情報設定について解説します。タスクには次のものがあります。

- **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。

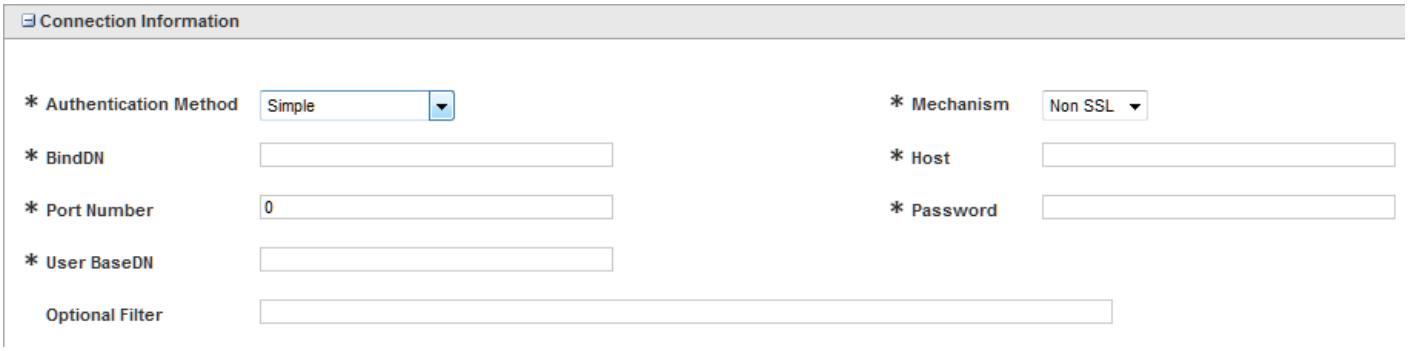
手順 4　[データソース名(Datasource name)]、[データソースの説明(Datasource Description)]、および必要な設定に入力します。隣接領域のすべての設定にアクセスするには ⊞、をクリックします。これらの設定の詳細については、データソース ワークシートを参照するか、または次の項を参照してください。

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。

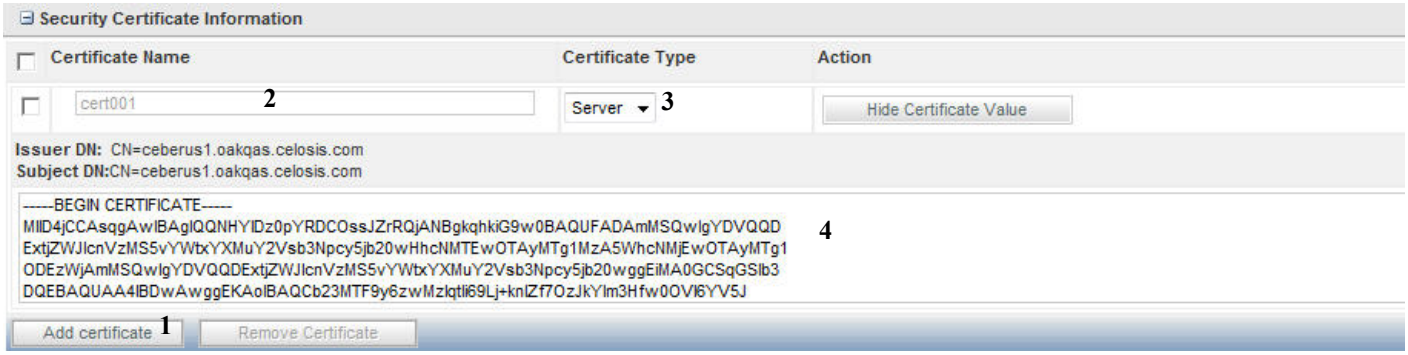
図 8-8　　接続情報の設定



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。

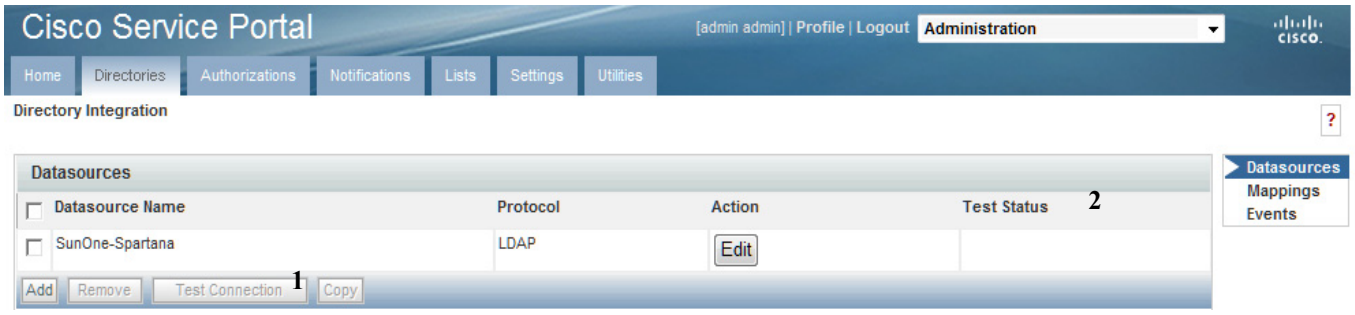
手順 5　[照会用データソース（Referral Datasource）] 領域が表示されます。[データソース名（Datasource Name）] ドロップダウン メニューからデータソース名を選択し、[マッピング名（Mapping Name）] ドロップダウン メニューからマッピング名を選択します。

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。

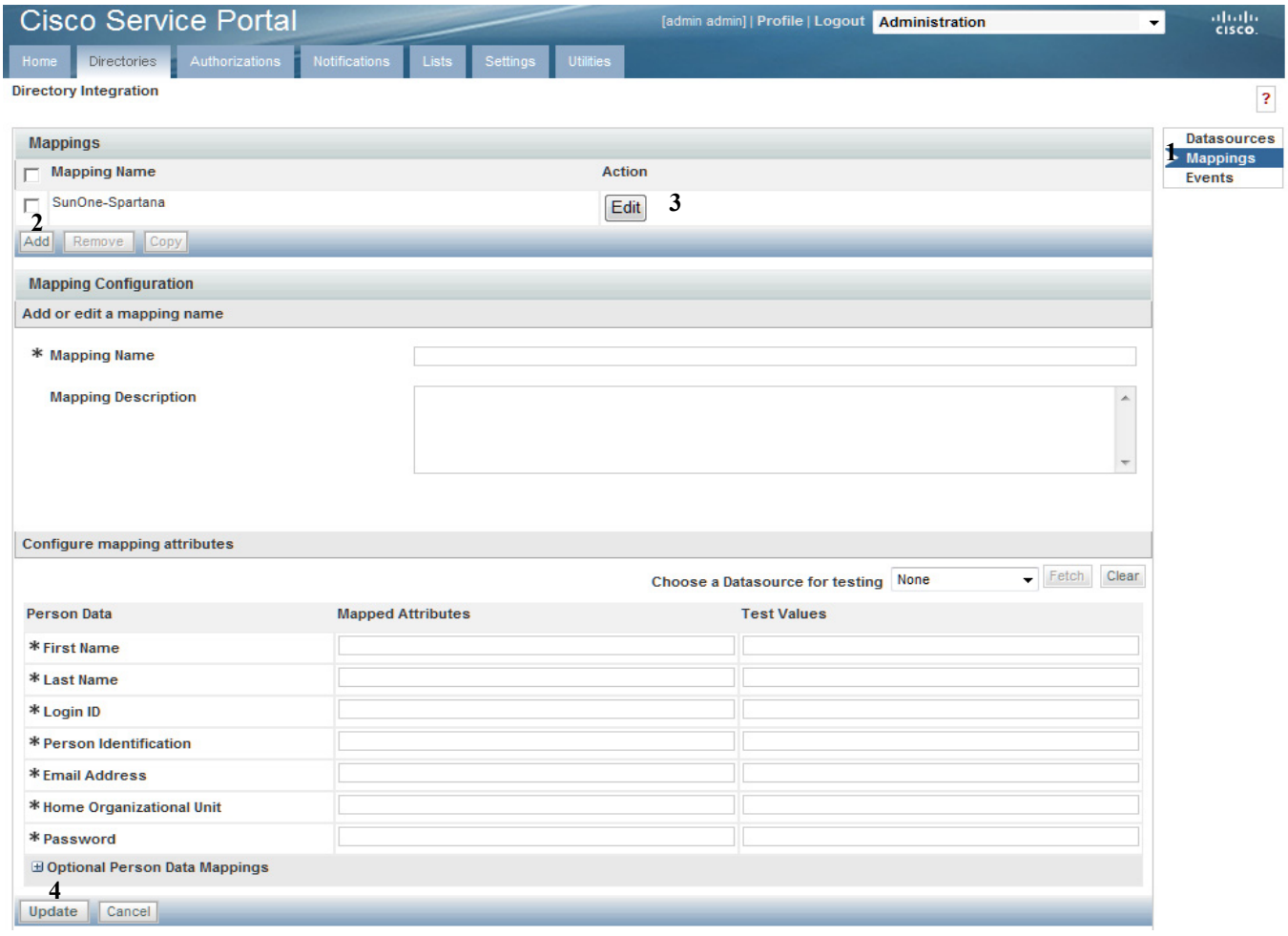
接続が成功した場合は [テストステータス(Test Status)] カラムに OK と表示され、接続が失敗した場合は  が表示されます。

