Informatica

Informatica® Multidomain MDM
10.4 HotFix 3

ビジネスエンティティサービスガイド

Informatica Multidomain MDM ビジネスエンティティサービスガイド
10.4 HotFix 3
2021 年 10 月

発行日: 2022-05-05

目次

序文

Web サービスとして使用可能なビジネスエンティティサービスの詳細については、Informatica(R) *Multidomain MDM ビジネスエンティティのサービスガイド*を参照してください。このガイドを使用して、Enterprise JavaBeans（EJB）、Representational State Transfer（REST）、Simple Object Access Protocol（SOAP）およびビジネスエンティティデータで操作を実行するための外部 Web サービス呼び出しの使用方法について学習します。ビジネスエンティティサービス呼び出しを行うためのカスタムユーザーインタフェースの設定についても学習できます。

Informatica のリソース

Informatica は、Informatica Network やその他のオンラインポータルを通じてさまざまな製品リソースを提供しています。リソースを使用して Informatica 製品とソリューションを最大限に活用し、その他の Informatica ユーザーや各分野の専門家から知見を得ることができます。

Informatica Network

Informatica Network は、Informatica ナレッジベースや Informatica グローバルカスタマサポートなど、多くのリソースへの入口です。Informatica Network を利用するには、https://network.informatica.com にアクセスしてください。

Informatica Network メンバーは、次のオプションを利用できます。

- ナレッジベースで製品リソースを検索できます。
- 製品の提供情報を表示できます。
- サポートケースを作成して確認できます。
- 最寄りの Informatica ユーザーグループネットワークを検索して、他のユーザーと共同作業を行えます。

Informatica ナレッジベース

Informatica ナレッジベースを使用して、ハウツー記事、ベストプラクティス、よくある質問に対する回答など、製品リソースを見つけることができます。

ナレッジベースを検索するには、https://search.informatica.com にアクセスしてください。ナレッジベースに関する質問、コメント、ご意見の連絡先は、Informatica ナレッジベースチーム（KB_Feedback@informatica.com）です。

Informatica マニュアル

Informatica マニュアルポータルでは、最新および最近の製品リリースに関するドキュメントの膨大なライブラリを参照できます。マニュアルポータルを利用するには、https://docs.informatica.com にアクセスしてください。

製品マニュアルに関する質問、コメント、ご意見については、Informatica マニュアルチーム（infa_documentation@informatica.com）までご連絡ください。

Informatica 製品可用性マトリックス

製品可用性マトリックス（PAM）には、製品リリースでサポートされるオペレーティングシステム、データベースなどのデータソースおよびターゲットが示されています。Informatica PAM は、https://network.informatica.com/community/informatica-network/product-availability-matrices で参照できます。

Informatica Velocity

Informatica Velocity は、Informatica プロフェッショナルサービスが開発したヒントとベストプラクティスのコレクションで、多数のデータ管理プロジェクトから得た実体験に基づいています。Informatica Velocity には、世界中の組織と連携してデータ管理ソリューションを計画、開発、デプロイ、管理する Informatica コンサルタントによる集合知を表しています。

Informatica Velocity リソースには、http://velocity.informatica.com からアクセスしてください。Informatica Velocity についての質問、コメント、またはアイデアがある場合は、ips@informatica.com から Informatica プロフェッショナルサービスにお問い合わせください。

Informatica Marketplace

Informatica Marketplace は、お使いの Informatica 製品を拡張したり強化したりするソリューションを検索できるフォーラムです。Marketplace で、Informatica デベロッパーやパートナーからの多数のソリューションを活用すれば、生産性を向上したり、プロジェクトでの実装時間を短縮したりできます。Informatica Marketplace は、https://marketplace.informatica.com からアクセスしてください。

Informatica グローバルカスタマサポート

電話または Informatica Network を介してグローバルカスタマサポートに連絡できます。

各地域の Informatica グローバルカスタマサポートの電話番号は、Informatica Web サイト（https://www.informatica.com/services-and-training/customer-success-services/contact-us.html）を参照してください。

Informatica Network のオンラインサポートリソースを見つけるには、https://network.informatica.com にアクセスして eSupport オプションを選択します。

第 1 章

ビジネスエンティティサービスについて

この章では、以下の項目について説明します。

ビジネスエンティティサービスの概要

ビジネスエンティティサービスは、MDM Hub コードを実行して、ビジネスエンティティのベースオブジェクトレコードを作成、更新、削除、検索する一連の操作です。Java コードまたは JavaScript コードを実行してビジネスエンティティサービス呼び出しを実行するカスタムユーザーインタフェースを作成できます。

例えば、Dun and Bradstreet 社のデータを使用してサプライヤレコードを拡張するビジネスエンティティサービスを作成できます。サプライヤレコードを入力として取得するビジネスエンティティサービスを設定し、Dun and Bradstreet 社からいくつかの情報を取得し、レコードを更新し、更新されたサプライヤレコードを出力します。

ビジネスエンティティのベースオブジェクトには、次のビジネスエンティティサービスがあります。

読み取り

各ビジネスエンティティには、読み取り操作を実行するためのビジネスエンティティサービスがあります。

書き込み

各ビジネスエンティティには、書き込み操作を実行するためのビジネスエンティティサービスがあります。

検索

検索可能フィールドがあるすべてのビジネスエンティティには、検索操作を実行するためのビジネスエンティティサービスがあります。

例えば、Person ビジネスエンティティには検索可能フィールドがあります。MDM Hub は、ReadPerson、WritePerson、および SearchPerson の各ビジネスエンティティサービスを生成します。読み取り、書き込み、および検索の各ビジネスエンティティサービス手順では、ビジネスエンティティ内のレコードの読み取り、作成、更新、削除、および検索を行うことができます。

ビジネスエンティティサービス

ビジネスエンティティサービスは操作を実行します。ReadBE、WriteBE、および SearchBE ビジネスエンティティサービスを使用できます。

ビジネスエンティティサービスにはサービス手順があります。入力要求は各サービス手順を通過します。1 つの手順の出力は、次の手順の入力です。1 つの手順の出力は、次の手順の入力に情報を渡すことができます。すべてのビジネスエンティティサービス手順は、単一トランザクション内の 1 つの Enterprise JavaBeans（EJB）呼び出しとして実行されます。MDM Hub は例外を処理します。

注: ビジネスエンティティサービスを使用する前に、オペレーショナル参照ストアを検証します。

ReadBE ビジネスエンティティサービス

ReadBE ビジネスエンティティサービスは、ビジネスエンティティのベースオブジェクトレコードからデータを読み取ります。

ReadBE 手順でパジネーションパラメータを指定することにより、返すレコードの数と表示する結果のページを設定できます。

ReadBE サービスの結果には論理削除が行われたレコードは含まれません。

ビジネスエンティティサービス要求で EffectiveDate パラメータを渡さないと、MDM Hub は有効日が NULL だと見なし、ビジネスエンティティサービスはベースオブジェクトからデータを読み取ります。EffectiveDate パラメータを渡すと、MDM Hub は相互参照レコードからベストバージョンオブトゥルースを計算し、読み取りビジネスエンティティサービスは最新の最善データを返します。

WriteBE ビジネスエンティティサービス

WriteBE ビジネスエンティティサービスは、ビジネスエンティティ要素のデータの更新、子ビジネスエンティティ要素の作成、または子ビジネスエンティティ要素の削除を行うことができます。

注: WriteBE ビジネスエンティティサービスは、既存の信頼設定を使用して、ベースオブジェクトの信頼を計算します。このサービスで信頼オーバーライドを実行することはできません。

オプションのパラメータ

次の表では、WriteBE ビジネスエンティティサービスで使用可能なオプションパラメータについて説明します。

パラメータ	説明
recordState	レコードの状態を ACTIVE、PENDING、または DELETED に設定します。 **注**: `recordState=ACTIVE` を設定し、論理的に削除されたレコードでサービスを実行すると、サービスはレコードをアクティブな状態に復元します。
EffectivePeriod	有効期間を指定します。`EffectivePeriod` パラメータを渡さないと、MDM Hub は期間に制限がないと見なします。 MDM Hub は、ルートオブジェクトと子オブジェクトの有効期間が整合しているかをチェックしません。レコードを作成または更新するときには、親レコードと子レコードの有効期間を確実に整合させる必要があります。

SearchBE ビジネスエンティティサービス

ビジネスエンティティのルートレコードを検索するには、SearchBE ビジネスエンティティサービスを使用します。

検索のためにビジネスエンティティを設定する方法については、*Multidomain MDM の設定ガイド*を参照してください。

ビジネスエンティティサービスエンドポイント

ビジネスエンティティサービスには、Enterprise JavaBeans（EJB）エンドポイント、Representational State Transfer（REST）エンドポイント、または Simple Object Access Protocol（SOAP）エンドポイントを介してアクセスできます。