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。

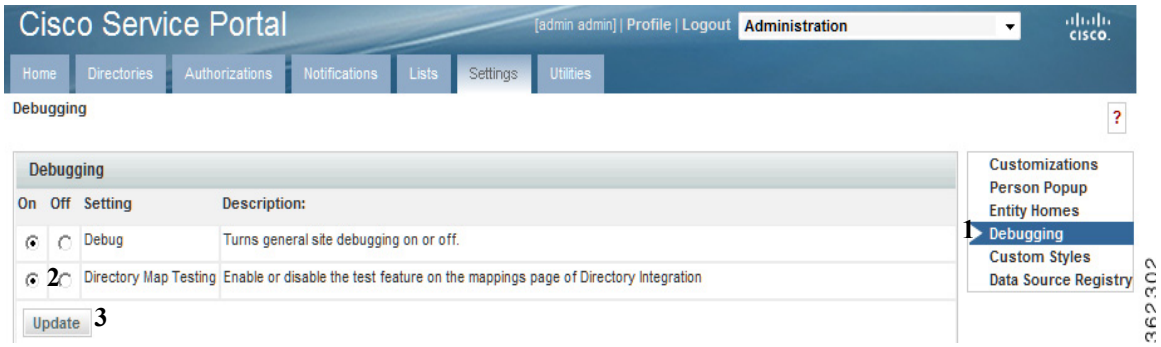
データ マッピング テスト機能の使用には、次の処理が含まれます。

- 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。

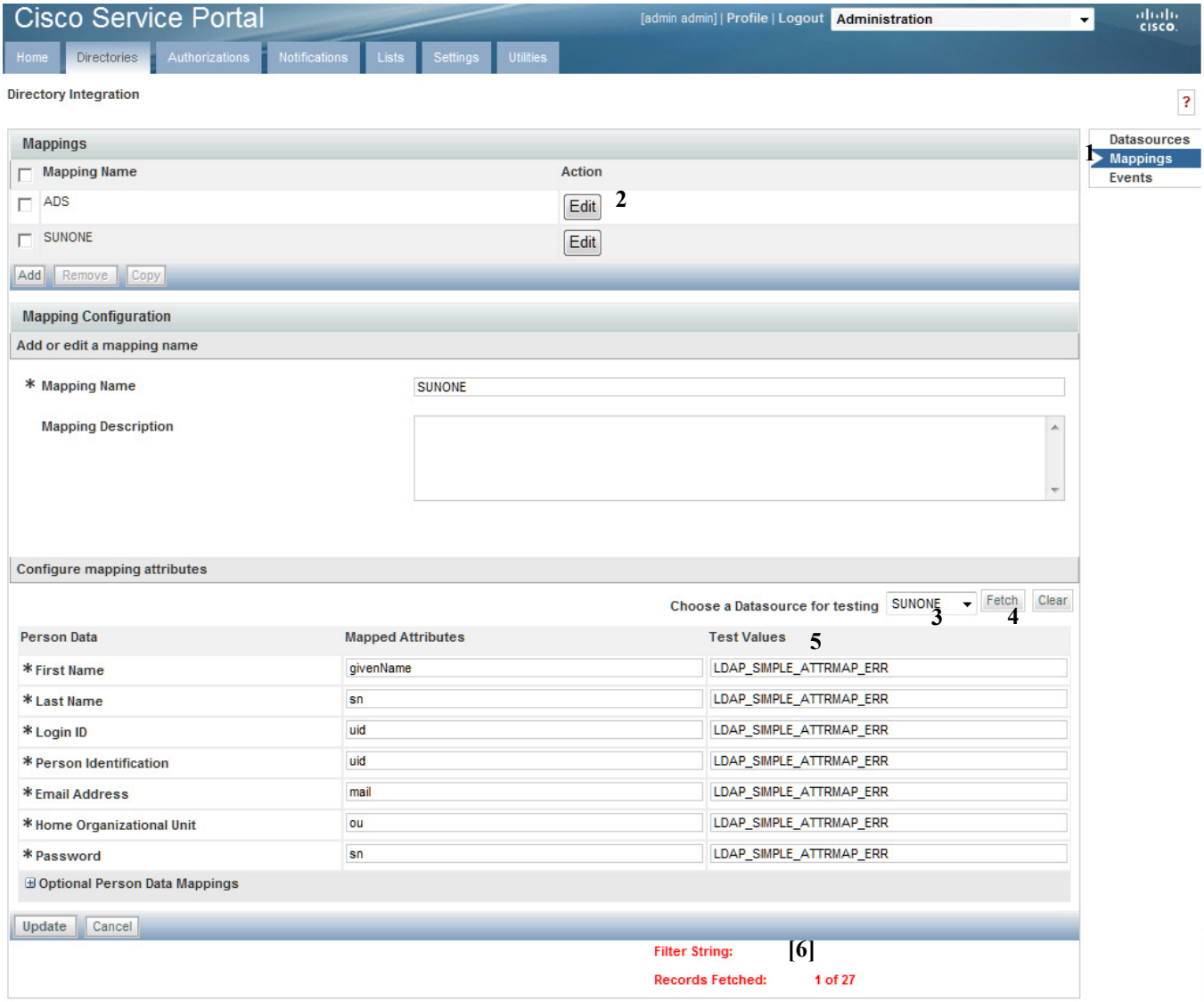
手順 4　[更新（Update）] をクリックします。

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。

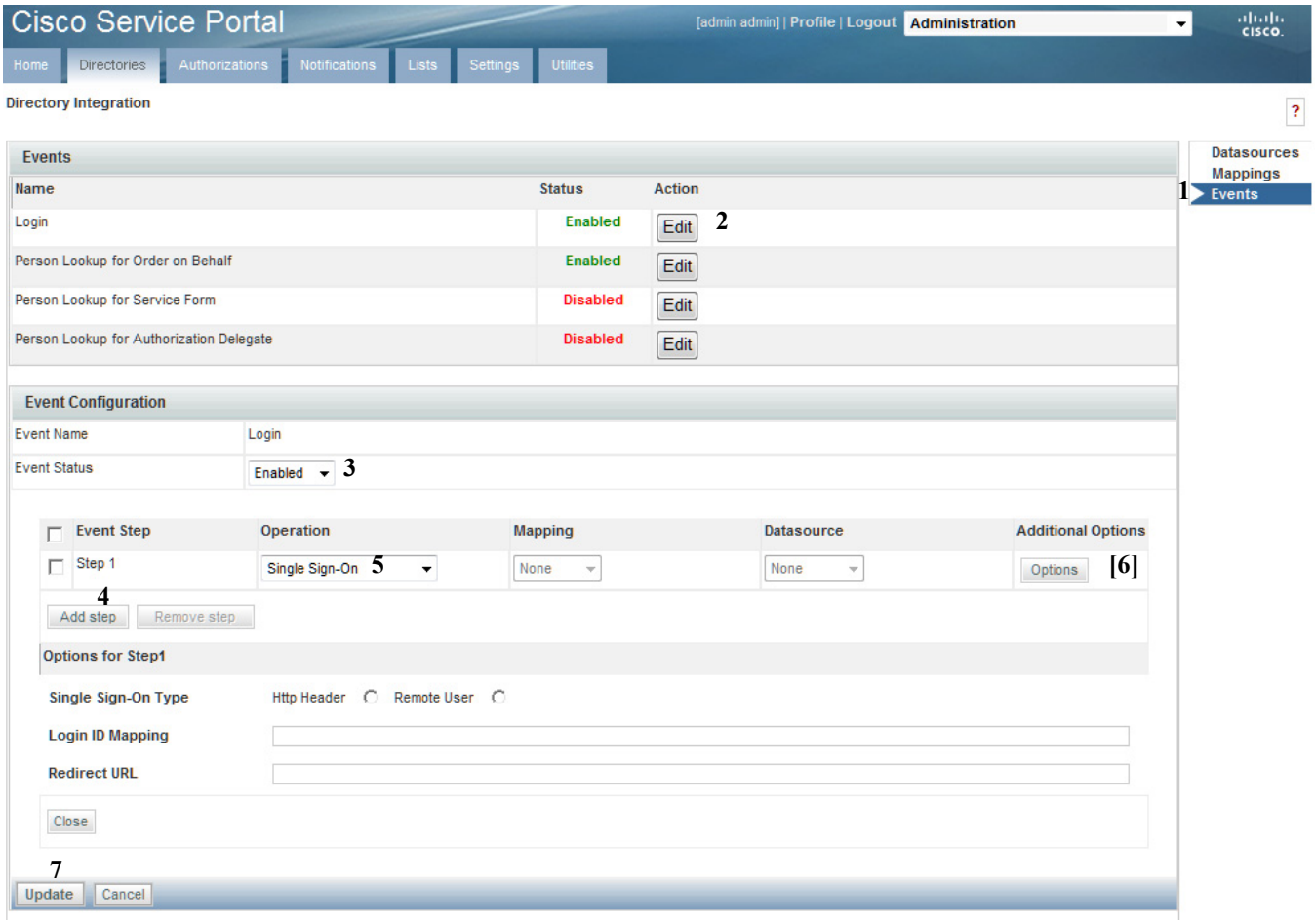
- SSO や EUA など、いくつかの操作は一部のイベント タイプに適用できませんが、このメニューではすべての操作を使用できます。

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

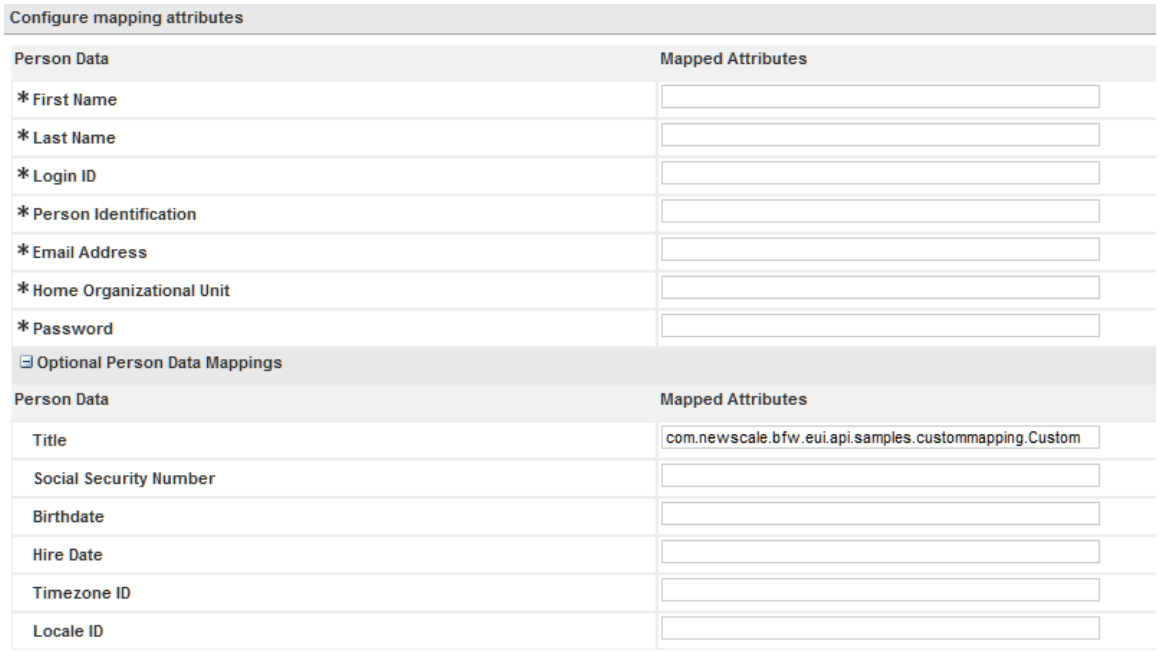
簡易、複合、または正規表現の各属性マッピングでは不十分な場合は、ディレクトリ統合属性マッピングでカスタム Java クラスを使用することができます。

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
```

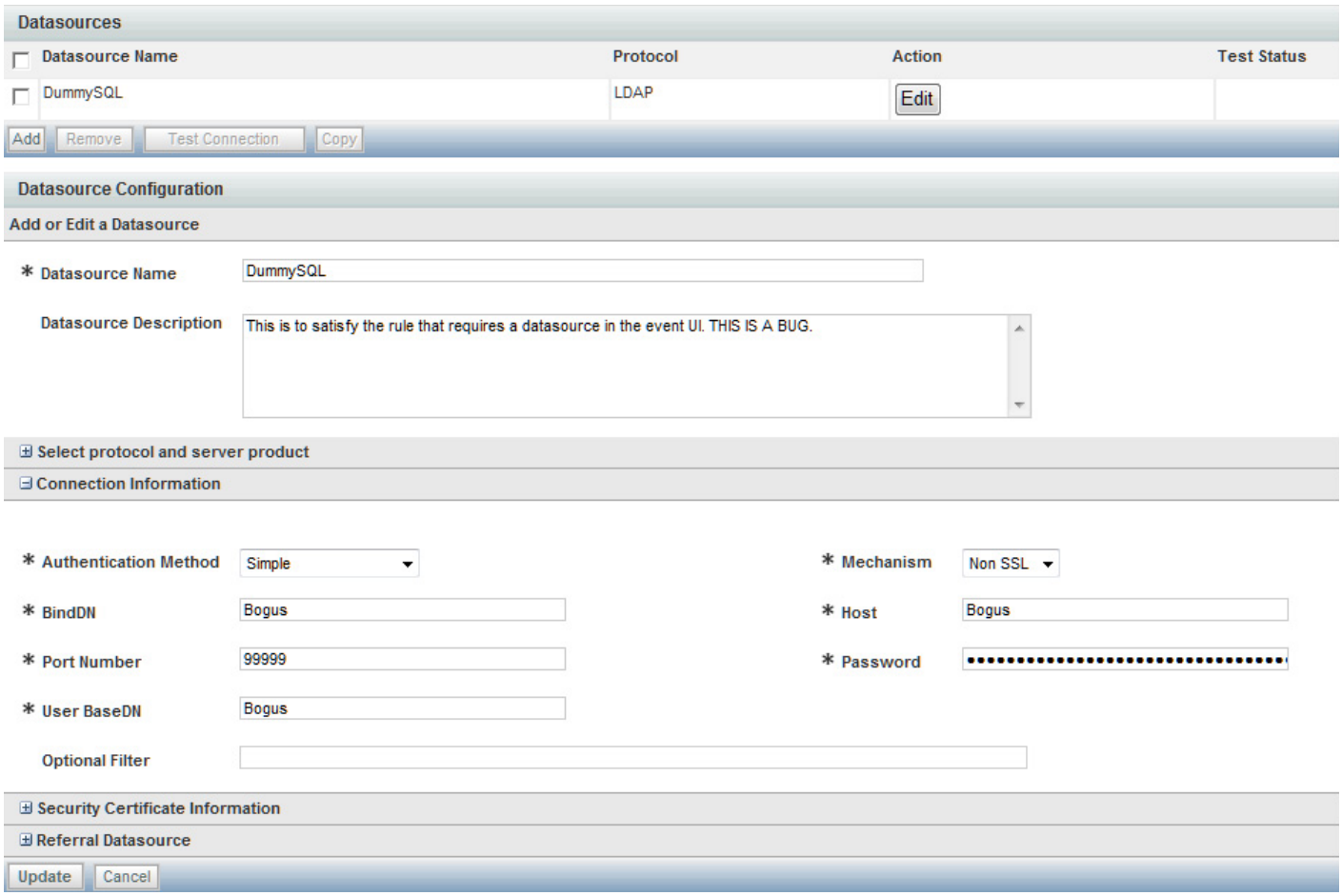
次に、上記のテーブル定義で使用するサンプル データの一部を示します。

```
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

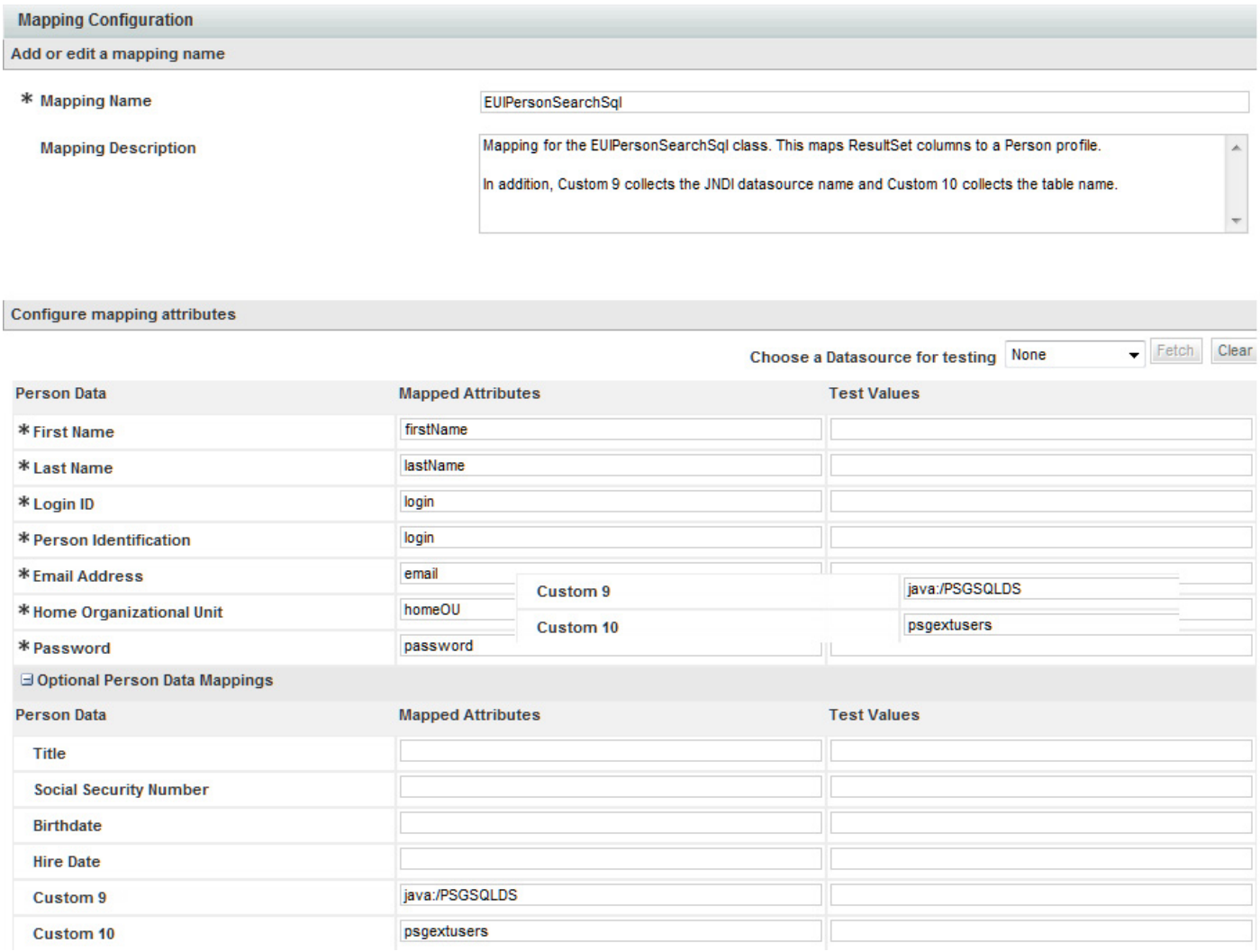


コンテナで管理されるデータソースの設定は、コンテナによって異なります。データソースの設定の詳細については、『*Cisco Prime Service Catalog Installation and Upgrade Guide*』を参照してください。