REST エンドポイントは EJB スタンドポイント上に作成されます。REST ビジネスエンティティサービス設定では、REST URL が EJB ビジネスエンティティサービス呼び出しにどのようにマッピングされるかを定義します。

ビジネスエンティティサービスの EJB エンドポイント

Enterprise JavaBeans（EJB）エンドポイントは、すべてのタイプのビジネスエンティティサービス呼び出しの基本エンドポイントです。その他のすべてのエンドポイントは、EJB エンドポイントにマッピングされます。

ビジネスエンティティサービスは、ステートレス EJB として公開されます。ステートレス EJB コンテナは、ドメイン内のさまざまなサーバー間で負荷を分散するために、インスタンスのプーリング、インスタンスの割り当て、および負荷分散の適用を行うことができます。

EJB エンドポイントは、認証用のユーザー名とパスワードを受け入れます。

ビジネスエンティティサービスの REST エンドポイント

Representational State Transfer（表現状態転送: REST）エンドポイント呼び出しを行うと、ビジネスエンティティサービスが Web サービスとして使用できるようになります。

MDM ハブは Swagger を使用して、URL およびパラメータとともにすべての REST Web サービスを一覧表示します。

Swagger には次の URL でアクセスできます。

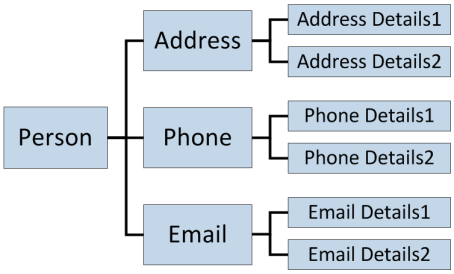
`http://<host: ホスト>:<port: ポート>/cmx/swagger-ui.html`

REST および EJB のビジネスエンティティサービス呼び出し

ビジネスエンティティサービス呼び出しを行うときに、ビジネスエンティティ全体を要求するのではなく特定の子ブランチを指定することがあります。

例えば、Person ルートノードと複数の子ブランチがあるビジネスエンティティで読み取り操作を行うとします。Person ベースオブジェクトには、Address、Phone、および Email 子ベースオブジェクトがあります。各子ベースオブジェクトには 2 つの孫ベースオブジェクトがあります。

次の図は、ブランチが複数存在するビジネスエンティティの構造を示しています。



1 つの要求でさまざまな深さの複数の子ブランチから読み取ることができます。例えば、1 つの要求で Person、Phone、Phone Details1、Phone Details2、Email、Email Details2 を読み取ることができます。

次の URL サンプルは、REST 読み取り要求を実行して、Address Details 1 および Email 子レコードに加えて、行 ID 1242 の Person レコードを取得する方法を示しています。

```
http://localhost:8080/cmx/cs/localhost-ORCL-DS_UI1/Person/1242?children=Address/Address_Details_1,Email
```

ビジネスエンティティサービスの SOAP エンドポイント

Simple Object Access Protocol（SOAP）エンドポイント呼び出しを行うと、ビジネスエンティティサービスが Web サービスとして使用できるようになります。

Web サービス記述言語（WSDL）ファイルには、Web サービス、SOAP 要求および応答の形式、およびすべてのパラメータの XML 記述が含まれています。MDM Hub は、オペレーショナル参照ストアごとに WSDL ファイルを生成します。

ルートレコードの識別

ルートレコードを識別するには、次のいずれかの方法を使用できます。

- 行 ID。レコードの ROWID_OBJECT 列の値。
- systemName と sourceKey。systemName はレコードが属するシステムの名前です。sourceKey は、レコードの PKEY_SRC_OBJECT 列の値です。
- オブジェクトのグローバル識別子（GBID）。GBID には複合値を指定でき、この場合すべての値を渡す必要があります。

  注: GBID のアプローチは、ReadBE サービスでのみ機能します。

次のサンプルコードでは、systemName と sourceKey を使用して、レコードを識別します。

```
String systemName = "SFA";

Properties config = new Properties();
config.put(SiperianClient.SIPERIANCLIENT_PROTOCOL, EjbSiperianClient.PROTOCOL_NAME);
CompositeServiceClient client = CompositeServiceClient.newCompositeServiceClient(config);
CallContext callContext = new CallContext(orsId, user, pass);
helperContext = client.getHelperContext(callContext);
DataFactory dataFactory = helperContext.getDataFactory();

//String personRowId = "1097";
String pkeySrcObject = "CST1379";

//Set custom key pkey
pkey = (Key) dataFactory.create(Key.class);
pkey.setSystemName(systemName);
```

```
pkey.setSourceKey(val);
writePerson.setKey(pkey);
```

セキュリティおよびデータフィルタ

ベースオブジェクトおよびリソースにユーザーロール特権がある場合、ビジネスエンティティはそれらの特権を継承します。ビジネスエンティティレコードにアクセスするには、ユーザーロールはビジネスエンティティのルートベースオブジェクトおよびその他のリソースに対する適切な特権を持っている必要があります。

ビジネスエンティティサービスは、ビジネスエンティティフィールドに設定したデータフィルタも継承します。

セキュリティおよびデータフィルタの詳細については、『*Multidomain MDM のプロビジョニングツールガイド*』を参照してください。

証明書ベースの認証

MDM Hub は MDM Hub コンポーネントと信頼されたアプリケーションの間の認証を保護するために、証明書ベースの認証メカニズムを使用します。この認証メカニズムは、ビジネスエンティティサービス API に対してもサポートされます。

外部クライアントアプリケーションは MDM Hub へのビジネスエンティティサービス要求を作成できます。外部クライアントアプリケーションユーザーを信頼できるアプリケーションユーザーとして MDM Hub に登録する必要があります。外部クライアントアプリケーションに関連付けられたユーザーの公開証明書を登録する必要もあります。

登録後、外部クライアントユーザーはプライベートキーを使用して、暗号化された認証要求を MDM Hub に送信できます。MDM Hub はプライベートキーを使用して認証要求を復号化し、プレーンテキストで要求されたデータに応答します。

ビジネスエンティティサービスペイロードのユーザー認証要求は、プライベートキーを使用して暗号化され、パブリックキーを使用して復号化されます。

注: パブリックキーおよび証明書を MDM Hub に登録するには、Informatica グローバルカスタマサポートにお問い合わせください。

信頼できるアプリケーションおよび証明書ベースの認証の構成方法の詳細については、『Informatica Multidomain MDM 10.4 セキュリティガイド』の証明書ベースの認証の章を参照してください。

第 2 章

EJB ビジネスエンティティサービス呼び出し

この章では、以下の項目について説明します。

EJB ビジネスエンティティサービス呼び出しの概要

Enterprise JavaBeans（EJB）ビジネスエンティティサービス呼び出しにより、ビジネスエンティティ内のベースオブジェクトレコードの作成、更新、削除、および検索を行うことができます。EJB ビジネスエンティティサービス呼び出しを実行する Java コードを作成できます。

標準 Service Data Objects（SDO）のクラスに基づいて Java コードを作成するか、または MDM Hub がビジネスエンティティおよびビジネスエンティティサービスの設定に基づいて生成する Java クラスに基づいて Java コードを作成することができます。

標準 SDO クラスを使用した Java コード例

次のサンプルは、標準の Service Data Objects（SDO）クラスに基づいて Enterprise JavaBeans（EJB）呼び出しを行う Java コードを示しています。

このサンプルは、Resource Kit の C:\<MDM Hub installation directory: MDM Hub のインストールディレクトリ>\hub\resourcekit\samples\COS\source\java\com\informatica\mdm\sample\cs\DynamicSDO.java というファイルにあります。