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

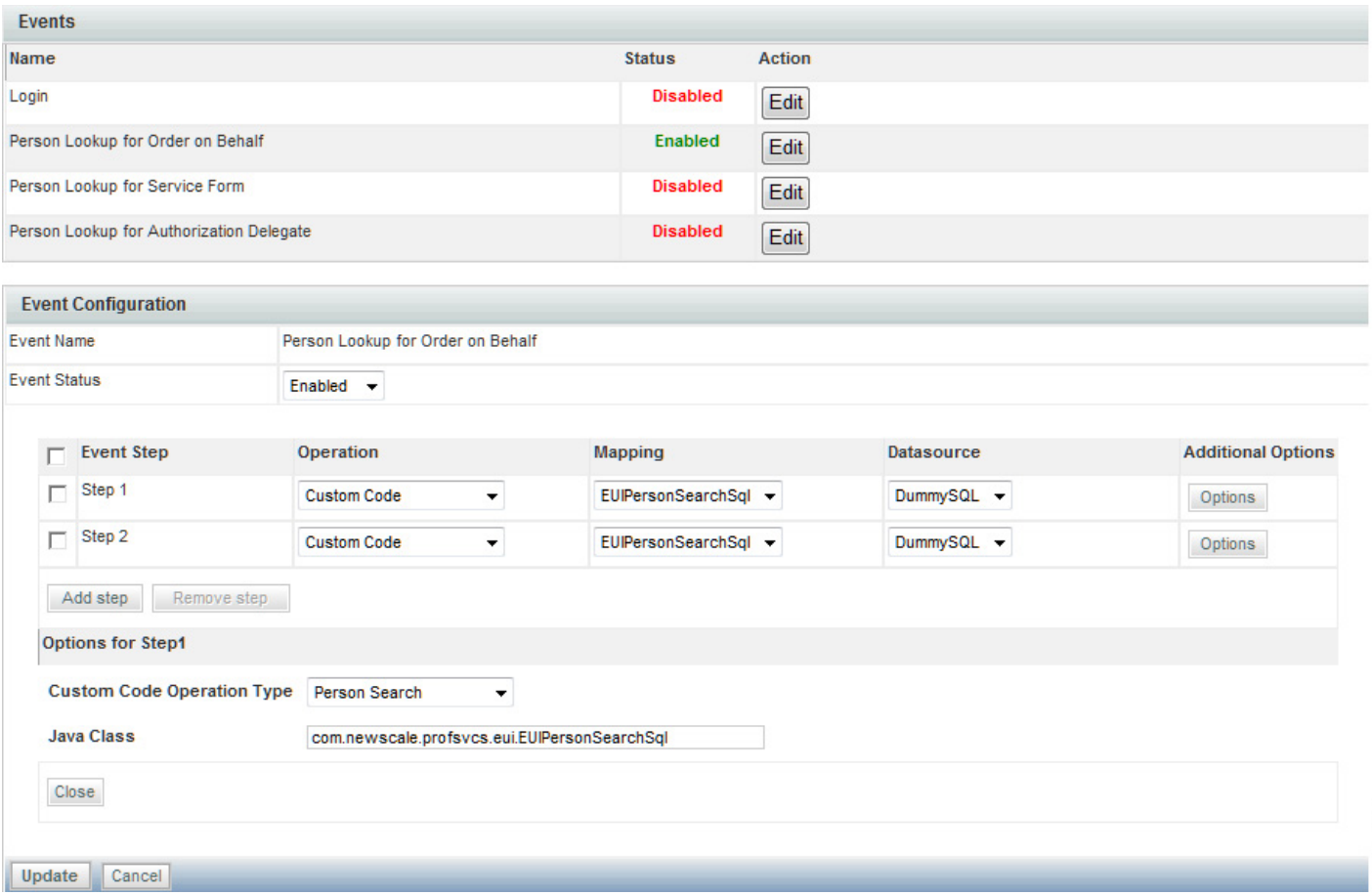


このマッピングには、Custom 9 として JNDI への参照が含まれ、Custom 10 にはテーブル名の参照が含まれています。このようなマッピングを使用すると、「select * from tablename」のような簡単なクエリーを実行し、JDBC のメタデータ機能を使用して、マッピングに基づいてカラムを選択することができます。

イベントの設定例

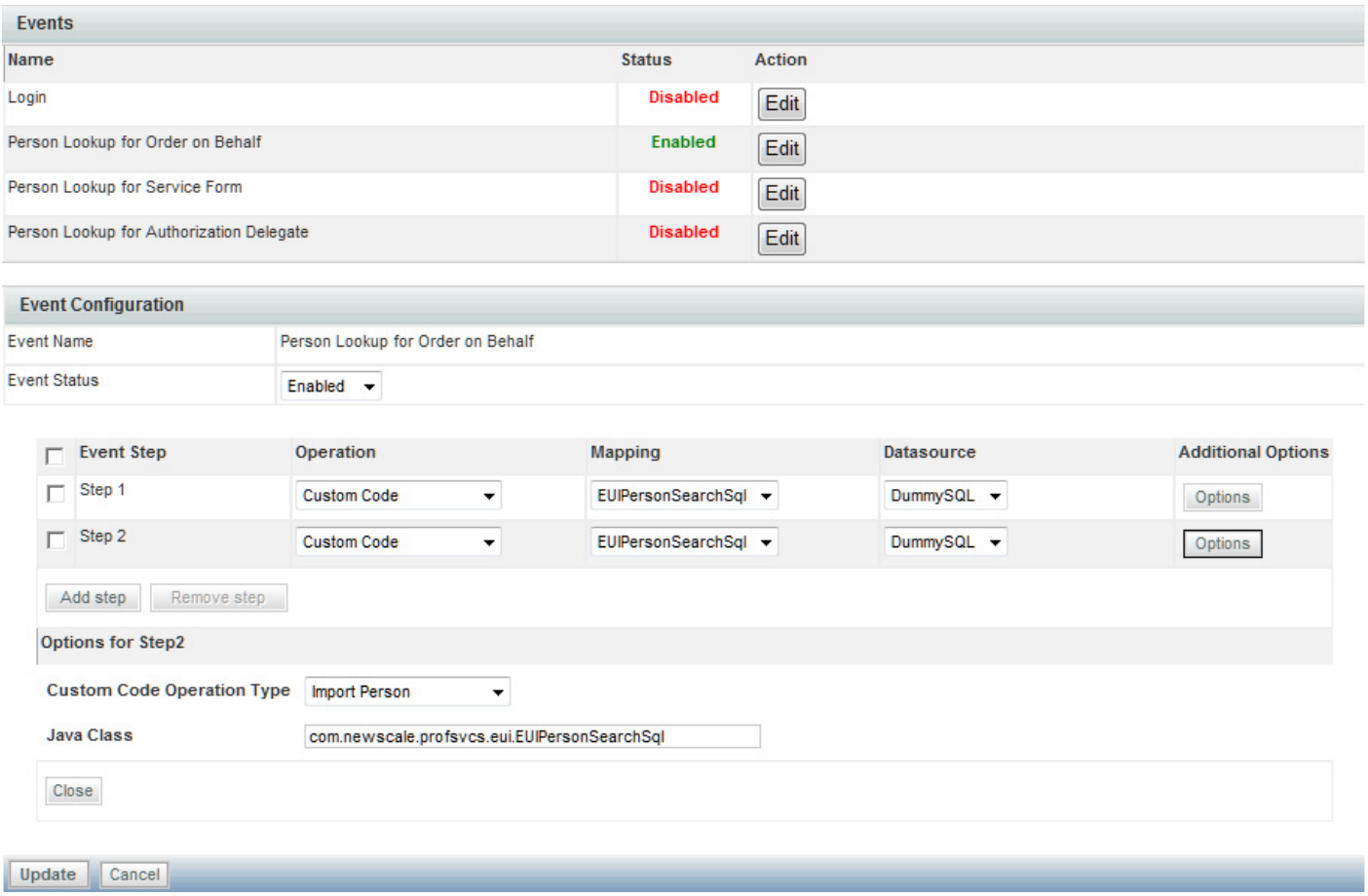
「Person Lookup for Order on Behalf」イベントには 2 つのステップがあり、最初のステップでは「Person Search」操作を実行する必要があります。クラスの名前はマッピングとして与えられます。パッケージの仕様全体は、Java クラスとして与えられます。

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>
```

CHAPTER 9

SAML での SSO の設定

Security Assertion Markup Language（SAML）は、当事者間で認証および承認の情報データを交換するための XML ベースのオープン標準のデータ形式です。

SAML には、次の 3 つの主要要素があります。

- **ユーザ**：サービス プロバイダ（Cisco Prime Service Catalog）へのログインを試行しているクライアント。
- **アイデンティティ プロバイダ（IDP）**：通常、ユーザがログインしているポータルであり、ユーザの ID に対する権限を保有します。ユーザのユーザ名、パスワード、および任意のグループ/属性の情報を保有しています。

（注） Prime Service Catalog 12.0 リリースのサポートでは、ユーザのログイン時に認証される IDP 接続は 1 つのみです。

- **サービス プロバイダ（SP）**：ユーザが使用を希望するアプリケーション。この場合、Cisco Prime Service Catalog です。

SAML は、Prime Service Catalog で実装されているため、Prime Service Catalog と統合するその他のアプリケーションは、認証の提供および IDP からのユーザ プロファイル情報のインポートを行う手段としてこれを使用できます。

注意 Prime Service Catalog では、LDAP と SSO ログイン用に設定された SAML の両方を設定することはできません。SAML SSO を使用する場合は、LDAP ログイン イベントを手動で無効にする必要があります。そうしないと、不正確なログイン動作につながります。
LADP ログインを無効にするには、[管理（Administration）] > [ディレクトリ（Directories）] > [イベント（Events）] に進み、ログイン イベントについて [編集（Edit）] をクリックします。イベントのステータスを [無効（Disabled）] に変更し、[更新（Update）] をクリックします。

ログイン動作

SAML シングル サインオンを実装するということは、サインイン プロセスとユーザ認証が完全に Prime Service Catalog の外部で処理されることを意味します。Prime Service Catalog は、SAML を IDP に対する認証を安全に行うための手段として使用します。承認は、Prime Service Catalog によって提供されます。システムに SAML が設定されている場合、ユーザはまず、IDP に対する認証を行う必要があります。正常に認証されると、ユーザは Prime Service Catalog にインポートされます。ユーザが存在せず、PSC にリダイレクトされた場合、そのユーザは、有効な権限があり、IDP が正しく設定されている場合のみ、アクセス権を付与されます。ユーザ セッションは、同一のブラウザ上で維持されます。

ログアウト動作

ログアウト動作は、*newscale.properties* ファイルで行う **saml.enable.globalLogout** プロパティの設定によって異なります。SAML 設定のためのプロパティ (9-3 ページ) を参照してください。

デフォルトでは、グローバル ログアウトは有効になっています。この場合、ユーザが Prime Service Catalog の 1 つのインスタンスからログアウトすると、そのユーザは、同じブラウザ上の他のインスタンスからもログアウトされます。

グローバル ログアウトが無効化されている場合、ユーザが Prime Service Catalog または Prime Service Catalog と統合されている他のアプリケーションからログアウトすると、SAML は、その特定のアプリケーションからのみユーザをログアウトします。これは、ローカル ログアウトと呼ばれます。

次のテーブルは、同じブラウザ上の 2 つの SP にグローバル ログアウトが設定されている場合のさまざまなログアウト動作を説明します。ここで、SP1 と SP2 は、2 つの Prime Service Catalog インスタンスです。

使用例	SP1 でのグローバル ログアウト設定	SP2 でのグローバル ログアウト設定	ログアウト動作
1	[はい(True)]	[はい(True)]	いずれかの SP がログアウトされると、SP1 と SP2 の両方がログアウトされる。
2	[はい(True)]	いいえ(False)	• SP1 がログアウトされると、SP2 もログアウトされる。 • SP2 がログアウトされても、SP1 はログアウトされない。
3	いいえ(False)	[はい(True)]	• SP1 がログアウトされても、SP2 はログアウトされない。 • SP2 がログアウトされると、SP1 もログアウトされる。
4	いいえ(False)	いいえ(False)	SP1 と SP2 のいずれかがログアウトされると、他方の SP もログアウトされる。

SAML でのユーザ管理

SAML を有効にすると、すべてのユーザ管理および認証は、Prime Service Catalog の外部で処理されます。ただし、Prime Service Catalog の外部で行われた変更は、Prime Service Catalog に即時に同期されます。ユーザ情報は IDP に対する認証を初めて試行したときにインポートされます。それ以降は、ユーザ情報がその後の試行によって更新されることはなく、ユーザの更新もありません。ユーザに対する変更は、Person Lookup OOB、Authorization Delegate、Person Lookup Service Form のために LDAP が有効化され、Import Person イベントが設定されると同期されます。システムからユーザを削除すると、そのユーザは Prime Service Catalog にサインインできなくなります（ただし、そのアカウントは Prime Service Catalog 内に存続します）。

SAML 設定のためのプロパティ

次の表では、お使いのシステムに SAML を設定するための *newscale.properties* での設定を説明します。

プロパティ	説明
saml.lb.protocol	LB の場合、「http」または「https」に設定します。
saml.lb.hostname	公開される RC エンドポイントに設定します。 ループ バック アドレス（127.0.0.1 または localhost）ではないことを確認してください。LB またはリバース プロキシを使用する場合、公開されているエンドポイントの IP アドレスまたはドメイン名になります。
saml.lb.port	適切なポート番号に設定します。
saml.lb.config.includeServerPortInRequestURL	true または false に設定します。 *true* に設定すると、ポートは、SP と IDP の間での SAML でのの交信の際に要求/応答を検証するために使用されます。
saml.matadata.refreshInterval	メタデータを更新する時間の間隔を設定します。
saml.provider.trustCheck	すべてのプロバイダについて、署名の信頼性の検証を設定します。
saml.force.auth	セッションが有効な場合でもユーザが認証を行う必要があるかどうかを設定します。
saml.enable.global.logout	グローバル ログアウトを有効にするか無効にするかを設定します。デフォルトでは、true に設定されます。
saml.certificate.validation.config	証明書検証の設定を指定します。詳細については、SAML 証明書の検証の設定（9-4 ページ）を参照してください。

SAML 証明書の検証の設定

この項では、SAML 証明書の検証を設定する際に Prime Service Catalog で可能な SAML 証明書の検証の設定について説明します。

SAML 仕様では、メッセージを受信する際にはメッセージがデジタル署名されている必要があります。SAML では、署名は常に必要です。SAML 証明書を検証するには、次のプロパティを設定します。

プロパティ	説明
checkFQDNValidity	true に設定すると、証明書内の完全修飾ドメイン名または共通名がチェックされます。
allowSelfSignedCertificates	true に設定すると、自己署名証明書が許可されます。
allowOnlyRootCertificates	true に設定すると、ルート証明書のみが許可されます。デフォルトは false です。 (注)　allowOnlyRootCertificates を true に設定すると、allowSelfSignedCerticates を false に設定していても、すべての自己署名証明書が許可されます。すべてのルート証明書は自己署名されているためです。
checkValidity	true に設定すると、証明書の有効期間がチェックされます。
checkMaxExpiryDays	true に設定すると、証明書の最大有効期間がチェックされます。
checkCertificateRevocation	true に設定すると、証明書内の動的証明書失効リストがチェックされます。
checkTrust	true に設定すると、信頼チェーンからの証明書が検証されます。

SAML 設定と IDP マッピングの設定

SAML の設定と、IDP の Prime Service Catalog へのマッピングの設定の詳細については、『Cisco Prime Service Catalog Administration and Operation Guide』の「*SAML Configuration* 」の項を参照してください。

SAML REST API

SAML nsAPI にアクセスできるのは、サイト管理者および SAML 設定が可能なユーザのみです。SAML の設定および IDP マッピング に対する nsAPI 認証には、SAML が有効な場合でも、RC DB を使用します。したがって、ユーザは、RC DB のクレデンシャルを使用する必要があります。

正常に送信されたオーダーの応答メッセージは 200 です。

エラー応答メッセージについては、「REST/Web サービスのエラー メッセージ」の表および「エラー メッセージ」を*参照*してください。

表 9-1　SAML REST API テーブル

領域	例
DELETE	**IDP 設定の削除** DELETE URL: http://<ServerURL>/RequestCenter/nsapi/v1/idp/configs/<idp configuration name> IDP 設定を削除するには、IDP の固有名を入力します。
GET	**IDP 設定の取得** GET URL: http://<ServerURL>/RequestCenter/nsapi/v1/idp/configs/<idp configuration name> IDP 設定を取得するには、IDP の固有名を入力します。
PUT	**ノードのメタデータの更新** PUT URL http://<ServerURL>/RequestCenter/nsapi/v1/idp/refreshThis

表 9-1　SAML REST API テーブル（続き）

領域	例

POST	**IDP 設定の保存**
	POST URL:
	http://<ServerURL>/RequestCenter/nsapi/v1/idp/configs
	入力例:
	{
	"name": "ssocirclef631a5967b044cec94893ac700851de3",
	"metadata": "<?xml version=\"1.0\"
	encoding=\"UTF-8\"?>\r\n<md:EntityDescriptor
	\r\n\txmlns:md=\"urn:oasis:names:tc:SAML:2.0:metadata\"
	\r\n\tentityID=\"https://auth.miniorange.com/moas\">\r\n\r\n\t<md:IDPSSODescr
	iptor \r\n\t\tWantAuthnRequestsSigned=\"true\"
	\r\n\t\tprotocolSupportEnumeration=\"urn:oasis:names:tc:SAML:2.0:protocol\">\
	r\n\r\n\t\t<md:KeyDescriptor \r\n\t\t\tuse=\"signing\">\r\n\t\t\t<ds:KeyInfo
	\r\n\t\t\t\txmlns:ds=\"http://www.w3.org/2000/09/xmldsig#\">\r\n\t\t\t\t<ds:X
	509Data>\r\n\t\t\t\t\t<ds:X509Certificate>MIICnjCCAgegAwIBAgIJAK3CyOFtrUj
	MA0GCSqGSIb3DQEBBQUAMGgxCzAJBgNVBAYTAklOMQswCQYDVQQIDAJNSDENMAsGA1UEBwwEUFVOR
	TETMBEGA1UECgwKbWluaU9yYW5nZTETMBEGA1UECwwKbWluaU9yYW5nZTETMBEGA1UEAwwKbWluaU
	9yYW5nZTAeFw0xNTAyMTEwNDQ1NDdaFw0xODAyMTAwNDQ1NDdaMGgxCzAJBgNVBAYTAklOMQswCQY
	DVQQIDAJNSDENMAsGA1UEBwwEUFVORTETMBEGA1UECgwKbWluaU9yYW5nZTETMBEGA1UECwwKbWlu
	aU9yYW5nZTETMBEGA1UEAwwKbWluaU9yYW5nZTCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAx
	WMW0HNXVL4VB14PklXDo6rJlK3W4XHsxD7rBsG8e2LgbfjEjC0b
	k2/5ODuP9OvVQyHaZhMPWbS2z5S6cxCIxPfAJC5pCn9EVVoSDbz4C1Biyg9NJAUYp7oF
	8JfKByLeWCOPRb9/G8/Bq5xQRAf CH/hSSsrNEQm5h
	NnhcCAwEAAaNQME4wHQYDVR0OBBYEFFq3KKnNFb1777slDNKfn30gXcvjMB8GA1UdIwQYMBaAFFq3
	KKnNFb1777slDNKfn30gXcvjMAwGA1UdEwQFMAMBAf8wDQYJKoZIhvcNAQEFBQADgYEACP2t4JNkG
	h2ElJltQ3FSdWWHsvhGpGnpAdltdC8vW/Sf3a97IDeixr5GcQVfUfyYE nMQU0g2NJLYG1
	hb13J58eQ9NhU8PgkSsJWaskST1KTNRu
	30K3Dm8TOhZShWEvYBUzSDjcSJFUguXeoK/gx4wBuA8WEaKb9PC6xvac/4=</ds:X509Certifica
	te>\r\n\t\t\t\t</ds:X509Data>\r\n\t\t\t</ds:KeyInfo>\r\n\t\t</md:KeyDescripto
	r>\r\n\r\n\t\t<md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:emai
	lAddress</md:NameIDFormat>\r\n\t\t<md:NameIDFormat>urn:oasis:names:tc:SAML:1.
	1:nameid-format:unspecified</md:NameIDFormat>\r\n\t\t<md:SingleSignOnService
	\r\n\t\t\tBinding=\"urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST\"
	\r\n\t\t\tLocation=\"https://auth.miniorange.com/moas/idp/samlsso\"/>\r\n\t\t
	<md:SingleSignOnService
	\r\n\t\t\tBinding=\"urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect\"
	\r\n\t\t\tLocation=\"https://auth.miniorange.com/moas/idp/samlsso\"/>\r\n\t</
	md:IDPSSODescriptor>\r\n</md:EntityDescriptor>",
	"attributesMapping": {
	"firstName": "FName",
	"lastName": "LName",
	"businessUnit": "bu",
	"localeCode": "locale",
	"costCenter": "costCenter",
	"organizationUnit": "Department",
	"login": "Email",
	"email": "Email"
	}
	}
	(注)　組織、ロケール、部門、およびコスト センターは任意です。これらのフィールドをマッピングしたくない場合は、空白の値を送信します。

表 9-1　SAML REST API テーブル(続き)

領域	例
PUT	**IDP 設定の更新** PUT URL: http://<ServerURL>/RequestCenter/nsapi/v1/idp/configs 入力例:

```
{
  "name": "oneloginp1",
  "metadata": "<?xml version=\"1.0\"?>\n<EntityDescriptor
xmlns=\"urn:oasis:names:tc:SAML:2.0:metadata\"
entityID=\"https://app.onelogin.com/saml/metadata/581650\">\n
<IDPSSODescriptor xmlns:ds=\"http://www.w3.org/2000/09/xmldsig#\"
protocolSupportEnumeration=\"urn:oasis:names:tc:SAML:2.0:protocol\">\n
<KeyDescriptor use=\"signing\">\n      <ds:KeyInfo
xmlns:ds=\"http://www.w3.org/2000/09/xmldsig#\">\n        <ds:X509Data>\n
<ds:X509Certificate>MIIEDjCCAvagAwIBAgIUBtd7yzkKX1N8+cmROGrTbzCn8OowDQYJKoZIh
vcNAQEF\nBQAwVTELMAkGA1UEBhMCVVMxDjAMBgNVBAoMBVBTQ1FBMRUwEwYDVQQLDAxPbmVM\nb2
dpbiBJZFAxHzAdBgNVBAMMFk9uZUxvZ2luIEFjY291bnQgOTA5MzYwHhcNMTYw\nODMxMDYzMjI4W
hcNMjEwOTAxMDYzMjI4WjBVMQswCQYDVQQGEwJVUzEOMAwGA1UE\nCgwFUFNDUUExFTATBgNVBAsM
DE9uZUxvZ2luIElkUDEfMB0GA1UEAwwWT25lTG9n\naW4gQWNjb3VudCA5MDkzNjCCASIwDQYJKoZ
IhvcNAQEBBQADggEPADCCAQoCggEB\nALjcSyIa/RW7w3fh+KpdhmXsw2WSuYFJkfmZEqwHTSHGd0
n1Kv6RvYtarWEvGsVN\njVTSgfMDZ14uW2qvTpCVjf5vWNvnGnOQFFjGWgMgsnrbKFGh62kvkNxKp
pqpdC1v\nwOZucoLvlaCJR/Od3SQNfQLwDAcpmbMiHb1bZm03bKMAPO+cw6mkKl8Ov3zuKt4I\nEd
vwCIzZraRW9RUPPKXX7Y5sli3ywaxEy/69mxwaeuhMtFCk2BwYT8AJ+LMoeLXx\nIURPSobdTpqBQ
PEOmFcJ/8SaMHsr+1EP1HGxKM4bXocE0soFYH5MxPCTmedxnQ7L\nhBiSdVEGTJMGAzpejf2f85cC
AwEAAaOB1TCB0jAMBgNVHRMBAf8EAjAAMB0GA1Ud\nDgQWBBRJ1vyA3mh0H7eH+OFrrq7oyTMAYTC
BkgYDVR0jBIGKMIGHgBRJ1vyA3mh0\nH7eH+OFrrq7oyTMAYaFZpFcwVTELMAkGA1UEBhMCVVMxDj
AMBgNVBAoMBVBTQ1FB\nMRUwEwYDVQQLDAxPbmVMb2dpbiBJZFAxHzAdBgNVBAMMFk9uZUxvZ2luI
EFjY291\nbnQgOTA5MzaCFAbXe8s5Cl9TfPnJkThq028wp/DqMA4GA1UdDwEB/wQEAwIHgDAN\nBg
kqhkiG9w0BAQUFAAOCAQEAQxh4/+8Vt2oSpWmMYPf8CpbH3sQuhphhEJzkEP7y\nkZILM1tV8szt9
YFlfjUIH/usGOx/aIBEDpPj0T/UTGl4QhZyv5V+T3DhcZeOK7g3\nGJTy0w6HfWuGBY8FZTMOdGOd
RSLQUOoKehIV0iIlrZAlEyMrPIx7qGrYf1zxFqoa\nyPzNT6/AUXqujcQjZwyRBwqT6429xX74ksV
e0C8KmfEUvgPfNj+wtf+KhsCqckLX\nH/HQo6Ua4nU6vuBQLym9E0OEKAOHIYccJFqlBoREtRw/V/
J7Gk5Z0yEq7XAM9EC/\n/A3vQG0DL6eIG9Tff5Tf+G4gyXQfGCDOhjkAxTbkoW+jJg==</ds:X509
Certificate>\n        </ds:X509Data>\n      </ds:KeyInfo>\n
</KeyDescriptor>\n    <SingleLogoutService
Binding=\"urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect\"
Location=\"https://pscqa.onelogin.com/trust/saml2/http-redirect/slo/581770\"/
>\n    \n
<NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress</NameIDF
ormat>\n    \n    <SingleSignOnService
Binding=\"urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect\"
Location=\"https://pscqa.onelogin.com/trust/saml2/http-redirect/sso/581770\"/
>\n    <SingleSignOnService
Binding=\"urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST\"
Location=\"https://pscqa.onelogin.com/trust/saml2/http-post/sso/581770\"/>\n
<SingleSignOnService Binding=\"urn:oasis:names:tc:SAML:2.0:bindings:SOAP\"
Location=\"https://pscqa.onelogin.com/trust/saml2/soap/sso/581770\"/>\n
</IDPSSODescriptor>\n  <ContactPerson contactType=\"technical\">\n
<SurName>Support</SurName>\n
<EmailAddress>support@onelogin.com</EmailAddress>\n
</ContactPerson>\n</EntityDescriptor>\n",
"attributesMapping": {
    "firstName": "FName",
    "lastName": "LName",
    "organizationUnit": "Department",
   "login": "Email",
    "email": "Email"
  }
}
```

表 9-1　SAML REST API テーブル（続き）

領域	例
GET	**SAML 設定の取得** GET URL: http://<ServerURL>/RequestCenter/nsapi/v1/saml/configs

PUT	**SAML 設定の更新**
	PUT URL:
	http://<ServerURL>/RequestCenter/nsapi/v1/saml/configs
	入力例:
	{
	"entityID": "75781d57-a5cd-4db2-a1d5-58407a8c7887",
	"b64Certificate":
	"MIIDsjCCApqgAwIBAgIEIXc9vjANBgkqhkiG9w0BAQsFADB5MUMwQQYDVQQDDDo3YjQwNDMwYS04
	\nODAxLTQ2NDctOTNjNy03YzNjMjVkZTBkYTQtc2Vydmljc2WNhdGFsb2dkZWZhdWx0MQ0wCwYDVQQ
	L\nDAROb25lMRQwEgYDVQQKDAtOb25lIEw9Tm9uZTENMAsGA1UEBhMETm9uZTAeFw0xNjExMDIxMz
	Uw\nNTBaFw0xNzAxMzExMzUwNTBaMHkxQzBBBgNVBAMMOjdiNDA0MzBhLTg4MDEtNDY0Ny05M2M3L
	Tdj\nM2MyNWRlMGRhNC1zZXJ2aWNlY2F0YWxvZ2RlZmF1bHQxDTALBgNVBAsMBE5vbmUxFDASBgNV
	BAoM\nC05vbmUgTD1Ob25lMQ0wCwYDVQQGEwROb25lMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMII
	BCgKC\nAQEAryLcEinIjhnUu9wP8H/AWn/rYA2IkcuacD6VNEzHaNCBR+k//2MNv5jsVGAxpxUkjM
	i8uIjM\nJvTvW7wVEzMGVTai6XDG48jpZSTIkftnpeZuO3iydJoSI5BOiYxn4d6VqZnEDPas1Qxrf
	iKsMqbC\nbfuWCtdOYE2Rqh8s0U6+BA2D/pXxbykfMYGa3hNbTgsvZjkfUropWTxrkNbP6mWOMbcC
	03e9ih9i\n95y3Et1APQ9uLDxcGf3Rr7h/md7k1S7pEunuJw7YSgmSDsg2gFnEnubT9SeWUvj5oT3
	/fHFElOvQ\nf8QlGKAJdRG1sP07mBSztDMlSYbtHWJfi+bYitD81wIDAQABo0IwQDAfBgNVHSMEGD
	AWgBQPOMLi\nmFPO0Ooj9Vs7UKmMdmhg3zAdBgNVHQ4EFgQUDzjC4phTztDqI/VbO1CpjHZoYN8wD
	QYJKoZIhvcN\nAQELBQADggEBAAwyRikaRzL/7ZahIonrsIxRr8QW+JRCAXJS52PRag/dGlpsxCp6
	/xD3QxJ+/EY2\n7gvO0lyBth23oKJVt3zgIH5tC+VHTdmT4Eeluv4iw4ZU0qYD/NCCEBilII68xOr
	ASbE5fiBWpn3Q\nm7le5IXK7KIFUa5VmfOuGgXap9s0AF1TelGPjjlNXmMxWJgxlu8ms7/Uoaju2H
	dFyznAyK0bdzSX\nguR2VsQiwbWTuBDKySc9hoZd4qVFTJmVTVrjbpmrAEY/xk+OCVb0T1JJBtlZQ
	EsYe6KR2xdnE6ny\nqycNHpclxVJ8yIXxeoLnJK2pmCbIcBt8v2fQPhPneBbaZOlerBg=",
	"b64PrivateKey":
	"MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQCvItwSKciOGdS73A/wf8Baf+tg
	\nDYiRy5pwPpU0TMdo0IFH6T//Yw2/mOxUYDGnFSSMyLy4iMwm9O9bvBUTMwZVNqLpcMbjyOllJMi
	R\n+2el5m47eLJ0mhIjkE6JjGfh3pWpmcQM9qzVDGt+IqwypsJt+5YK105gTZGqHyzRTr4EDYP+lf
	Fv\nKR8xgZreE1tOCy9mOR9SuilZPGuQ1s/qZY4xtwLTd72KH2L3nLcS3UA9D24sPFwZ/dGvuH+Z3
	uTV\nLukS6e4nDthKCZIOyDaAWcSe5tP1J5ZS+PmhPf98cUSU69B/xCUYoAl1EbWw/TuYFLO0MyVJ
	hu0d\nYl+L5tiK0PzXAgMBAAECggEALqim4N/o4pLXLkVuqbAfWv0BhFGWtOD9gDHsJkbeSXpjNvl
	ZZ3zI\nSOdA7ynBkLX9StSgErm/ShGvQ01UgAzz/vfTZ0X4du8r3xpxpRLJhlVhwM5jHNV/R6JGij
	ax5mca\nkFi69okxeoEYkj5CiiwLWKnSS4kZBGcmC6DKm+jSjtlop+ErzcLmiBqBPlQHL/rZpp0T6
	2ojOMB/\nD8Au0IFecNIyitnTORBaOVRt1ohQXBhsrjSHQcXmP7TsDrm6H5XmE3sDfDT6UrYyvLNM
	uCNBfmrj\noE/kNnFUiQZthJWkFWoHSM1eehuUR6nsubg0q0KGrsI9ta+rof0FY51Ogr5jYQKBgQD
	44/5LT1u1\n6NLfM24dd2f6gD8cSV4VVfRLRktLogjqa8n3kTZOb/ELgLQDPotcHOQXDwDmK2OYpc
	fRG2RgGt22\nMXdLHawjWItmr2wkzhanojapdssiCU9NDb209eHOUpT82pz0Vouw9LlzV26J1++Ki
	BoyGMO5Xh+L\nKjm5aNZQHQKBgQC0I4nuCvFMvJ14gIRvVmcCcHbHREVmuSeFOKsXL8kYkYsrUvcJ
	mSkw6GnMtiSh\nfsHwFtJmakZa+QDBNUJhKuvyhfC+9vaUsPjXK20Oa5dd8eQoN9Bz9dTptjx001f
	phFiidNE4+f/1\nsKN/0YnKoBOJsEb7Zv3yzJCMPBoPHvmWgwKBgQCvTl+iCf6N7bUB88a+yIkbf1
	N0iBTVsFpG3vdQ\nCYyAGXYDg2ud6ej9ciTZGCeutMbPmwjGFo+rSDGrDsEvlBzQJJ1i8j56Evb1V
	+AzOFnqry4TRRIl\nIiuSGXiiyoHHApHgW9crnv37oRQyssWwH8GgcOcKnDjYCvzq184a00YI3QKB
	gDWqMLkdwW0e87qm\nbs3Ma7uqTXhnulUz67Ygf7fUoJAVK+SoPrg5TLApTPuTd6402QnxgpTILFW
	FwNfOSgwwgUIq7OG3\nKRZ68mcHpOGa4+k02seweQVSwy8s/y2+mH4U02LycjILKFnFWbAGeIpIzg
	lC3qKeuCDRGG7uqMTA\ncZKJAoGAcrP9/zpxLyyBm8WjmAmC0UVgpCZmBDEEQKZxqNmqP/oIYbXCK
	ClS5sQc7ybeXigyq37B\ncAuyHa+rVVl/FClnWlsgG9DmZOTjqyL7ttJSP9hJjHzlJp5dw6uVVexz
	WheZWFKbGC0obLod5522\nM+n5j+epGNK6tTRWfVERYnXthcc="
	}

WildFly 8.2 アプリケーション サーバでの SSL の有効化

WildFly 11.1、11.1.1、および 12.0 での SSL の有効化

前提条件

- 次の Java 開発キットのいずれかがあることを確認します。
  - Oracle JDK 1.8.0_77 以上
  - サーバ内に Open JDK 1.8.0_77 以上があること。
- クライアント VM に JAVA_HOME 変数を設定する必要があります。
- Linux マシンおよび Windows マシン では、すべてのクラスタ管理スクリプト内の --connect 変数に次のプレフィックスを追加します。
  - --user=adminuser --password=newscale
- cisco.com で利用可能なファイルをダウンロードします。これらのダウンロードされたファイルの名前を次の表に記載するとおり既存のファイル名に変更し、既存のファイルを新しいファイルに置き換えます。次の表は、置き換えが必要なファイルのすべての詳細を示します。

(注) ユーザ名とパスワードが変更される場合は、必要な変更を行う必要があります。

インストール タイプ	データベース タイプ	既存ファイル	新しいファイル	コメント	新しいファイルのダウンロード リンク
標準的なスタンドアロン/カスタムのスタンドアロン	SQL	C:\InstallDirectory\wildfly-8.2.0.Final\ServiceCatalogServer\configuration**standalone-full.xml**	Standalone-full_RC_SQL.xml	新しいファイルを既存のパスに移動し、その名前を**太字**で記載する名前に置き換えます。	cisco.com
標準的なスタンドアロン/カスタムのスタンドアロン	SQL	C:\InstallDirectory\wildfly-8.2.0.Final\ServiceLinkServer\configuration**standalone-full.xml**	Standalone-full_SL_SQL.xml		
標準的なスタンドアロン/カスタムのスタンドアロン	Oracle	C:\InstallDirectory\wildfly-8.2.0.Final\ServiceCatalogServer\configuration**standalone-full.xml**	Standalone-full_RC_Oracle.xml		
標準的なスタンドアロン/カスタムのスタンドアロン	Oracle	C:\InstallDirectory\wildfly-8.2.0.Final\ServiceLinkServer\configuration**standalone-full.xml**	Standalone-full_SL_Oracle.xml		
2VM クラスタ	SQL	C:\InstallDirectory\wildfly-8.2.0.Final\domain\configuration**domain.xml**	2VM_SQL_domain.xml		
2VM クラスタ	Oracle	C:\InstallDirectory\wildfly-8.2.0.Final\domain\configuration**domain.xml**	2VM_Oracle_domain.xml		
4VM クラスタ	SQL	C:\InstallDirectory\wildfly-8.2.0.Final\domain\configuration**domain.xml**	4VM_SQL_domain.xml		

インストール タイプ	データベース タイプ	既存ファイル	新しいファイル	コメント	新しいファイルのダウンロード リンク
4VM クラスタ	Oracle	C:\InstallDirectory\wildfly-8.2.0.Final\domain\configuration**domain.xml**	4VM_Oracle_domain.xml		
標準的なスタンドアロン/カスタムのスタンドアロン/2VM クラスタ/4VM クラスタ	SQL/Oracle	同上	https-users.properties		
標準的なスタンドアロン/カスタムのスタンドアロン/2VM クラスタ/4VM クラスタ	SQL/Oracle	同上	https-roles.properties		
4VM クラスタ	SQL/Oracle	C:\InstallDirectory\wildfly-8.2.0.Final\domain\configuration**host1.xml**	4VM_host1_backup.xml		
2VM クラスタ	SQL/Oracle	C:\InstallDirectory\wildfly-8.2.0.Final\domain\configuration**hostva_backup.xml**	2VM_hostva_backup.xml		
4VM クラスタ	SQL/Oracle	C:\InstallDirectory\wildfly-8.2.0.Final\domain\configuration**host_default.xml**	4VM_host_default.xml		
2VM クラスタ	SQL/Oracle	C:\InstallDirectory\wildfly-8.2.0.Final\domain\configuration**host2.xml**	2VM_host2_backup.xml		
4VM クラスタ	SQL/Oracle	C:\InstallDirectory\wildfly-8.2.0.Final\domain\configuration**host2.xml**	4VM_host2_backup.xml		

標準的なスタンドアロン モードでの SSL の有効化

手順 1　C:\SSL パスで作成を行い、ディレクトリに移動して次の手順に従います。
これにより、サーバ キー（秘密および公開）とクライアント キーのペアが作成されます。

a. 次のコマンドを入力し、サーバの秘密キー（serverkey）とクライアントの秘密キー（clientkey）をそれぞれ作成します。