次の Java コードは標準の SDO クラスに基づいており、EJB ビジネスエンティティサービス呼び出しを実行して、Person ベースオブジェクトレコードの作成、複数の子レコードの追加、1 つの子レコードの削除、および Person レコードとすべての子レコードの削除を行います。

```
package com.informatica.mdm.sample.cs;

import com.informatica.mdm.cs.CallContext;
import com.informatica.mdm.cs.api.CompositeServiceException;
import com.informatica.mdm.cs.client.CompositeServiceClient;
import com.siperian.sif.client.EjbSiperianClient;
import com.siperian.sif.client.SiperianClient;
import commonj.sdo.DataObject;
```

```
import commonj.sdo.Property;
import commonj.sdo.Type;
import commonj.sdo.helper.DataFactory;
import commonj.sdo.helper.HelperContext;

import java.io.PrintStream;
import java.util.Arrays;
import java.util.Properties;

public class DynamicSDO {

    public static void main(String[] args) throws CompositeServiceException {

        if(args.length != 3) {
            System.err.println("USAGE: DynamicSDO <ors> <user> <pass>");
            return;
        }

        new DynamicSDO(args[0], args[1], args[2]).execute();
    }


    private String orsId;
    private String user;
    private String pass;
    private HelperContext helperContext;
    private PrintStream out = System.out;

    public DynamicSDO(String orsId, String user, String pass) {
        this.orsId = orsId;
        this.user = user;
        this.pass = pass;
    }

    public void execute() throws CompositeServiceException {

        String systemName = "Admin";

        Properties config = new Properties();
        config.put(SiperianClient.SIPERIANCLIENT_PROTOCOL, EjbSiperianClient.PROTOCOL_NAME);
        CompositeServiceClient client = CompositeServiceClient.newCompositeServiceClient(config);

        CallContext callContext = new CallContext(orsId, user, pass);

        helperContext = client.getHelperContext(callContext);

        DataFactory dataFactory = helperContext.getDataFactory();

        // types for Read requests
        Type coFilterType = helperContext.getTypeHelper().getType("urn:cs-base.informatica.mdm", "CoFilter");
        Type coFilterNodeType = helperContext.getTypeHelper().getType("urn:cs-base.informatica.mdm",
"CoFilterNode");
        Type keyType = helperContext.getTypeHelper().getType("urn:cs-base.informatica.mdm", "Key");

        // ReadCO & WriteCO request types
        Type readPersonType = helperContext.getTypeHelper().getType("urn:cs-ors.informatica.mdm",
"ReadPerson");
        Type writePersonType = helperContext.getTypeHelper().getType("urn:cs-ors.informatica.mdm",
"WritePerson");

        // 1. Create new person
        DataObject createPerson = dataFactory.create(writePersonType);
        DataObject createPersonParameters = createPerson.createDataObject("parameters");
        createPersonParameters.setString("systemName", systemName);
        DataObject person = createPerson.createDataObject("object");

        person.getChangeSummary().beginLogging();

        DataObject personRoot = person.createDataObject("Person");
        personRoot.setString("firstName", "John");
        personRoot.setString("lastName", "Smith");
```

```
person.getChangeSummary().endLogging();

dump("*** CREATE NEW PERSON ...", createPerson);

DataObject createPersonResponse = client.process(callContext, createPerson);

dump("*** PERSON CREATED:", createPersonResponse);

String personRowId = createPersonResponse.getString("object/Person/rowidObject");

DataObject readPerson = dataFactory.create(readPersonType);
DataObject readPersonParameters = readPerson.createDataObject("parameters");
DataObject coFilter = readPersonParameters.createDataObject("coFilter");
DataObject coFilterNode = coFilter.createDataObject("object");
coFilterNode.set("name", "Person");
DataObject key = coFilterNode.createDataObject("key");
key.set("rowid", personRowId);

dump("*** READ CREATED PERSON...", readPerson);

DataObject readPersonResponse = client.process(callContext, readPerson);

dump("*** READ RESULT:", readPersonResponse);

person = readPersonResponse.getDataObject("object");
person.detach();

person.getChangeSummary().beginLogging();

personRoot = person.getDataObject("Person");
// add new 'one' child
DataObject genderCd = personRoot.createDataObject("genderCd");
genderCd.setString("genderCode", "M");

// add two 'many' children
DataObject phonePager = personRoot.createDataObject("TelephoneNumbers");
Property item = phonePager.getInstanceProperty("item");
Type phoneType = item.getType();

DataObject phone1 = dataFactory.create(phoneType);
phone1.setString("phoneNumber", "111-11-11");
DataObject phone2 = dataFactory.create(phoneType);
phone2.setString("phoneNumber", "222-22-22");

phonePager.setList(item, Arrays.asList(phone1, phone2));

person.getChangeSummary().endLogging();

DataObject updatePerson = dataFactory.create(writePersonType);
updatePerson.setDataObject("object", person);
DataObject updatePersonParameters = updatePerson.createDataObject("parameters");
updatePersonParameters.setString("systemName", systemName);
updatePersonParameters.setString("interactionId", "");

dump("*** UPDATE PERSON...", updatePerson);

DataObject updatePersonResponse = client.process(callContext, updatePerson);

dump("*** PERSON UPDATED:", updatePersonResponse);

coFilterNode.set("depth", 3);

readPersonParameters.setBoolean("readSystemFields", true);

dump("*** READ UPDATED PERSON WITH CHILDREN...", readPerson);

readPersonResponse = client.process(callContext, readPerson);

dump("*** READ RESULT:", readPersonResponse);
```

```
        person = readPersonResponse.getDataObject("object");
        person.detach();

        person.getChangeSummary().beginLogging();

        genderCd = person.getDataObject("Person").createDataObject("genderCd");
        genderCd.setString("genderCode", "F");

        // delete one phone
        DataObject phoneItem = person.getDataObject("Person/TelephoneNumbers/item[1]");
        phoneItem.delete();

        person.getChangeSummary().endLogging();

        DataObject deletePhone = dataFactory.create(writePersonType);
        deletePhone.setDataObject("object", person);
        DataObject deletePhoneParameters = deletePhone.createDataObject("parameters");
        deletePhoneParameters.setString("systemName", systemName);

        dump("*** DELETE CHILD...", deletePhone);

        DataObject deletePhoneResponse = client.process(callContext, deletePhone);

        dump("*** CHILD DELETED:", deletePhoneResponse);

        readPersonParameters.setBoolean("readSystemFields", false);

        dump("*** READ PERSON AFTER CHILD WAS DELETEED...", readPerson);

        readPersonResponse = client.process(callContext, readPerson);

        dump("*** READ RESULT:", readPersonResponse);

        person = readPersonResponse.getDataObject("object");
        person.detach();

        person.getChangeSummary().beginLogging();

        person.getDataObject("Person").detach();

        person.getChangeSummary().endLogging();

        DataObject deletePerson = dataFactory.create(writePersonType);
        deletePerson.setDataObject("object", person);
        DataObject deletePersonParameters = deletePerson.createDataObject("parameters");
        deletePersonParameters.setString("systemName", systemName);

        dump("*** DELETE PERSON...", deletePerson);

        DataObject deletePersonResponse = client.process(callContext, deletePerson);

        dump("*** PERSON DELETED:", deletePersonResponse);

        dump("*** TRY TO READ PERSON AFTER DELETE", readPerson);

        try {
            readPersonResponse = client.process(callContext, readPerson);

            dump("*** READ RESULT:", readPersonResponse);
        } catch (CompositeServiceException e) {
            out.println("*** READ RESULT: " + e.getLocalizedMessage());
        }


    }

    private void dump(String title, DataObject dataObject) {
        String xml = helperContext.getXMLHelper().save(
                dataObject,
                dataObject.getType().getURI(),
                dataObject.getType().getName());
```

```
        out.println(title);
        out.println(xml);
        out.println();
    }

}
```

生成された SDO クラスを使用した Java コード例

この例は、MDM Hub がビジネスエンティティおよびビジネスエンティティサービスの設定に基づいて生成する Java クラスに基づいて Enterprise JavaBeans（EJB）呼び出しを実行する Java コードを示しています。

このサンプルは、Resource Kit の C:\<MDM Hub installation directory: MDM Hub のインストールディレクトリ>\hub\resourcekit\samples\COS\source\java\com\informatica\mdm\sample\cs\GeneratedSDO.java というファイルにあります。

次の Java コードは生成されたクラスに基づいており、EJB ビジネスエンティティサービス呼び出しを実行して、Person ベースオブジェクトレコードの作成、複数の子レコードの追加、1 つの子レコードの削除、および Person レコードとすべての子レコードの削除を行います。

```
package com.informatica.mdm.sample.cs;

import com.informatica.mdm.cs.CallContext;
import com.informatica.mdm.cs.api.CompositeServiceException;
import com.informatica.mdm.cs.client.CompositeServiceClient;
import com.informatica.mdm.sdo.cs.base.CoFilter;
import com.informatica.mdm.sdo.cs.base.CoFilterNode;
import com.informatica.mdm.sdo.cs.base.Key;
import com.siperian.sif.client.EjbSiperianClient;
import com.siperian.sif.client.SiperianClient;
import commonj.sdo.DataObject;
import commonj.sdo.helper.DataFactory;
import commonj.sdo.helper.HelperContext;
import mdm.informatica.co_ors.*;
import mdm.informatica.cs_ors.*;

import java.io.PrintStream;
import java.util.Arrays;
import java.util.Properties;

public class GeneratedSDO {

    public static void main(String[] args) throws CompositeServiceException {

        if(args.length != 3) {
            System.err.println("USAGE: GeneratedSDO <ors> <user> <pass>");
            return;
        }

        new GeneratedSDO(args[0], args[1], args[2]).execute();
    }


    private String orsId;
    private String user;
    private String pass;
    private HelperContext helperContext;
    private PrintStream out = System.out;

    public GeneratedSDO(String orsId, String user, String pass) {
        this.orsId = orsId;
        this.user = user;
        this.pass = pass;
    }
```

```
    public void execute() throws CompositeServiceException {

        String systemName = "Admin";

        Properties config = new Properties();
        config.put(SiperianClient.SIPERIANCLIENT_PROTOCOL, EjbSiperianClient.PROTOCOL_NAME);
        CompositeServiceClient client = CompositeServiceClient.newCompositeServiceClient(config);

        CallContext callContext = new CallContext(orsId, user, pass);

        helperContext = client.getHelperContext(callContext);

        DataFactory dataFactory = helperContext.getDataFactory();

        // 1. Create new person
        WritePerson createPerson = (WritePerson)dataFactory.create(WritePerson.class);
        WritePersonParameters createPersonParameters =
(WritePersonParameters)dataFactory.create(WritePersonParameters.class);
        createPersonParameters.setSystemName(systemName);
        createPerson.setParameters(createPersonParameters);

        Person person = (Person)dataFactory.create(Person.class);
        createPerson.setObject(person);

        person.getChangeSummary().beginLogging();

        PersonRoot personRoot = (PersonRoot)dataFactory.create(PersonRoot.class);
        personRoot.setFirstName("John");
        personRoot.setLastName("Smith");
        person.setPerson(personRoot);

        person.getChangeSummary().endLogging();

        dump("*** CREATE NEW PERSON ...", createPerson);

        WritePersonReturn createPersonResponse = (WritePersonReturn)client.process(callContext,
(DataObject)createPerson);

        dump("*** PERSON CREATED:", createPersonResponse);

        String personRowId = createPersonResponse.getObject().getPerson().getRowidObject();

        Key key = (Key)dataFactory.create(Key.class);
        key.setRowid(personRowId);
        CoFilterNode coFilterNode = (CoFilterNode)dataFactory.create(CoFilterNode.class);
        coFilterNode.setName(Person.class.getSimpleName());
        coFilterNode.setKey(key);
        CoFilter coFilter = (CoFilter)dataFactory.create(CoFilter.class);
        coFilter.setObject(coFilterNode);
        ReadPersonParameters readPersonParameters =
(ReadPersonParameters)dataFactory.create(ReadPersonParameters.class);
        readPersonParameters.setCoFilter(coFilter);

        ReadPerson readPerson = (ReadPerson)dataFactory.create(ReadPerson.class);
        readPerson.setParameters(readPersonParameters);

        dump("*** READ CREATED PERSON...", readPerson);

        ReadPersonReturn readPersonResponse = (ReadPersonReturn)client.process(callContext,
(DataObject)readPerson);

        dump("*** READ RESULT:", readPersonResponse);

        person = readPersonResponse.getObject();
        ((DataObject)person).detach();

        person.getChangeSummary().beginLogging();

        personRoot = person.getPerson();
        // add new 'one' child
```

```
        LUGenderLookup genderCd = (LUGenderLookup)dataFactory.create(LUGenderLookup.class);
        genderCd.setGenderCode("M");
        personRoot.setGenderCd(genderCd);

        // add two 'many' children
        PersonTelephoneNumbersPager phonePager =
(PersonTelephoneNumbersPager)dataFactory.create(PersonTelephoneNumbersPager.class);

        PersonTelephoneNumbers phone1 =
(PersonTelephoneNumbers)dataFactory.create(PersonTelephoneNumbers.class);
        phone1.setPhoneNumber("111-11-11");
        PersonTelephoneNumbers phone2 =
(PersonTelephoneNumbers)dataFactory.create(PersonTelephoneNumbers.class);
        phone2.setPhoneNumber("222-22-22");

        phonePager.setItem(Arrays.asList(phone1, phone2));
        personRoot.setTelephoneNumbers(phonePager);

        person.getChangeSummary().endLogging();

        WritePerson updatePerson = (WritePerson)dataFactory.create(WritePerson.class);
        updatePerson.setObject(person);
        WritePersonParameters updatePersonParameters =
(WritePersonParameters)dataFactory.create(WritePersonParameters.class);
        updatePersonParameters.setSystemName(systemName);
        updatePersonParameters.setInteractionId("");
        updatePerson.setParameters(updatePersonParameters);

        dump("*** UPDATE PERSON...", updatePerson);

        WritePersonReturn updatePersonResponse = (WritePersonReturn)client.process(callContext,
(DataObject)updatePerson);

        dump("*** PERSON UPDATED:", updatePersonResponse);

        coFilterNode.setDepth(3);

        readPersonParameters.setReadSystemFields(true);

        dump("*** READ UPDATED PERSON WITH CHILDREN (with system fields)...", readPerson);

        readPersonResponse = (ReadPersonReturn)client.process(callContext, (DataObject)readPerson);

        dump("*** READ RESULT:", readPersonResponse);

        person = readPersonResponse.getObject();
        ((DataObject)person).detach();

        person.getChangeSummary().beginLogging();

        // delete one phone
        person.getPerson().getTelephoneNumbers().getItem().remove(0);

        // change gender
        genderCd = (LUGenderLookup)dataFactory.create(LUGenderLookup.class);
        genderCd.setGenderCode("F");
        personRoot.setGenderCd(genderCd);

        person.getChangeSummary().endLogging();

        WritePerson deletePhone = (WritePerson)dataFactory.create(WritePerson.class);
        deletePhone.setObject(person);
        WritePersonParameters deletePhoneParameters =
(WritePersonParameters)dataFactory.create(WritePersonParameters.class);
        deletePhoneParameters.setSystemName(systemName);
        deletePhone.setParameters(deletePhoneParameters);

        dump("*** DELETE CHILD...", deletePhone);

        WritePersonReturn deletePhoneResponse = (WritePersonReturn)client.process(callContext,
(DataObject)deletePhone);
```

```
        dump("*** CHILD DELETED:", deletePhoneResponse);

        readPersonParameters.setReadSystemFields(false);

        dump("*** READ PERSON AFTER CHILD WAS DELETEED...", readPerson);

        readPersonResponse = (ReadPersonReturn)client.process(callContext, (DataObject)readPerson);

        dump("*** READ RESULT:", readPersonResponse);

        person = readPersonResponse.getObject();
        ((DataObject)person).detach();

        person.getChangeSummary().beginLogging();

        ((DataObject)person.getPerson()).delete();

        person.getChangeSummary().endLogging();

        WritePerson deletePerson = (WritePerson)dataFactory.create(WritePerson.class);
        deletePerson.setObject(person);
        WritePersonParameters deletePersonParameters =
(WritePersonParameters)dataFactory.create(WritePersonParameters.class);
        deletePersonParameters.setSystemName(systemName);
        deletePerson.setParameters(deletePersonParameters);

        dump("*** DELETE PERSON...", deletePerson);

        WritePersonReturn deletePersonResponse = (WritePersonReturn)client.process(callContext,
(DataObject)deletePerson);

        dump("*** PERSON DELETED:", deletePersonResponse);

        dump("*** TRY TO READ PERSON AFTER DELETE", readPerson);

        try {
            readPersonResponse = (ReadPersonReturn)client.process(callContext, (DataObject)readPerson);

            dump("*** READ RESULT:", readPersonResponse);
        } catch (CompositeServiceException e) {
            out.println("*** READ RESULT: " + e.getLocalizedMessage());
        }

    }

    private void dump(String title, Object object) {
        DataObject dataObject = (DataObject)object;
        String xml = helperContext.getXMLHelper().save(
                dataObject,
                dataObject.getType().getURI(),
                dataObject.getType().getName());
        out.println(title);
        out.println(xml);
        out.println();
    }
}
```

第 3 章

REST ビジネスエンティティサービス呼び出し

この章では、以下の項目について説明します。

ビジネスエンティティサービスの REST API の概要

Representational State Transfer（REST）エンドポイント呼び出しを行うと、ビジネスエンティティサービスが Web サービスとして使用できるようになります。

REST 呼び出しは、ビジネスエンティティ内にベースオブジェクトレコードや関連する子レコードを作成したり、ビジネスエンティティ内のレコードを更新、削除、検索したりする場合に実行できます。レコードのマージ、マージ解除、照合などの操作も実行できます。REST 呼び出しは、タスクの作成、更新、検索、実行に利用できます。また、REST 呼び出しは、タスクまたはレコードの添付ファイルなどのファイルの作成、更新、削除にも利用できます。REST API を使用してビジネスエンティティサービスを呼び出す前に、オペレーショナル参照ストアを検証します。