```
keytool -genkeypair -alias serverkey -keyalg RSA -keysize 2048 -validity 7360
-keystore server.keystore
keytool -genkeypair -alias clientkey -keyalg RSA -keysize 2048 -validity 7360
-keystore client.keystore
```

（注） 秘密キーと公開キーを作成するために入力する情報が、クライアントとサーバの両方で一致していることを確認します。

（注） キーの作成に使用されるデフォルトのパスワードは *secret* です。新しいパスワードを作成する場合は、*rcjms.properties*、*integration-server.properties*、および *standalone-full.xml* ファイルにある *secret* という語を選択したパスワードに置き換えます。

（注） 名前と姓を要求されたら、ホスト コンピュータの IP アドレスを入力します。

b. 次のコマンドを入力し、キーを証明書にエクスポートします。

```
keytool -export -alias serverkey -keystore server.keystore -rfc -file server.crt
keytool -export -alias clientkey -keystore client.keystore -rfc -file client.crt
```

c. 次のコマンドを入力し、公開キーを証明書からエクスポートし、信頼ストアにインポートします。

```
keytool -import -file server.crt -keystore client.truststore
keytool -import -file client.crt -keystore server.truststore
```

d. **キーストア** ファイルおよび**信頼ストア** ファイルを以下の場所に配置します。

```
C:\Install_Dir\wildfly-8.2.0.Final\ServiceCatalogServer\configuration
C:\Install_Dir\wildfly-8.2.0.Final\ServiceLinkServer\configuration
C:\Install_Dir\bin
```

手順 2　**RequestCenter.war** の **rcjms.properties** および **ServiceLink.war** の **integration-server.properties** を編集します。

a. **Request Center** ファイルで、**rcjms.properties** セクションの次の変数を編集します。

- **http-remoting** という値をすべて **https-remoting** に置き換えます。
- ポート **6080** という値をすべてポート **6443** に置き換えます。

（注） 次の情報は、11.1 で SSL を有効にする場合は無視してください。

- 次のエントリを追加します。

```
## FOR SSL ##

BEEERequisitions.CLIENT_KEYSTORE=client.keystore
```

```
BEEERequisitions.CLIENT_TRUSTSTORE=client.truststore
BEEERequisitions.KEYSTORE_PASSWORD=secret
BEEERequisitions.TRUSTSTORE_PASSWORD=secret
BEEERequisitions.TRUSTSTORE_TYPE=JCEKS

## FOR SSL ###
```

b. **Service Link** ファイルで、**integration-server.properties** セクションの次の変数を編集します。
   - **http-remoting** という値をすべて **https-remoting** に置き換えます。
   - ポート **6080** という値をすべてポート **6443** に置き換えます。

（注） 次の情報は、11.1 で SSL を有効にする場合は無視してください。

   - 次のエントリを追加します。

```
## FOR SSL ##

ISEEOutbound.CLIENT_KEYSTORE=client.keystore
ISEEOutbound.CLIENT_TRUSTSTORE=client.truststore
ISEEOutbound.KEYSTORE_PASSWORD=secret
ISEEOutbound.TRUSTSTORE_PASSWORD=secret
ISEEOutbound.TRUSTSTORE_TYPE=JCEKS

## FOR SSL ###
```

手順 3　添付されている **https-users.properties** および **https-roles.properties** を次の場所にコピーします。

```
C:\Install_Dir\wildfly-8.2.0.Final\standalone\configuration
C:\Install_Dir\wildfly-8.2.0.Final\ServiceCatalogServer\configuration
C:\Install_Dir\wildfly-8.2.0.Final\ServiceLinkServer\configuration
```

手順 4　**standalone-full-RC.xml** を **Standalone-full_RC_SQL.xml** または **Standalone-full_RC_Oracle.xml** に、**standalone-full-SL.xml** を **Standalone-full_SL_SQL.xml** または **Standalone-full_SL_Oracle.xml** に置き換え、それらを次の場所に配置します。これらの xml ファイルはここからダウンロードします。

```
C:\Install_Dir\wildfly-8.2.0.Final\ServiceCatalogServer\configuration
C:\Install_Dir\wildfly-8.2.0.Final\ServiceLinkServer\configuration
```

（注） インストールが Oracle である場合、データ ソースをそれに応じて変更する必要があります。

上記のファイルの名前を **standalone-full.xml** に変更し、**Service Link Oracle** に対し、次の操作を実行します。

a. **IP_ADDRESS** および **DB_NAME** を顧客が使用する IP アドレスおよびデータベースの名前に置き換えます。

```
<connection-url>jdbc:sqlserver://IP_ADDRESS:1433;DatabaseName=DB_NAME</connection-url>
```

b. **IP_ADDRESS** をアプリケーションがインストールされているマシンの IP アドレスに置き換えます。

```
<outbound-socket-binding name="my-http">
          <remote-destination host="IP_ADDRESS"port="${jboss.https.port:6443}"/>
</outbound-socket-binding>
```

手順 5　次のパスにあるモジュール ファイルに依存関係を追加します。
C:\Install_Dir\wildfly-8.2.0.Final\modules\system\layers\base\io\netty\main\module.xml

```
<dependencies>
     <module name="javax.api"/>
</dependencies>
```

手順 6　**newscale.properties** を編集し、**isee.base.url** 変数が **https://<ip-address>:6443** に設定されていることを確認します。

（注）　この手順 7 の使用が必要なのは、11.1 の場合のみです。

手順 7　次を **startServiceCatalog.conf.cmd/startServiceCatalog.conf.sh** にコピーします。

```
## For Windows START ##

set "JAVA_OPTS=%JAVA_OPTS% -Djavax.net.ssl.keyStore=client.keystore"
set "JAVA_OPTS=%JAVA_OPTS% -Djavax.net.ssl.trustStore=client.truststore"
set "JAVA_OPTS=%JAVA_OPTS% -Djavax.net.ssl.keyStorePassword=secret"
set "JAVA_OPTS=%JAVA_OPTS% -Djavax.net.ssl.trustStorePassword=secret"
set "JAVA_OPTS=%JAVA_OPTS% -Djavax.net.ssl.trustStoreType=JCEKS"

## For Windows END ##

## For Linux START ##

JAVA_OPTS="$JAVA_OPTS -Djavax.net.ssl.keyStore=client.keystore"
JAVA_OPTS="$JAVA_OPTS -Djavax.net.ssl.trustStore=client.truststore"
JAVA_OPTS="$JAVA_OPTS -Djavax.net.ssl.keyStorePassword=secret"
JAVA_OPTS="$JAVA_OPTS -Djavax.net.ssl.trustStorePassword=secret"
JAVA_OPTS="$JAVA_OPTS -Djavax.net.ssl.trustStoreType=JCEKS"

## For Linux END ##
```

手順 8　サーバのログを消去して、Service Link、Request Center の順に再起動します。

カスタムのスタンドアロン モードでの SSL の有効化

手順 1　次のコマンドを入力し、カスタムのスタンドアロン用の秘密キーを作成します。

a. **RC** で次のコマンドを入力し、サーバの秘密キー（server1key）とクライアントの秘密キー（client1key）をそれぞれ作成します。

```
keytool -genkeypair -alias server1key -keyalg RSA -keysize 2048 -validity 7360
-keystore server1.keystore
keytool -genkeypair -alias client1key -keyalg RSA -keysize 2048 -validity 7360
-keystore client1.keystore
```

b. **SL** で次のコマンドを入力し、サーバの秘密キー（server2key）とクライアントの秘密キー（client2key）をそれぞれ作成します。

```
keytool -genkeypair -alias server2key -keyalg RSA -keysize 2048 -validity 7360
-keystore server2.keystore
keytool -genkeypair -alias client2key -keyalg RSA -keysize 2048 -validity 7360
-keystore client2.keystore
```

手順 2　次のコマンドを入力し、キーを証明書にエクスポートします。

a. RC で、