REST ビジネスエンティティサービス呼び出しは、URL（Uniform Resource Locator）形式での Web サービス要求です。MDM Hub は、ビジネスエンティティ内の各ベースオブジェクトに一意の URL を割り当てます。この一意の URL を使用し、どのベースオブジェクトを作成、更新または削除するのかを特定できます。

ほとんどの REST API 呼び出しの URL で、ビジネスエンティティまたはビジネスエンティティビューのいずれかを指定できます。例えば、レコードの作成 REST API 呼び出しでは、次の形式を使用できます。

```
http://<host>:<port>/<context>/<databaseID>/<business entity>?systemName=<name of the source system>
```

```
http://<host>:<port>/<context>/<databaseID>/<business entity view name>?systemName=<name of the source system>
```

一部の REST API は、ビジネスエンティティビューをサポートしていません。例えば、マージのプレビュー、保留中のマージ、マージの昇格などのマージタスクに関連する REST API は、ビジネスエンティティビューをサポートしていません。

注: REST API を使用してビジネスエンティティサービスを呼び出す前に、オペレーショナル参照ストアを検証します。

サポートされる REST メソッド

ビジネスエンティティサービスの REST API は、標準の HTTP メソッドを使用して、レコード、タスク、ファイルなどのリソースの操作を実行します。

ビジネスエンティティサービスの REST API は、次の HTTP 要求メソッドをサポートします。

メソッド	説明
GET	レコード、タスク、またはファイルに関する情報を取得します。
POST	タスク、ルートレコード、子レコード、またはファイルを作成します。 **注**: POST 要求のオペレーショナル参照ストア（ORS）名では大文字と小文字が区別されるが、ORS 名の文字種が MDM Hub の名前と一致しない場合、エラーが発生する。
PUT	ルートレコード、子レコード、タスク、またはファイルを更新します。
PATCH	タスクを部分的に更新します。
DELETE	ルートレコード、子レコード、またはファイルを削除します。

認証方法

ビジネスエンティティサービスの REST エンドポイントは、ユーザーの認証に HTTP の基本認証を使用します。ブラウザを使用して初めてビジネスエンティティサービスに接続するときに、MDM Hub のユーザー名とパスワードを指定する必要があります。認証に成功すると、ビジネスエンティティサービス REST API を使用して操作を実行できます。

ブラウザはユーザー資格情報をキャッシュし、ビジネスエンティティサービスに対するその後のすべての要求にそれらを使用します。

サードパーティアプリケーションからログインする場合の認証クッキー

認証 Cookie を使用して MDM Hub ユーザーを認証し、Postman などのサードパーティアプリケーションからビジネスエンティティサービスを呼び出します。認証 Cookie を使用すると、ユーザー名とパスワードをハードコーディングする必要がありません。

認証 Cookie を使用するプロセスには、次の手順が含まれます。

1. 認証されたユーザーの資格情報を使用して、認証 Cookie を取得します。
2. 認証 Cookie を保存します。
3. 認証 Cookie を使用して REST API を呼び出します。

すべての MDM セッションは、Informatica CSRF トークン（ICT）を使用します。ICT は HTTP ヘッダーに表示されます。

ユーザーがセッション ID を要求すると、応答は HTTP ヘッダーの一部として ICT を返します。外部アプリケーションの場合、外部アプリケーションが使用できる認証用の URL があります。この URL は、ヘッダーにある ICT と認証 Cookie を返します。認証 Cookie は MDM セッション ID です。

以降のビジネスエンティティサービスの呼び出しでは、呼び出しに認証 Cookie と ICT が含まれている必要があります。後続の要求では、セッションが期限切れになるまで ICT および MDM セッション ID を使用できます。

認証 Cookie と ICT を取得するための要求を送信する場合、userinfo セクションはオプションです。

セッションを管理するには、REST 呼び出しを使用します。

認証 Cookie と Informatica CSRF トークンの取得

次の例は、認証 Cookie と Informatica CSRF トークン（ICT）を取得する方法を示しています。

```
POST: https://<host>:<port>/cmx/auth/<database ID>

Authorization:   No Auth

Body:
{
    "username" : "admin",
    "password" : {
        "encrypted" : false,
        "password" : "admin"
    },
    "userInfo" : {
        "sessionTimeout":300,
        "role":"Manager"
    }
}
```

次の例は、ICT がヘッダーにどのように表示されるかを示しています。

```
ICT:  0ffa3b95c67e111b9c14c140a70273a15db266993d0c763d555135de4c4d6110
```

次の例は、認証 Cookie が応答にどのように表示されるかを示しています。

```
Name: mdmsessionid
Value: vppOTOhJLKMjYJf7Diil
Domain: 10.xx.xx.xx
Path: /cmx
Expires: Session
HttpOnly: true
Secure: false
```

Person レコードの作成

次の例は、認証 Cookie と Informatica CSRF トークン（ICT）を使用して Person レコードを作成する方法を示しています。

```
POST: https://<host:port>/cmx/cs/<Database ID>/Person?systemName=Admin

Authorization: No Auth (or Inherit auth from the parent)

Headers:
Cookie:  mdmsessionid: vppOTOhJLKMjYJf7Diil
ICT:  0ffa3b95c67e111b9c14c140a70273a15db266993d0c763d555135de4c4d6110

Body:

{
    "firstName": "John",
    "lastName": "Smith"
}
```

この例の ICT は、前のサンプルからのものです。

次のサンプルは、要求で認証 Cookie と ICT を渡さなかった場合に応答に表示されるエラーを示しています。

```
{
    "errorCode": "SIP-50201",
    "errorMessage": "SIP-50201: The MDM Hub did not process the business entity service request. The request
did not contain credentials. Specify the appropriate credentials in the business entity service request."
}
```

セッションタイムアウトの設定

認証 Cookie と Informatica CSRF トークン（ICT）を取得するための要求を送信する場合、userinfo セクションはオプションです。セッションタイムアウトを設定するには、要求の userinfo セクションで、セッションタイムアウトパラメータを指定します。パラメータの値は秒単位です。

次のサンプルコードは、ユーザー情報セクションとセッションタイムアウトパラメータを示しています。

```
"userInfo" : {
        "sessionTimeout":300,
        "role":"Manager"
    }
```

セッションログイン要求でのユーザー情報の取得

認証 Cookie と Informatica CSRF トークン（ICT）を要求したユーザーの詳細を取得できます。

次の要求例は、セッションログイン要求からユーザー情報を取得する方法を示しています。

```
GET: https://<host:port>/cmx/session/info

Headers:

Cookie: mdmsessionid=zjl4PDN8MEqiQWqAXwxX
ICT: 138d0a12d038323d7a436c73633700011846b97b0bb51006507dd0c435a983ce9
```

次のサンプル応答出力は、ロールとセッションのタイムアウト情報を示しています。

```
{
          "role": "Manager",
          "sessionTimeout": "300"
        }
```

セッションの更新

更新要求ごとに、セッションの有効期間を 30 分延長できます。セッションログイン要求でセッションタイムアウト間隔を指定できます。

次の要求例は、セッションを延長または更新する方法を示しています。

```
GET: https://<host:port>/cmx/session/refresh

Headers:

Cookie: mdmsessionid=zjl4PDN8MEqiQWqAXwxX
```

セッションの検証

セッションがアクティブで有効かどうかを検証できます。

次の要求例は、セッションを検証する方法を示しています。

```
GET: https://<host:port>/cmx/session/validate

Headers:

Cookie: mdmsessionid=zjl4PDN8MEqiQWqAXwxX
```

HTTP ステータスが OK の場合は、セッションがアクティブで有効であることを示します。

セッションからのログアウト

セッションからログアウトすると、セッションに関連する認証 Cookie と Informatica CSRF トークン（ICT）の有効期限が切れます。要求で使用すると、応答にエラーが表示されます。

次の要求例は、セッションからログアウトする方法を示しています。

```
GET: https://<host:port>/cmx/session/logout

Headers:

Cookie: mdmsessionid=zjl4PDN8MEqiQWqAXwxX
ICT: 138d0a12d038323d7a436c7363370011846b97b0bb51006507dd0c435a983ce9
```

HTTP ステータスが OK の場合は、セッションからのログアウトが成功したことを示します。

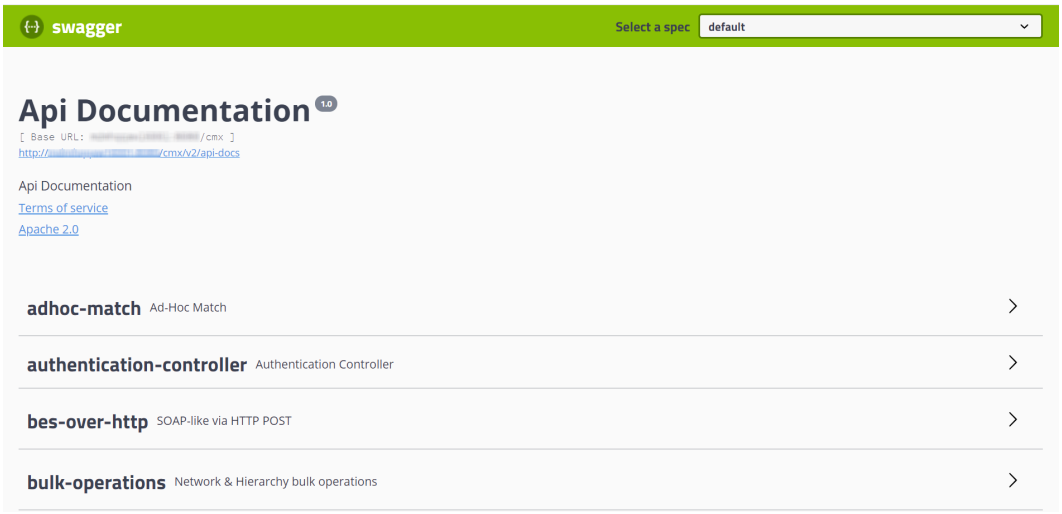
Swagger

MDM ハブは Swagger を使用して、URL およびパラメータとともにすべての REST Web サービスを一覧表示します。

Swagger には次の URL でアクセスできます。

```
http:://<host: ホスト>:<port: ポート>/cmx/swagger-ui.html
```

次の図は、Swagger のユーザーインタフェースを示しています。



REST URL

REST URL を使用し、ビジネスエンティティサービスの REST 呼び出しを行うことができます。

REST URL の構文は次のとおりです。

http://<host: ホスト>:<port: ポート>/<context: コンテキスト>/<database ID: データベース ID>/<path: パス>

この URL には以下のフィールドがあります。

host

データベースを実行しているホスト。

ポート

データベースリスナが使用するポート番号。

context

ビジネスエンティティ、検索、クエリ、一致、タスク、および階層の各 API のコンテキストは、cmx/cs です。

一致カラム API のコンテキストは cmx です。

ファイル API のコンテキストは cmx/file です。

タスク API のコンテキストは cmx/task です。

一括タスク管理 API のコンテキストは cmx/task/operations です。

一括リレーション API のコンテキストは cmx/bulk です。

チャートレポート API のコンテキストは cmx/report です。

注: ホストされた MDM 環境では、コンテキストにテナント名を含めます。例えば、コンテキストは、<テナント名>/cmx/cs や<テナント名>/cmx/file のようになります。

データベース ID

Hub コンソールのデータベースツールで登録された ORS の ID。

パス

API を使用するオブジェクト（レコード、タスク、ファイルなど）。

ルートレコードの URL の場合、パスはルートオブジェクト名の後に一意の識別子が続いたものになります。

Person ルートレコードの場合、例えば Person/798243.json のようになります。

ルートオブジェクトの直接の子であるレコードの URL の場合、パスには子レコード名と一意の識別子も含まれます。

Person ルートレコードの子である請求先住所レコードのパスの場合、例えば次のようになります。

Person/798243/BillAddresses/121522.json

2 以上の深さにある子レコードの URL の場合、パスにはその深さも含まれます。

次に、深さが 2 である子レコードの REST URL の例を示します。

http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/798243/BillAddresses/121522.json?depth=2

注: パラメータは大文字/小文字の区別が必要です。REST URL 内のパラメータ名の文字ケース（大文字/小文字の区別）を REST 設定内のパラメータ名の文字ケースと一致させてください。

ヘッダーと本文の設定

REST 操作は、HTTP メソッドとリソースへの完全 URL を組み合わせます。完全な要求の場合、REST 操作と適切な HTTP ヘッダおよび必要なデータを組み合わせます。REST 要求にはヘッダーと本文コンポーネントがあります。JSON 形式または XML 形式を使用して要求を定義できます。

要求ヘッダー

操作パラメータ、または REST 操作のメタデータを定義するには、要求ヘッダーを使用します。このヘッダーは、一連のフィールドと値のペアから構成されます。API 要求行にはメソッドと URL が含まれます。要求行の後にヘッダーフィールドを指定します。

REST API 要求ヘッダーを作成するには、次の例に示すように、<METHOD> <<host>:<port>/<context>/<database ID>/<Path>要求行の後にヘッダーフィールドを追加します。

```
<METHOD> <<host>:<port>/<context>/<database ID>/<Path>
Content-Type: application/<json/xml>
Accept: application/<json/xml>
```

次の表に、よく使用される要求ヘッダーフィールドをいくつか示します。

要求コンポーネント	説明
Content-Type	要求内のデータのメディアタイプ。REST 要求に本文を含める場合は、その本文のメディアタイプを Content-Type ヘッダーフィールドに指定する必要があります。PUT 要求と POST 要求には Content-Type ヘッダーフィールドを含めます。
Accept	応答内のデータのメディアタイプ。要求形式を指定するには、ヘッダー内に application/<json/xml>を使用するか、または URL に.json または.xml を使用します。デフォルトは XML です。

要求本文

要求でデータを送信するには、REST API 要求本文を使用します。要求本文は、本文をそれ自体にアタッチできる POST や PUT などのメソッドとともに使用します。データの本文は、ヘッダー行の後に記述します。要求メッセージに本文がある場合は、Content-Type ヘッダーフィールドを使用して要求ヘッダーに本文の形式を指定します。

XML スキーマ定義（XSD）ファイルは、使用できる要素と属性を説明したものです。要求本文のコンテンツは、XSD ファイルに定義する要素タイプによって異なります。

XSD ファイルは、以下の場所に格納されます。

```
http:://<host: ホスト>:<port: ポート>/cmx/csfiles
```

XML 形式

XML 要求形式を使用する場合は、要求オブジェクトをタグの包含セットとして定義します。

次の XML 形式を使用して要求オブジェクトを定義します。

```
<request object>
        <attribute1>value1</attribute1>
        <attribute2>value2</attribute2>
</request object>
```

次のサンプルは、要求オブジェクトを XML 形式で表現したものです。

```
<task>
    <taskType>
                <name>UpdateWithApprovalWorkflow</name>
    </taskType>
    <taskId>urn:b4p2:5149</taskId>
    <owner>manager</owner>
    <title>Smoke test task 222</title>
    <comments>Smoke testing</comments>
    <dueDate>2015-06-15T00:00:00</dueDate>
    <status>OPEN</status>
    <priority>NORMAL</priority>
    <creator>admin</creator>
    <createDate>2015-06-15T00:00:00</createDate>
    <updatedBy>admin</updatedBy>
    <lastUpdateDate>2015-06-15T00:00:00</lastUpdateDate>
    <businessEntity>Person</businessEntity>
    <orsId>localhost-orcl-DS_UI1</orsId>
    <processId>IDDUpdateWithApprovalTask</processId>
    <taskRecord>
                <businessEntity>
                        <name>Person</name>
                        <key>
                                    <rowid>123</rowid>
                                    <systemName></systemName>
                                    <sourceKey></sourceKey>
                        </key>
                </businessEntity>
    </taskRecord>
</task>
```

JSON 形式

JSON 要求形式を使用する場合は、type 属性を使用して要求オブジェクトを定義します。

次の JSON 形式を使用して要求オブジェクトを指定します。

```
{
 "type"="<request object>"
    "<attribute1>": "<value1>",
    "<attribute2>": "<value2>",
}
```

次のサンプルは、要求オブジェクトを JSON 形式で表現したものです。

```
{
     "type"="task"
     taskType: {name:"UpdateWithApprovalWorkflow"},
     taskId: "urn:b4p2:5149",
     owner: "manager",
     title: "Smoke test task 222",
     comments: "Smoke testing",
     dueDate: "2015-06-15T00:00:00",
     status: "OPEN",
     priority: "NORMAL",
     creator: "admin",
     createDate: "2015-06-15T00:00:00",
     updatedBy: "admin",
     lastUpdateDate: "2015-06-15T00:00:00",
     businessEntity: "Person",
     orsId: "localhost-orcl-DS_UI1",
     processId: 'IDDUpdateWithApprovalTask',
     taskRecord: [{
       businessEntity:{
         name: 'Person',
         key:{
            rowid: '123',
            systemName: '',
            sourceKey: ''
         }
       }
     }]
}
```