```
keytool -export -alias server1key -keystore server1.keystore -rfc -file server1.crt
keytool -export -alias client1key -keystore client1.keystore -rfc -file client1.crt
```

b. SL で、

```
keytool -export -alias server2key -keystore server2.keystore -rfc -file server2.crt
keytool -export -alias client2key -keystore client2.keystore -rfc -file client2.crt
```

手順 3　次のコマンドを入力し、公開キーを証明書からエクスポートし、信頼ストアにインポートします。

a. RC で、

server2.crt を SL から RC（C:\SSL）にコピーした後に、次のコマンドを入力します。

```
keytool -import -file server1.crt -keystore client1.truststore
keytool -import -file client1.crt -keystore server1.truststore
keytool -import -alias client1finaltrust -file server2.crt -keystore client1.truststore
```

b. SL で、

client1.crt を RC から SL（C:\SSL）にコピーした後に、次のコマンドを入力します。

```
keytool -import -file server2.crt -keystore client2.truststore
keytool -import -file client2.crt -keystore server2.truststore
keytool -import -alias server2finaltrust -file client1.crt -keystore server2.truststore
```

（注）　RC で、

- server1.keystore、server1.truststore、client1.keystore、client1.truststore を /INSTALL_DIR/wildfly-8.2.0.Final/ServiceCatalogServer/configuration にコピーします。
- server1.keystore、server1.truststore、client1.keystore、client1.truststore を RC の /INSTALL_DIR/bin にコピーします。

（注）　SL で、

- server2.keystore、server2.truststore、client2.keystore、および client2.truststore の各ファイルを /INSTALL_DIR/wildfly-8.2.0.Final/ServiceLinkServer/configuration にコピーします。
- server2.keystore、server2.truststore、client2.keystore、および client2.truststore の各ファイルを SL の /INSTALL_DIR/bin にコピーします。

手順 4　**RequestCenter.war**（VM1）の **rcjms.properties** および **ServiceLink.war**（VM2）の **integration-server.properties** を編集します。

a. **Request Center** ファイルで、**rcjms.properties** セクションの次の変数を編集します。

- **http-remoting** という値をすべて **https-remoting** に置き換えます。
- ポート **6080** という値をすべてポート **6443** に置き換えます。

（注）　次の情報は、11.1 で SSL を有効にする場合は無視してください。

– 次のエントリを追加します。

```
## FOR SSL ##

BEEERequisitions.CLIENT_KEYSTORE=client.keystore
BEEERequisitions.CLIENT_TRUSTSTORE=client.truststore
BEEERequisitions.KEYSTORE_PASSWORD=secret
BEEERequisitions.TRUSTSTORE_PASSWORD=secret
BEEERequisitions.TRUSTSTORE_TYPE=JCEKS

## FOR SSL for VM1###
```

b. **Service Link** ファイルで、**integration-server.properties** セクションの次の変数を編集します。

– **http-remoting** という値をすべて **https-remoting** に置き換えます。

– ポート **6080** という値をすべてポート **6443** に置き換えます。

（注）　次の情報は、11.1 で SSL を有効にする場合は無視してください。

– 次のエントリを追加します。

```
## FOR SSL for VM2##

ISEEOutbound.CLIENT_KEYSTORE=client.keystore
ISEEOutbound.CLIENT_TRUSTSTORE=client.truststore
ISEEOutbound.KEYSTORE_PASSWORD=secret
ISEEOutbound.TRUSTSTORE_PASSWORD=secret
ISEEOutbound.TRUSTSTORE_TYPE=JCEKS

## FOR SSL ###
```

手順 5　添付されている **https-users.properties** および **https-roles.properties** を次の場所にコピーします。

```
C:\Install_Dir\wildfly-8.2.0.Final\standalone\configuration
C:\Install_Dir\wildfly-8.2.0.Final\ServiceCatalogServer\configuration
C:\Install_Dir\wildfly-8.2.0.Final\ServiceLinkServer\configuration
```

手順 6　**standalone-full-RC.xml** を **Standalone-full_RC_SQL.xml** または **Standalone-full_RC_Oracle.xml** に、**standalone-full-SL.xml** を **Standalone-full_SL_SQL.xml** または **Standalone-full_SL_Oracle.xml** に置き換え、それらを次の場所に配置します。これらの xml ファイルはここからダウンロードします。

```
C:\Install_Dir\wildfly-8.2.0.Final\ServiceCatalogServer\configuration
C:\Install_Dir\wildfly-8.2.0.Final\ServiceLinkServer\configuration
```

（注）　インストールが Oracle である場合、データ ソースをそれに応じて変更する必要があります。

上記のファイルの名前を **standalone-full.xml** に変更し、**Service Link Oracle** に対し、次の操作を実行します。

a. **IP_ADDRESS** および **DB_NAME** を顧客が使用する IP アドレスおよびデータベースの名前に置き換えます。

```
<connection-url>jdbc:sqlserver://IP_ADDRESS:1433;DatabaseName=DB_NAME</connection-url>
```

b. **IP_ADDRESS** をアプリケーションがインストールされているマシンの IP アドレスに変更します。

```
<outbound-socket-binding name="my-http">
          <remote-destination host="IP_ADDRESS"port="${jboss.https.port:6443}"/>
</outbound-socket-binding>
```

手順 7　次のパスにあるモジュール ファイルに依存関係を追加します。
C:\Install_Dir\wildfly-8.2.0.Final\modules\system\layers\base\io\netty\main\module.xml

```
<dependencies>
     <module name="javax.api"/>
</dependencies>
```

（注）　手順 5 から手順 7 までを VM1 と VM2 の両方に対して実行します。

手順 8　VM1 の **newscale.properties** を編集し、**isee.base.url** 変数が **https://<ip-address>:6443** に設定されていることを確認します。

（注）　この手順 9 の使用が必要なのは、11.1 の場合のみです。

手順 9　次を **startServiceCatalog.conf.cmd/startServiceCatalog.conf.sh** にコピーします。

```
## For Windows START ##

set "JAVA_OPTS=%JAVA_OPTS% -Djavax.net.ssl.keyStore=client.keystore"
set "JAVA_OPTS=%JAVA_OPTS% -Djavax.net.ssl.trustStore=client.truststore"
set "JAVA_OPTS=%JAVA_OPTS% -Djavax.net.ssl.keyStorePassword=secret"
set "JAVA_OPTS=%JAVA_OPTS% -Djavax.net.ssl.trustStorePassword=secret"
set "JAVA_OPTS=%JAVA_OPTS% -Djavax.net.ssl.trustStoreType=JCEKS"

## For Windows END ##

## For Linux START ##

JAVA_OPTS="$JAVA_OPTS -Djavax.net.ssl.keyStore=client.keystore"
JAVA_OPTS="$JAVA_OPTS -Djavax.net.ssl.trustStore=client.truststore"
JAVA_OPTS="$JAVA_OPTS -Djavax.net.ssl.keyStorePassword=secret"
JAVA_OPTS="$JAVA_OPTS -Djavax.net.ssl.trustStorePassword=secret"
JAVA_OPTS="$JAVA_OPTS -Djavax.net.ssl.trustStoreType=JCEKS"

## For Linux END ##
```

手順 10　サーバのログを消去して、Service Link、Request Center の順に再起動します。

2VM クラスタ トポロジでの SSL の有効化

手順 1　C:\SSL パスで作成を行い、ディレクトリに移動して次の手順に従います。
これにより、サーバ キー（秘密および公開）とクライアント キーのペアが作成されます。

（注）　秘密キーと公開キーを作成するために入力する情報が、クライアントとサーバの両方で一致していることを確認します。

（注）　キーの作成に使用されるデフォルトのパスワードは *secret* です。新しいパスワードを作成する場合は、*rcjms.properties*、*integration-server.properties*、および *domain.xml* ファイルにある *secret* という語を選択したパスワードに置き換えます。

(注)　名前と姓を要求されたら、ホスト コンピュータの IP アドレスを入力します。

a. 次のコマンドを入力し、サーバの秘密キー（serverkey）とクライアントの秘密キー（clientkey）をそれぞれ作成します。

VM1（RC および SL）で、

```
keytool -genkeypair -alias server1key -keyalg RSA -keysize 2048 -validity 7360 -keystore
server1.keystore
keytool -genkeypair -alias client1key -keyalg RSA -keysize 2048 -validity 7360 -keystore
client1.keystore
```

RC2 で、

```
keytool -genkeypair -alias server2key -keyalg RSA -keysize 2048 -validity 7360 -keystore
server2.keystore
keytool -genkeypair -alias client2key -keyalg RSA -keysize 2048 -validity 7360 -keystore
client2.keystore
```

b. 次のコマンドを入力し、キーを証明書にエクスポートします。

VM1（RC および SL）で、

```
keytool -export -alias server1key -keystore server1.keystore -rfc -file server1.crt
keytool -export -alias client1key -keystore client1.keystore -rfc -file client1.crt
```

RC2 で、

```
keytool -export -alias server2key -keystore server2.keystore -rfc -file server2.crt
keytool -export -alias client2key -keystore client2.keystore -rfc -file client2.crt
```

c. 次のコマンドを入力し、公開キーを証明書からエクスポートし、信頼ストアにインポートします。

VM1 で、

server2.crt と client2.crt を VM2 から VM1 にコピーします。

```
Copy the server2.keystore, client2.keystore and server2.truststore, client2.truststore to
/<INSTALL_DIR>/wildfly-8.2.0.Final
Copy the server2.keystore,client2.keystore and server2.truststore, client2.truststore to
/<INSTALL_DIR>/wildfly-8.2.0.Final/domain/configuration
```

VM1 で次のコマンドを入力します。

```
keytool -import -file server2.crt -keystore client1.truststore
keytool -import -file client2.crt -keystore server1.truststore
keytool -import -alias client1finaltrust -file server1.crt -keystore client1.truststore
keytool -import -alias server1finaltrust -file client1.crt -keystore server1.truststore
```

VM2 で、

server1.crt と client1.crt を VM1 から VM2 にコピーします。

```
Copy the server1.keystore, client1.keystore and server1.truststore, client1.truststore to
/<INSTALL_DIR>/wildfly-8.2.0.Final
Copy the server1.keystore, client1.keystore and server1.truststore to
/<INSTALL_DIR>/wildfly-8.2.0.Final/domain/configuration
```

VM2 で次のコマンドを入力します。

```
keytool -import -file server1.crt -keystore client2.truststore
keytool -import -file client1.crt -keystore server2.truststore
keytool -import -alias client2finaltrust -file server2.crt -keystore client2.truststore
keytool -import -alias server2finaltrust -file client2.crt -keystore server2.truststore
```

手順 2　**client.keystore** および **client.truststore** へのパスを **password** および **truststoretype** とともに入力します。

(注)　HC2 の場合、物理ファイル client2.keystore、client2.truststore、server2.keystore、および server2.truststore の名前を VM1 のものと同じ名前に変更します。これは、VM1 から VM2 に rcjms.properties を持つバイナリが展開されるためです。

VM1 および VM2 で、

startServiceCatalogCluster.conf.sh または startServiceCatalogCluster.conf.bat に次のコードを入力します。

```
rem # Properties for SSL over Wildfly
## For Windows START ##

set "JAVA_OPTS=%JAVA_OPTS% \-Djavax.net.ssl.keyStore=client1.keystore"
set "JAVA_OPTS=%JAVA_OPTS% \-Djavax.net.ssl.trustStore=client1.truststore"
set "JAVA_OPTS=%JAVA_OPTS% \-Djavax.net.ssl.keyStorePassword=secret"
set "JAVA_OPTS=%JAVA_OPTS% \-Djavax.net.ssl.trustStorePassword=secret"
set "JAVA_OPTS=%JAVA_OPTS% \-Djavax.net.ssl.trustStoreType=JCEKS"

## For Windows End##

## For Linux START ##

JAVA_OPTS="$JAVA_OPTS -Djavax.net.ssl.keyStore=client.keystore"
JAVA_OPTS="$JAVA_OPTS -Djavax.net.ssl.trustStore=client.truststore"
JAVA_OPTS="$JAVA_OPTS -Djavax.net.ssl.keyStorePassword=secret"
JAVA_OPTS="$JAVA_OPTS -Djavax.net.ssl.trustStorePassword=secret"
JAVA_OPTS="$JAVA_OPTS -Djavax.net.ssl.trustStoreType=JCEKS"

## For Linux END ##
```

手順 3　**/content/RequestCenter.war** および **/content/ISEE.war** に次の変更を加えます。
次に、**/INSTALL_DIR/dist** にある **RequestCenter.war** および **ISEE.war** を解凍し、war ファイルを再作成します。

a. **RC** については **rcjms.properties** に、**SL** については **integrationserver.properties** に、次の値を入力します。
   - **http-remoting** という値をすべて **https-remoting** に置き換えます。
   - ポート **6080** という値をすべてポート **6443** に置き換えます。

(注)　次の情報は、11.1 で SSL を有効にする場合は無視してください。

   - **rcjms.properties** および **integrationserver.properties** に次のエントリを追加します。

```
## FOR RC in rcjms.properties ###
BEEERequisitions.CLIENT_KEYSTORE=client1.keystore
BEEERequisitions.CLIENT_TRUSTSTORE=client1.truststore
BEEERequisitions.KEYSTORE_PASSWORD=secret
BEEERequisitions.TRUSTSTORE_PASSWORD=secret
```

```
BEEERequisitions.TRUSTSTORE_TYPE=JCEKS
## END OF FOR SSL ##

## FOR SL in integrationserver.properties ###
ISEEOutbound.CLIENT_KEYSTORE=client2.keystore
ISEEOutbound.CLIENT_TRUSTSTORE=client2.truststore
ISEEOutbound.KEYSTORE_PASSWORD=secret
ISEEOutbound.TRUSTSTORE_PASSWORD=secret
ISEEOutbound.TRUSTSTORE_TYPE=JCEKS
## END OF FOR SSL ##
```

b. **newscale.properties** を編集し、**isee.base.url** 変数が **https://<ip-address>:6443** に設定されていることを確認します。

手順 4　このページに添付されている **https-users.properties** および **https-roles.properties** をコピーするか、次の手順に従って独自に作成します。

a. 添付されている **https-users.properties** および **https-roles.properties** を次の場所に移動します。

JBOSS_HOME/domain/configuration

（注）　*JBOSS_HOME* は、ユーザによってクライアント VM に設定される変数です。

（注）　ユーザ名およびパスワードを新規作成するか既存のものを変更し、次の手順に従います。

b. 正しい **16 進数のパスワード**を、httpRealm で次のコマンドを使用して作成した **https-user.properties ファイル**に作成します。

```
<INSTALL_DIR\wildfly-8.2.0.Final\modules\system\layers\base\org\jboss\sasl\main>java
-classpath jboss-sasl-1.0.4.Final.jar org.jboss.sasl.util.UsernamePasswordHashUtil
jmsuser httpsRealm newscale

The above command will generate the below line:
jmsuser=b8b4553774410fded0a8c8f317217f44