標準のクエリパラメータ

ビジネスエンティティサービスの REST API は、標準のクエリパラメータを使用して結果のフィルタリング、ページ区切り、および展開を行います。

疑問符（?）を 使用して、クエリパラメータを他のパラメータから切り離します。クエリパラメータは、等号で区切られた、キーと値のペアです。連続したクエリパラメータを区切るには、アンパサンド（&）を使用します。

次の REST 要求 URL は、クエリパラメータの使用方法を示しています。

```
/Person/123/Phone/SFA:456/PhoneUse?recordsToReturn=100&recordStates=ACTIVE,PENDING
```

次に、使用できる標準のクエリパラメータを示します。

パラメータ	説明
recordsToReturn	返す行の数を指定します。デフォルトは 10 です。
firstRecord	結果の最初の行を指定します。デフォルトは 1 です。さらにページを読み取るために後続の呼び出しで使用されます。
searchToken	前の要求で返された検索トークンを指定します。検索トークンを使用すると、検索結果の後続のページを取得できます。例えば、/Person/123/Phone というクエリでは、最初のページが表示されます。 /Person/123/Phone?searchToken=SVR1.AZAM5&firstRecord=10 というクエリでは、2 つ目のページが返ります。

パラメータ	説明
returnTotal	true に設定すると、結果内のレコードの数が返ります。デフォルトは false です。
depth	結果に含める子レベルの数を指定します。

UTC での日付と時刻の形式

要求および応答では、すべての日付と時刻は、特定のタイムゾーンのオフセットありまたはなしで UTC（協定世界時）で指定されます。

要求本文で日付と時刻を指定する際は、Date and Time Formats (NOTE-datetime)（ISO 8601）に定義されているいずれかの形式を使用します。

次のガイドラインは、NOTE-datetime ドキュメントからの抜粋です。

タイプ	構文	例
日付: 年	YYYY	1997
日付: 年と月	YYYY-MM	1997-07
日付: 年、月、日	YYYY-MM-DD	1997-07-16
日付、時間、分	YYYY-MM-DDThh:mmTZD	1997-07-16T19:20+01:00
日付と時間、分および秒	YYYY-MM-DDThh:mm:ssTZD	1997-07-16T19:20:30+01:00
日付と時間、分、秒および小数秒	YYYY-MM-DDThh:mm:ss.sTZD	1997-07-16T19:20:30.45+01:00

説明：

- YYYY = 4 桁の年
- MM = 2 桁の月（01 から 12）
- DD = 2 桁の月の日付（01 から 31）
- T = 日付の後に時間を表記するためのリテラル値
- hh = 2 桁の時間（00 から 23）
- mm = 2 桁の分（00 から 59）
- ss = 2 桁の秒（00 から 59）
- s = 秒の小数部分を表す 1 桁以上の数字
- TZD = タイムゾーン指定子（Z または+hh:mm または-hh:mm）
  - Z: UTC 時間
  - +hh:mm: UTC より進んでいる現地タイムゾーン
  - -hh:mm: UTC より遅れている現地タイムゾーン

ビジネスエンティティサービス REST 呼び出しを実行するための WebLogic の設定

ビジネスエンティティサービス REST 呼び出しは HTTP の基本認証を使用するため、REST 呼び出しに対する WebLogic Server 認証を無効にする必要があります。ビジネスエンティティサービス REST 呼び出しを実行するように WebLogic を設定するには、WebLogic の config.xml ファイルを編集します。

1. 次の WebLogic ディレクトリに移動します。

   UNIX の場合:

   ```
   <WebLogic installation directory: Weblogic のインストールディレクトリ>/user_projects/domains/base_domain/config
   ```

   Windows の場合:

   ```
   <WebLogic installation directory: Weblogic のインストールディレクトリ>\user_projects\domains\base_domain\config
   ```

2. テキストエディタで次のファイルを開きます。

   config.xml

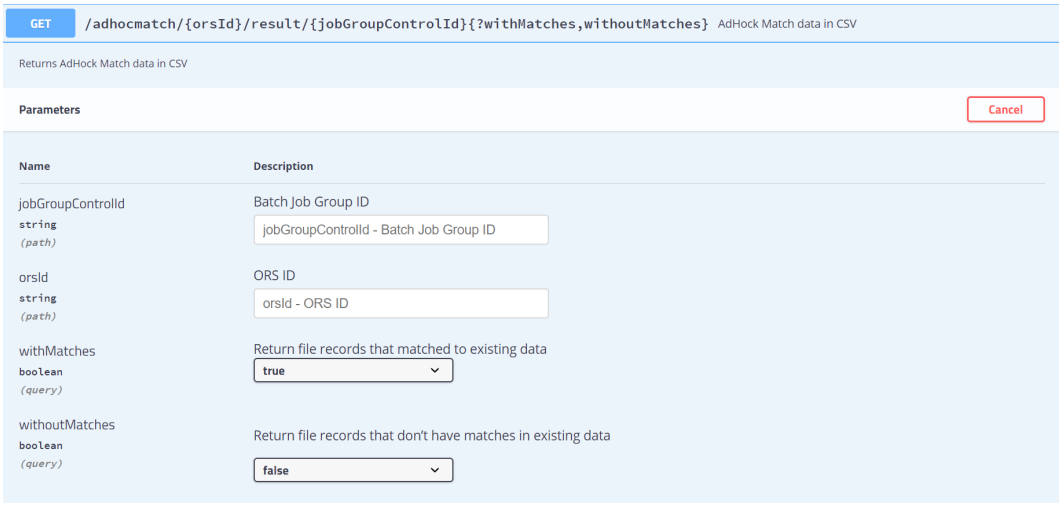
3. 終了タグ</security-configuration>の前に、次の XML コードを追加します。

   ```
   <enforce-valid-basic-auth-credentials>
      false
   </enforce-valid-basic-auth-credentials>
   ```

入力パラメータと出力パラメータの表示

Swagger を使用して、RESTAPI の入力パラメータと出力パラメータを表示できます。

次の図は、アドホック照合 REST API の Swagger での入力パラメータを示しています。



JavaScript テンプレート

次のコードサンプルは、REST ビジネスエンティティサービス呼び出し用の JavaScript コードを作成するために変更できる基本テンプレートを示しています。jQuery Java スクリプトライブラリが必要です。

```
(function ($) {
    window.CSClient = window.CSClient || {
        baseUrl: "/cmx/cs/" + "[siperian-client.orsId]",
        user: "[siperian-client.username]",
        pass: "[siperian-client.password]",

        process: function (method, url, body, params) {
            var fullUrl = this.baseUrl + url + ".json?" + $.param(params);
            return $.ajax({
                method: method,
                contentType: "application/json",
                url: fullUrl,
                data: JSON.stringify(body),
                beforeSend: function (xhr) {
                    xhr.setRequestHeader("Authorization", "Basic " + btoa(CSClient.user + ":" +
CSClient.pass));
                }
            });
        },

        readCo: function (url, params) {
            return this.process("GET", url, null, params);
        },
        createCo: function (url, body, params) {
            return this.process("POST", url, body, params);
        },
        updateCo: function (url, body, params) {
            return this.process("PUT", url, body, params);
        },
        deleteCo: function (url, params) {
            return this.process("DELETE", url, null, params);
        }
    };
})(jQuery);
```

JavaScript サンプル

リソースキットには、REST ビジネスエンティティサービス呼び出しの方法を示すサンプル Java ソースコードが含まれています。

このサンプルコードは次のファイルに含まれています。

<MDM Hub installation directory: MDM Hub のインストールディレクトリ>\hub\resourcekit\samples\COS\source\resources\webapp\rest-api.html