<INSTALL_DIR\wildfly-8.2.0.Final\modules\system\layers\base\org\jboss\sasl\main>java
-classpath jboss-sasl-1.0.4.Final.jar org.jboss.sasl.util.UsernamePasswordHashUtil
adminuser httpsRealm admin123

The above command will generate the below line:
adminuser=ad61acb853db393c03846f3670b999b5


<INSTALL_DIR\wildfly-8.2.0.Final\modules\system\layers\base\org\jboss\sasl\main>java
-classpath jboss-sasl-1.0.4.Final.jar org.jboss.sasl.util.UsernamePasswordHashUtil
HOST2 httpsRealm HOST2
The above command will generate the below line:
HOST2=43e484d6af217b513fccf36c8b13cb27

<INSTALL_DIR>\wildfly-8.2.0.Final\modules\system\layers\base\org\jboss\sasl\main>java
-classpath jboss-sasl-1.0.4.Final.jar org.jboss.sasl.util.UsernamePasswordHashUtil
HOST1 httpsRealm HOST1HOST1=91e12144aab41d877d778aff1b1921bb
```

手順 5　**JMS クレデンシャル**、**管理者クレデンシャル**、**HOST1 のクレデンシャル**、および **HOST2 のクレデンシャル** を JBOSS_HOME/domain/configuration の **https-user.properties** にコピーします。

手順 6　**https-users.properties** および **https-roles.properties** を JBOSS_HOME/domain/configuration から次の場所にコピーします。

```
<INSTALL_DIR>/wildfly-8.2.0.Final/domain/servers/server-host1-RC/configuration
<INSTALL_DIR>/wildfly-8.2.0.Final/domain/servers/server-host1-SL/configuration
```

（注）　インストールが Oracle である場合、データ ソースをそれに応じて変更する必要があります。

手順 7　**domain.xml** および **hostva_backup.xml** または **hostva.xml** を、添付されている **VM1 の 2VM_SQL_domain.xml または 4VM_Oracle_domain.xml** および **2VM_hostva_backup.xml** に置き換えます。また、VM2 内の **host2_backup.xml** または **host2.xml** を添付されている **2VM_host2_backup.xml** に置き換えます。これらの xml ファイルはここからダウンロードします。

a. **IP_ADDRESS** および **DB_NAME** を顧客が使用する IP アドレスおよびデータベースの名前に置き換えます。

```
<connection-url>jdbc:sqlserver://IP_ADDRESS:1433;DatabaseName=DB_NAME</connection-url>
```

b. **IP_ADDRESS** をアプリケーションがインストールされているマシンの IP アドレスに置き換えます。

```
<outbound-socket-binding name="my-http">
        <remote-destination host="IP_ADDRESS" port="${jboss.https.port:6443}"/>
</outbound-socket-binding>
```

手順 8　次のパスにあるモジュール ファイルに依存関係を追加します。**VM1 と VM2 両方の <INSTALL_DIR>\wildfly-8.2.0.Final\modules\system\layers\base\io\netty\main\module.xml**

```
<dependencies>
<module name="javax.api"/>
</dependencies>
```

手順 9　サーバのログを消去して、Service Link、Request Center の順に再起動します。

4VM クラスタ トポロジでの SSL の有効化

手順 1　C:\SSL パスで作成を行い、ディレクトリに移動して次の手順に従います。
これにより、サーバ キー（秘密および公開）とクライアント キーのペアが作成されます。

VM1（ドメイン コントローラ）について、

a. 次のコマンドを入力し、サーバの秘密キー（server.keystore）とクライアントの秘密キー（client.keystore）をそれぞれ作成します。

```
keytool -genkeypair -alias serverkey -keyalg RSA -keysize 2048 -validity 7360
-keystore server.keystore
keytool -genkeypair -alias clientkey -keyalg RSA -keysize 2048 -validity 7360
-keystore client.keystore
```

（注）　秘密キーと公開キーを作成するために入力する情報が、クライアントとサーバの両方で一致していることを確認します。

(注)　キーの作成に使用されるデフォルトのパスワードは *secret* です。新しいパスワードを作成する場合は、*rcjms.properties*、*integration-server.properties*、および *domain.xml* ファイルにある *secret* という語を選択したパスワードに置き換えます。

(注)　名前と姓を要求されたら、ホスト コンピュータの IP アドレスを入力します。

b. 次のコマンドを入力し、キーを証明書にエクスポートします。

```
keytool -export -alias serverkey -keystore server.keystore -rfc -file server.crt
keytool -export -alias clientkey -keystore client.keystore -rfc -file client.crt
```

VM2（ホスト コントローラ - HOST1）について、

a. 次のコマンドを入力し、サーバの秘密キー（server1.keystore）とクライアントの秘密キー（client1.keystore）をそれぞれ作成します。

```
keytool -genkeypair -alias server1key -keyalg RSA -keysize 2048 -validity 7360
-keystore server1.keystore
keytool -genkeypair -alias client1key -keyalg RSA -keysize 2048 -validity 7360
-keystore client1.keystore
```

(注)　秘密キーと公開キーを作成するために入力する情報が、クライアントとサーバの両方で一致していることを確認します。

(注)　キーの作成に使用されるデフォルトのパスワードは *secret* です。新しいパスワードを作成する場合は、*rcjms.properties*、*integration-server.properties*、および *domain.xml* ファイルにある *secret* という語を選択したパスワードに置き換えます。

(注)　名前と姓を要求されたら、ホスト コンピュータの IP アドレスを入力します。

b. 次のコマンドを入力し、キーを証明書にエクスポートします。

```
keytool -export -alias server1key -keystore server1.keystore -rfc -file server1.crt
keytool -export -alias client1key -keystore client1.keystore -rfc -file client1.crt
```

VM3（Service Link – スタンドアロン）について、

a. 次のコマンドを入力し、サーバの秘密キー（serverSL.keystore）とクライアントの秘密キー（clientSL.keystore）をそれぞれ作成します。

```
keytool -genkeypair -alias serverSLkey -keyalg RSA -keysize 2048 -validity 7360
-keystore serverSL.keystore
keytool -genkeypair -alias clientSLkey -keyalg RSA -keysize 2048 -validity 7360
-keystore clientSL.keystore
```

(注)　秘密キーと公開キーを作成するために入力する情報が、クライアントとサーバの両方で一致していることを確認します。

(注)　キーの作成に使用されるデフォルトのパスワードは *secret* です。新しいパスワードを作成する場合は、*rcjms.properties*、*integration-server.properties*、および *domain.xml* ファイルにある *secret* という語を選択したパスワードに置き換えます。

(注)　名前と姓を要求されたら、ホスト コンピュータの IP アドレスを入力します。

b. 次のコマンドを入力し、キーを証明書(serverSL.crt および clientSL.crt)にエクスポートします。

```
keytool -export -alias serverSLkey -keystore serverSL.keystore -rfc -file serverSL.crt
keytool -export -alias clientSLkey -keystore clientSL.keystore -rfc -file clientSL.crt
```

手順 2　全 VM のキーを生成し、次のコマンドを入力して、公開キーを証明書からエクスポートし、信頼ストアにインポートします。

a. server.crt と client.crt を VM1 から VM2 にコピーします。VM2 で次のコマンドを入力します。

```
keytool -import -alias serverTrustclient1 -file server.crt -keystore
client1.truststore
keytool -import -alias clientTrustserver1 -file client.crt -keystore
server1.truststore
```

b. serverSL.crt と clientSL.crt を VM3 から VM2 にコピーします。VM2 で次のコマンドを入力します。

```
keytool -import -alias serverSLTrustclient1 -file serverSL.crt -keystore
client1.truststore
keytool -import -alias clientSLTrustserver1 -file clientSL.crt -keystore
server1.truststore
```

c. server1.crt と client1.crt を VM2 から VM1 にコピーします。VM1 で次のコマンドを入力します。

```
keytool -import -alias server1Trustclient -file server1.crt -keystore
client.truststore
keytool -import -alias client1Trustserver -file client1.crt -keystore
server.truststore
```

d. server1.crt と client1.crt を VM2 から VM3 にコピーします。VM3 で次のコマンドを入力します。

```
keytool -import -alias server1TrustclientSL -file server1.crt -keystore
clientSL.truststore
keytool -import -alias client1TrustserverSL -file client1.crt -keystore
serverSL.truststore
```

e. serverSL.crt および clientSL.crt が VM3 にあることを確認します。VM3 で次のコマンドを入力します。

```
keytool -import -alias serverSLTrustclientSL -file serverSL.crt -keystore
clientSL.truststore
keytool -import -alias clientSLTrustserverSL -file clientSL.crt -keystore
serverSL.truststore
```

手順 3　キーストア ファイルおよび信頼ストア ファイルを以下の場所に配置します。

a. VM1 と VM2 の信頼ストアおよびキーストアを同じマシンの次の場所にコピーします。

```
C:\Install_Dir
C:\Install_Dir\bin
C:\Install_Dir\wildfly-8.2.0.Final
C:\Install_Dir\wildfly-8.2.0.Final\domain\configuration
C:\Install_Dir\wildfly-8.2.0.Final\domain\servers\configuration
```

b. VM3 の信頼ストアおよびキーストアを次の場所にコピーします。

```
C:\Install_Dir
C:\Install_Dir\bin
C:\Install_Dir\wildfly-8.2.0.Final
C:\Install_Dir\wildfly-8.2.0.Final\ServiceLinkServer\configuration
C:\Install_Dir\wildfly-8.2.0.Final\standalone\configuration
```

手順 4　起動スクリプトの変更:

a. **VM1** の **startServiceCatalogCluster.conf.cmd または startServiceCatalogCluster.conf.sh** に次のスニペットを追加します。

```
## For Windows START ##

set "JAVA_OPTS=%JAVA_OPTS% -Djavax.net.ssl.keyStore=client.keystore"
set "JAVA_OPTS=%JAVA_OPTS% -Djavax.net.ssl.trustStore=client.truststore"
set "JAVA_OPTS=%JAVA_OPTS% -Djavax.net.ssl.keyStorePassword=secret"
set "JAVA_OPTS=%JAVA_OPTS% -Djavax.net.ssl.trustStorePassword=secret"
set "JAVA_OPTS=%JAVA_OPTS% -Djavax.net.ssl.trustStoreType=JCEKS"

## For Windows END ##


## For Linux START ##

JAVA_OPTS="$JAVA_OPTS -Djavax.net.ssl.keyStore=client.keystore"
JAVA_OPTS="$JAVA_OPTS -Djavax.net.ssl.trustStore=client.truststore"
JAVA_OPTS="$JAVA_OPTS -Djavax.net.ssl.keyStorePassword=secret"
JAVA_OPTS="$JAVA_OPTS -Djavax.net.ssl.trustStorePassword=secret"
JAVA_OPTS="$JAVA_OPTS -Djavax.net.ssl.trustStoreType=JCEKS"

## For Linux END ##
```

b. **VM2** の **startServiceCatalogCluster.conf.cmd または startServiceCatalogCluster.conf.sh** に次をコピーします。

```
## For Windows START ##

set "JAVA_OPTS=%JAVA_OPTS% -Djavax.net.ssl.keyStore=client1.keystore"
set "JAVA_OPTS=%JAVA_OPTS% -Djavax.net.ssl.trustStore=client1.truststore"
set "JAVA_OPTS=%JAVA_OPTS% -Djavax.net.ssl.keyStorePassword=secret"
set "JAVA_OPTS=%JAVA_OPTS% -Djavax.net.ssl.trustStorePassword=secret"
set "JAVA_OPTS=%JAVA_OPTS% -Djavax.net.ssl.trustStoreType=JCEKS"

## For Windows END ##


## For Linux START ##

JAVA_OPTS="$JAVA_OPTS -Djavax.net.ssl.keyStore=client1.keystore"
JAVA_OPTS="$JAVA_OPTS -Djavax.net.ssl.trustStore=client1.truststore"
JAVA_OPTS="$JAVA_OPTS -Djavax.net.ssl.keyStorePassword=secret"
JAVA_OPTS="$JAVA_OPTS -Djavax.net.ssl.trustStorePassword=secret"
JAVA_OPTS="$JAVA_OPTS -Djavax.net.ssl.trustStoreType=JCEKS"

## For Linux END ##
```

c. **VM3** の **startServiceLink.conf.cmd または startServiceLink.conf.sh** に次をコピーします。

```
## For Windows START ##
set "JAVA_OPTS=%JAVA_OPTS% -Djavax.net.ssl.keyStore=clientSL.keystore"
set "JAVA_OPTS=%JAVA_OPTS% -Djavax.net.ssl.trustStore=clientSL.truststore"
set "JAVA_OPTS=%JAVA_OPTS% -Djavax.net.ssl.keyStorePassword=secret"
set "JAVA_OPTS=%JAVA_OPTS% -Djavax.net.ssl.trustStorePassword=secret"
```

```
set "JAVA_OPTS=%JAVA_OPTS% -Djavax.net.ssl.trustStoreType=JCEKS"
## For Windows END ##


## For Linux START ##
JAVA_OPTS="$JAVA_OPTS -Djavax.net.ssl.keyStore=clientSL.keystore"
JAVA_OPTS="$JAVA_OPTS -Djavax.net.ssl.trustStore=clientSL.truststore"
JAVA_OPTS="$JAVA_OPTS -Djavax.net.ssl.keyStorePassword=secret"
JAVA_OPTS="$JAVA_OPTS -Djavax.net.ssl.trustStorePassword=secret"
JAVA_OPTS="$JAVA_OPTS -Djavax.net.ssl.trustStoreType=JCEKS"

## For Linux END ##
```

手順 5　全 VM の **C:\4VM_SSL_Domain\wildfly-8.2.0.Final\bin\jboss-cli.xml** に次の変更を加えます。

（注）　次のコード スニペット内のタグ IP_ADDRESS を各 VM の IP アドレスに変更します。

```
<default-controller>
   <protocol>https-remoting</protocol>
   <host>IP_ADDRESS</host>
   <port>9993</port>
</default-controller>
```

手順 6　添付されている 4VM-https-roles.properties および 4VM-https-users.properties を（それぞれ **https-roles.properties** および **https-users.properties** として）、VM1 および VM2 の **C:\4VM_SSL_Domain\wildfly-8.2.0.Final\domain\configuration** にコピーします。

手順 7　設定ファイルの変更:

a.　添付されている **4VM_SQL_domain.xml または 4VM_Oracle_domain.xml** を（**domain.xml** として）VM1 にコピーします。

- **<outbound-socket-binding name="remote-http">** セクションで、**IP アドレス**（デフォルトでは 10.76.82.36）を VM1 の IP アドレスに変更します。
- **<interfaces>** セクションで、**IP アドレス**（デフォルトでは 10.76.82.36）を VM1 の IP アドレスに置き換えます。
- **<datasource jndi-name="java:/REQUESTCENTERDS" pool-name="REQUESTCENTERDS" enabled="true">** セクションで、**IP アドレス**（デフォルトでは 10.76.81.198）とデータベースの名前を、（データベース サーバがある）マシンの IP アドレスと現在使用されているデータベースの名前に置き換えます。

b.　添付されている 4VM_host_default.xml を（**host_default.xml** として）VM1 にコピーします。

- **<interface name="unsecure">** セクションで、**IP アドレス**（デフォルトでは 10.76.82.36）を VM1 の IP アドレスに置き換えます。

c.　VM2 で、添付されている 4VM_host1_backup.xml を（**host1.xml または host1_backup.xml** として）コピーし、次の変更を加えます。

- **<interfaces>** セクションで、**IP アドレス**（デフォルトでは 10.76.82.38）を VM2 の IP アドレスに置き換えます。
- **<domain-controller>** セクションで、**IP アドレス**（デフォルトでは 10.76.82.36）を VM1 の IP アドレスに置き換えます。

d.　VM3 で、添付されている 4VM-standalone-SL.xml を（**standalone-full.xml** として）コピーし、以下の変更を行います。

- **<outbound-socket-binding name="my-http">** セクションで、**IP アドレス**（デフォルトでは 10.76.82.37）を VM3 の IP アドレスに置き換えます。

手順 8　WAR ファイルの変更:

a. **VM2** で、展開されているバージョンの **RequestCenter.war** に次の変更を加えます。

- **Request Center** ファイルで、**rcjms.properties** セクションの次の変数を編集します。
  - **http-remoting** という値をすべて **https-remoting** に置き換えます。
  - ポート **6080** という値をすべてポート **6443** に置き換えます。

（注） 次の情報は、11.1 で SSL を有効にする場合は無視してください。

  - 次のコードを追加します。

```
BEEERequisitions.CLIENT_KEYSTORE=client1.keystore
BEEERequisitions.CLIENT_TRUSTSTORE=client1.truststore
BEEERequisitions.KEYSTORE_PASSWORD=secret
BEEERequisitions.TRUSTSTORE_PASSWORD=secret
BEEERequisitions.TRUSTSTORE_TYPE=JCEKS
```

IP-ADDRESS を **VM3** の IP アドレスに、**newscale.properties** 内の**ポート 6443** を次のキーと値のペアに置き換えます。**isee.base.url=https://<IP-ADDRESS>:6443**

b. VM3 の展開ディレクトリにある **ServiceLink.war** に次の変更を加えます。

- **Service Link** ファイルで、**integration-server.properties** セクションの次の変数を編集します。
  - **http-remoting** という値をすべて **https-remoting** に置き換えます。
  - ポート **6080** という値をすべてポート **6443** に置き換えます。

c. **VM1** に次の変更を加えます。

- **INSTALL_DIR/dist** フォルダ内の **RequestCenter.war** および **ISEE.war** を解凍する必要があります。
  Request Center ファイルで、**rcjms.properties** セクションの次の変数を編集します。
  - **http-remoting** という値をすべて **https-remoting** に置き換えます。
  - ポート **6080** という値をすべてポート **6443** に置き換えます。

（注） 次の情報は、11.1 で SSL を有効にする場合は無視してください。

  - 次のコードを追加します。

```
BEEERequisitions.CLIENT_KEYSTORE=client1.keystore
BEEERequisitions.CLIENT_TRUSTSTORE=client1.truststore
BEEERequisitions.KEYSTORE_PASSWORD=secret
BEEERequisitions.TRUSTSTORE_PASSWORD=secret
BEEERequisitions.TRUSTSTORE_TYPE=JCEKS
```

（注） CLIENT_KEYSTORE および CLIENT_TRUSTSTORE については、VM1 のキーを参照してください。

  - **IP-ADDRESS** を **VM3** の IP アドレスに、newscale.properties 内のポート **6443** を次のキーと値のペアに置き換えます。**isee.base.url=https://<IP-ADDRESS>:6443**
  - 上述の変更が完了したら、RequestCenter.war を再度圧縮し、展開スクリプトを実行します。展開スクリプトを実行する際に、ユーザは、証明書を永久的に受け入れるか、一時的に受け入れるかを選択できます。

手順 9　次のパスにあるモジュール ファイルに以下の依存関係を追加します。全 VM の **C:\Install_Dir\wildfly-8.2.0.Final\modules\system\layers\base\io\netty\main\module.xml**

```
<dependencies>
     <module name="javax.api"/>
</dependencies>
```

手順 10　**Linux 環境**で **4VM クラスタ** をセットアップするには、**jboss-cli.sh** を実行し、証明書を永久的に受け入れます。

11.1 Fix でのクラスタのセットアップ

この項では、11.1 fix のみのクラスタ セットアップの情報を提供します。

（注）　Linux システムの場合、端末でコマンドを実行する際は、バック スラッシュ（\）をスラッシュ（/）に置き換えてください。

（注）　次の表記を使用します。VM1 = DC、VM2 = SL、VM3 = HC1、VM4 = HC2。

2VM でのクラスタのセットアップ

手順 1　VM1 の Internet Explorer ブラウザで、VM1 の **RequestCenter** URL を入力します。

（注）　クラスタ内で 2 つの VM サーバが起動していることを確認します。

手順 2　表示されたウィンドウから [このサイトの閲覧を続行する (推奨されません)。(Continue to this website (not recommended))] を選択します。

手順 3　証明書情報ウィンドウの下部のペインで [証明書（Certificates）] をクリックします。

手順 4　表示されたウィンドウの [詳細（Details）] タブを選択して、[ファイルへコピー（Copy to File）] をクリックします。

手順 5　[証明書のエクスポート ウィザード（Certificate Export Wizard）] で [次へ（Next）] をクリックします。

手順 6　[証明書のエクスポート ウィザード（Certificate Export Wizard）] で **DER encoded binary X.509.cer** を選択します。

手順 7　[コピー（Copy）] をクリックして、たとえば次のような証明書をエクスポートするパスを参照します。

```
c:\root.cer
```

手順 8　VM1 で、**JAVA_LOCATION\jdk\jre\bin** ディレクトリから次のコマンドを入力し、cacerts ファイル内のエントリを決定します。

```
C:\Java\jdk1.7.0_71\jre\bin>keytool -list -keystore ..\lib\security\cacerts
```

手順 9　キーストアのパスワードとして **changeit** を入力します。

（注）　cacerts ファイル内のエントリの数と cacerts ファイルのサイズを書き留めておいてください。

手順 10　次のコマンドを実行して、サーバ証明書を追加します。

（注）　Java ランタイム環境（JRE）は、サーバ証明書がキーストアに追加されるまで、その存在を認識しません。

```
keytool -import -file C:/root.cer -keystore C:\Java\jdk1.7.0_71\jre\lib\security\cacerts
```

手順 11　キーストアのパスワードとして **changeit** を入力します。

手順 12　**bin** ディレクトリからの次のコマンドを入力し、cacerts ファイル内のエントリを決定します。

```
C:\Java\jdk1.7.0_71\jre\bin>keytool -list -keystore ..\lib\security\cacerts
```

手順 13　キーストアのパスワードとして **changeit** を入力します。

（注）　cacerts 内のエントリの数と cacerts ファイルのサイズを確認します。エントリの数は 1 増えており、cacerts ファイルのサイズは 1 KB 増えている必要があります。

これにより、プライベート ルート証明書が信頼された認証局としてエクストラネット サーバの cacerts キーストアに追加されたことが確認できます。

手順 14　2VM クラスタについて、

VM1 から VM2 に証明書をコピーし、同じ手順を繰り返します。次に例を示します。

```
VM1 = DC+HC1 and VM2 = HC2
```

手順 15　サーバを再起動し、クラスタ操作を実行します。

4VM でのクラスタのセットアップ

前提条件

- 使用する JDK のデフォルトのバージョンは、1.7.0_79 です。
- serverSL.crt および clientSL.crt は、SSL モードの 4VM クラスタでスタンドアロンの Service Link をセットアップする際に生成される証明書です。

4VM でクラスタをセットアップするには、次の手順に従います。VM2 で生成される serverSL.crt と clientSL.crt は、次の手順で cacerts ファイルに追加されます。

手順 1　コマンド プロンプトで、現在のディレクトリを指す java の bin フォルダで次のコマンドを入力します。

（注）　Linux の場合は、*Terminal* を使用する必要があります。

```
cd C:\Java\jdk1.7.0_79\jre\bin
```

手順 2　次のコマンドを入力し、cacerts ファイル内のエントリを決定します。
要求された場合、キーストアのパスワードとして **changeit** を入力します。

（注）　検証のため、ファイルのサイズを書き留めておきます。

```
keytool -list -keystore ..\lib\security\cacerts
```

手順 3　次のコマンドを入力し、**clientSL.crt** を cacerts ファイルに追加します。

（注）　.crt ファイルは C:\SSL ディレクトリなどにあります。

```
keytool -import -alias clientSL -file C:\SSL\clientSL.crt -keystore
C:\Java\jdk1.7.0_79\jre\lib\security\cacerts
```

手順 4　次のコマンドを入力し、**serverSL.crt** を cacerts ファイルに追加します。

（注）　.crt ファイルは C:\SSL ディレクトリなどにあります。

```
keytool -import -alias serverSL -file C:\SSL\serverSL.crt -keystore
C:\Java\jdk1.7.0_79\jre\lib\security\cacerts
```

手順 5　次のコマンドを入力し、cacerts ファイル内のエントリを決定します。
要求された場合、キーストアのパスワードとして **changeit** を入力します。

```
keytool -list -keystore ..\lib\security\cacerts
```

（注）　cacerts ファイル内のエントリの数が 2 つ増えていることを確認します。また、cacerts ファイルのサイズは 2 KB 増加しています。

手順 6　**変更された cacerts** ファイルを VM2 からコピーし、VM1、VM3、VM4 のそれぞれ該当する場所にある cacerts ファイルと置き換えます。

（注）　変更された cacerts ファイルに置き換える前に、必要に応じ、cacerts ファイルのバックアップを取ります。

手順 7　すべてのサーバを再起動し、クラスタ スクリプトを実行してクラスタを起動します。

2VM クラスタ上の SSL が有効な Wildfly アプリケーション サーバに接続するための SSL が有効な Apache Httpd

手順 1　**2VM の SSL** が有効化されたクラスタをセットアップします。「2VM でのクラスタのセットアップ」を*参照してください*。

手順 2　host1 と host2 でサーバを起動し、展開プロセスを完了します。

手順 3　WildFly クラスタとは別の VM に **Apache httpd バージョン 2.4.18** をダウンロードおよびインストールします。*次の Web サイトを参照してください。*
https://www.apachehaus.com/cgi-bin/download.plx?dli=gWy82MONVWy0kej9SWYZFbJVlUGRVYSZ1YxIUN

- C:\Apache24\bin>httpd -version
- Server version: Apache/2.4.18 (Win32)
- Server built: Jan 28 2016 09:58:25

手順 4　Windows 上の **Apache** で SSL を有効化します。

a. 必要なもの:
- SSL サポートを含む Apache のコピー。
- OpenSSL のコピー。
- openssl.cnf ファイル。

(注) ダウンロードされた Apache には SSL サポートがあるので、その他のモジュールを別途ダウンロードしないでください。

b. openSSL の **openssl.cnf** ファイルをディレクトリ **C:\Apache\bin** にコピーします。

手順 5　**自己署名証明書**を作成します。

(注) 証明書のために作成される各種のファイルは、同じ名前に異なる拡張子が付いたものになります。

(注) 次のコマンド例で使用されている *bob* という名前は、必要に応じて置き換えてください。

a. 新しい証明書を作成するには、コマンド プロンプトで OpenSSL を含むディレクトリ（たとえば *C:\Apache\bin*）に切り替えた後、次のコマンドを入力します。

```
openssl req -config openssl.cnf -new -out bob.csr -keyout bob.pem
```

b. 多くの質問に答えるよう要求されますが、次以外は空白のままでかまいません。
- PEM パスフレーズ: 生成する秘密キー（bob.pem）に関連付けられるパスワードです。次の手順でのみ使用されます。お好きなように設定してください。
- 共通名: この証明書に関連付けられる完全修飾ドメイン名です。この例では、IP アドレスを使用しています。

コマンドが完了すると、フォルダには **bob.csr** および **bob.pem** という 2 つのファイルが作成されます。

c. Apache が使用するためのパスワード保護のないキーを作成します。

```
openssl rsa -in bob.pem -out bob.key
```

上記で作成したパスワードを要求されます。その後、フォルダには **bob.key** というファイルが作成されます。

d. 次のコマンドを入力し、Apache にも必要な **X.509** 証明書を作成します。

```
openssl x509 -in bob.csr -out bob.cert -req -signkey bob.key -days 365
```

e. Apache が SSL を有効にするために使用する自己署名証明書が作成されます。ファイルは、たとえば次のようなパスに追加されています。
   – C:\Apache24\bin
   – C:\Apache24\conf
   – C:\Apache24\conf\ssl.

手順 6　ローカルにある **C:\Apache24\conf\httpd.conf** および **C:\Apache24\conf\extra\httpd-ssl.conf** を、このページに添付されている対応するファイルに置き換えます。

（注）　*bob* という名前は、証明書/キーに与えた名前に変更できます。

手順 7　**httpd.conf** と **httpd-ssl.conf** の両方で、**IP アドレス**をお使いの環境に応じて置き換えます。

手順 8　次のコマンドを入力し、Apache Httpd を起動します。

```
C:\Apache24\bin>httpd -k start
```

手順 9　次の URL でテストします。https://<ip>/RequestCenter

手順 10　Apache httpd Web サーバを停止するには、次を入力します。

```
C:\Apache24\bin>httpd -k stop
```

手順 11　Linux 環境では、添付で追加されたファイルを置き換えます。

スクリプトの変更

スタンドアロン モード（標準およびカスタム）の Wildfly アプリケーション サーバ上で SSL を有効化したら、動作が可能になるよう、次の変更を加えます。

手順 1　パス **C:\Installation Directory\wildfly-8.2.0.Final\ServiceCatalogServer\configuration\standalone-full.xml** にある **standalone-full.xml** の次のコード スニペットを置き換えます。

```
<management-interfaces>
     <http-interface security-realm="httpsRealm">
               <socket interface="management" port="9990" secure-port="9993"/>
     </http-interface>
</management-interfaces>
```

置き換え後：

```
<management-interfaces>
     <http-interface security-realm="ManagementRealm" http-upgrade-enabled="true">
               <socket-binding http="management-https"/>
     </http-interface>
</management-interfaces>
```

手順 2　パス **C:\Installation Directory\wildfly-8.2.0.Final\ServiceLinkServer\configuration\standalone-full.xml**. にある **standalone-full.xml** の次のコード スニペットを置き換えます。

```
<management-interfaces>
     <http-interface security-realm="httpsRealm">
               <socket interface="management" port="7990" secure-port="7443"/>
     </http-interface>
</management-interfaces>
```

置き換え後:

```
<management-interfaces>
     <http-interface security-realm="ManagementRealm" http-upgrade-enabled="true">
             <socket-binding http="management-https"/>
     </http-interface>
</management-interfaces>
```

手順 3　**stopServiceCatalog.cmd** または **stopServiceCatalog.sh** で、**CONTROLLER_PORT** 変数を **9993** に設定します。

手順 4　**stopServiceLink.cmd** または **stopServiceLink.sh** で、**CONTROLLER_PORT** 変数を **7443** に設定します。