次のコードは、Person ルートレコードを作成し、複数の子レコードを追加し、子レコードを 1 つ削除し、続いて Person レコードとすべての子レコードを削除する REST API 呼び出しを示しています。

```
<html>
<head>
    <script type="text/javascript" src="jquery-1.11.1.js"></script>
    <script type="text/javascript" src="cs-client.js"></script>
</head>
<body>

<script type="text/javascript" language="javascript">
    $(document).ready(function () {
```

```
        $("#run").click(function () {

            log = function(msg, json) {
                $('#log').before("<hr/><b>" + msg + "</b>");
                $('#log').before("<pre>" + JSON.stringify(json, undefined, 2) + "</pre>");
            };

            CSClient.createCo(
                    "/Person",
                    {
                        firstName: "John",
                        lastName: "Smith"
                    },
                    {
                        systemName: "Admin"
                    }
            ).then(
                    function (result) {
                        log("PERSON CREATED:", result);
                        return CSClient.readCo(
                                "/Person/" + result.Person.rowidObject.trim(),
                                {
                                    depth: 1
                                }
                        );
                    }
            ).then(
                    function (result) {
                        log("READ CREATED PERSON:", result);
                        return CSClient.updateCo(
                                "/Person/" + result.rowidObject.trim(),
                                {
                                    genderCd: {
                                        genderCode: "M"
                                    },
                                    TelephoneNumbers: {
                                        item: [
                                            {
                                                phoneNumber: "111-11-11"
                                            },
                                            {
                                                phoneNumber: "222-22-22"
                                            }
                                        ]
                                    }
                                },
                                {
                                    systemName: "Admin"
                                }
                        );
                    }
            ).then(
                    function (result) {
                        log("PERSON UPDATED:", result);
                        return CSClient.readCo(
                                "/Person/" + result.Person.rowidObject.trim(),
                                {
                                    depth: 3,
                                    readSystemFields: true
                                }
                        );
                    }
            ).then(
                    function (result) {
                        log("READ UPDATED PERSON:", result);
                        return CSClient.deleteCo(
                                "/Person/" + result.rowidObject.trim() + "/TelephoneNumbers/" +
result.TelephoneNumbers.item[0].rowidObject.trim(),
                                {
                                    systemName: "Admin"
                                }
```

```
                    );
                }
            ).then(
                    function (result) {
                        log("TELEPHONE DELETED:", result);
                        return CSClient.readCo(
                                "/Person/" + result.Person.rowidObject.trim(),
                                {
                                    depth: 3
                                }
                        );
                    }
            ).then(
                    function (result) {
                        log("READ PERSON AFTER TELEPHONE IS DELETED:", result);
                        return CSClient.deleteCo(
                                "/Person/" + result.rowidObject.trim(),
                                {
                                    systemName: "Admin"
                                }
                        );
                    }
            ).then(
                    function (result) {
                        log("PERSON DELETED:", result);
                        return CSClient.readCo(
                                "/Person/" + result.Person.rowidObject.trim(),
                                {
                                    depth: 1,
                                    recordStates: "ACTIVE,PENDING,DELETED",
                                    readSystemFields: true
                                }
                        );
                    }
            ).then(
                    function (result) {
                        log("READ PERSON AFTER DELETE (HSI -1):", result);
                    }
            );
        });

    });
</script>

<input type="button" id="run" value="Run..."/>
<p/>
<div id="log"></div>
</body>
</html>
```

HTTP 基本認証を使用した AJAX 呼び出しの例

ビジネスエンティティサービスを呼び出すには、HTTP 基本認証が必要です。

次のコードは、jQuery を使用する HTTP 基本認証による AJAX 呼び出しの例を示しています。

```
<script type="text/javascript">
    function readMeta(user, pass){
            $.ajax({
                    type: "GET",
                    url: "/cmx/cs/localhost-mdm-DS_UI1/Person?action=meta",
                    dataType: 'json',
                    async: false,
                    headers: {
                            "Authorization": "Basic " + btoa(user + ":" + pass)
                    },
```

```
                success: function (data) {
                        alert('Data received! ' + data.object.label);
                }
        });
    }
</script>
```

この例では、btoa は文字列を Base64 形式でエンコードする標準関数です。

この呼び出しを実装する開発者は、ユーザー名とパスワードを渡す方法を決定できます。

注: AJAX 呼び出しを成功させるには、MDM Hub と同じドメインで JavaScript アプリケーションをホストします。

ビジネスエンティティサービス用の REST API リファレンス

このセクション「ビジネスエンティティサービス用の REST API リファレンス」では、REST API をリストアップし、各 API について説明しています。この API リファレンスには、URL、クエリパラメータ、サンプル要求、サンプル応答などの情報も挙げられています。

メタデータの取得

［メタデータの取得］REST API は、ビジネスエンティティのデータ構造またはビジネスエンティティリレーションを返します。

この API は、GET メソッドを使用して、ビジネスエンティティの次のメタデータを返します。

- ビジネスエンティティの構造
- フィールドのリスト
- データ型やサイズなどのフィールド型
- 作成、更新、マージなどの操作のセキュリティ設定
- ノードまたはフィールドのローカライズされたラベル
- ルックアップフィールドのコードおよび表示フィールドの名前
- フィールドおよびルックアップフィールドのデフォルト値

  注: 依存ルックアップフィールドでは、複数のデフォルト値はサポートされません。

この API は、ビジネスエンティティリレーションの次の詳細を返します。

- リレーションの名前とラベル。
- 開始および終了ビジネスエンティティ。
- リレーションの方向。

要求 URL

ビジネスエンティティの［メタデータの取得］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/<business entity>?action=meta
```

リレーションの［メタデータの取得］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/<relationship>?action=meta
```

メタデータ情報を取得するには、クエリパラメータ「action=meta」を使用します。ビジネスエンティティの名前は確実に正しく指定してください。

［メタデータの取得］URL に対して、次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/<context>/<database ID>/<business entity/relationship>?action=meta
```

要求には HTTP ヘッダーを追加できます。

サンプル API 要求

次のサンプル要求は、Person ビジネスエンティティとルートノードのメタデータ情報を取得します。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person?action=meta
```

次のサンプル要求には、JSON 形式の Person ビジネスエンティティのメタデータ情報を取得するヘッダが含まれます。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person?action=meta
Accept: application/json
```

次のサンプル要求は、HouseholdContainsMemberPerson リレーションのメタデータ情報を取得します。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/
ProductGroupProductGroupIsParentOfProductGroupProductGroup?action=meta
```

サンプル API 応答

エンティティとリレーションのサンプル応答には、セキュリティ権限が含まれています。最初の operations セクションでは、可能な権限が定義されています。object セクションには、ビジネスエンティティまたはリレーション全体に対する権限がリストされています。fields セクションでは、フィールドレベルでの権限が定義されています。

次の例は、JSON 形式の Person ビジネスエンティティの部分データ構造を示しています。

```
{
  "operations": {
    "read": {
      "allowed": true
    },
    "search": {
      "allowed": true
    },
    "create": {
      "allowed": true,
      "task": {
        "template": {
          "title": "Review changes in {taskRecord[0].label}",
          "priority": "NORMAL",
          "dueDate": "2018-04-24T09:28:13.455-04:00",
          "taskType": "AVOSBeUpdate",
          "comment": "This is urgent. Please review ASAP"
        },
        "comment": "AS_REQUIRED",
        "attachment": "OPTIONAL"
      }
    },
    "update": {
      "allowed": true,
      "task": {
        "template": {
          "title": "Review changes in {taskRecord[0].label}",
          "priority": "NORMAL",
          "dueDate": "2018-04-24T09:28:13.455-04:00",
          "taskType": "AVOSBeUpdate",
          "comment": "This is urgent. Please review ASAP"
        },
        "comment": "AS_REQUIRED",
```

```
      "attachment": "OPTIONAL"
    }
  },
  "merge": {
    "allowed": true,
    "task": {
      "template": {
        "title": "Review changes in {taskRecord[0].label}",
        "priority": "NORMAL",
        "dueDate": "2018-04-24T09:28:13.455-04:00",
        "taskType": "AVOSBeMerge",
        "comment": "This is urgent. Please review ASAP"
      },
      "comment": "AS_REQUIRED",
      "attachment": "OPTIONAL"
    }
  },
  "delete": {
    "allowed": true
  },
  "unmerge": {
    "allowed": true,
    "task": {
      "template": {
        "title": "Review changes in {taskRecord[0].label}",
        "priority": "NORMAL",
        "dueDate": "2018-04-24T09:28:13.455-04:00",
        "taskType": "AVOSBeUnmerge",
        "comment": "This is urgent. Please review ASAP"
      },
      "comment": "AS_REQUIRED",
      "attachment": "OPTIONAL"
    }
  }
},
"objectType": "ENTITY",
"timeline": true,
"object": {
  "operations": {
    "read": {
      "allowed": true
    },
    "create": {
      "allowed": true
    },
    "update": {
      "allowed": true
    },
    "merge": {
      "allowed": true
    },
    "delete": {
      "allowed": true
    },
    "unmerge": {
      "allowed": true
    }
  },
  "field": [
    {
      "operations": {
        "read": {
          "allowed": true
        },
        "create": {
          "allowed": true
        },
        "update": {
          "allowed": true
        }
      },
```

```
        "allowedValues": [
          "Person"
        ],
        "searchable": {
          "filterable": true,
          "facet": true
        },
        "name": "partyType",
        "label": "Party Type",
        "dataType": "String",
        "length": 255
      },
      {
        "operations": {
          "read": {
            "allowed": true
          },
          "create": {
            "allowed": true
          },
          "update": {
            "allowed": true
          }
        },
        "name": "lastName",
        "label": "Last Name",
        "dataType": "String",
        "length": 50
      },
      {
        "operations": {
          "read": {
            "allowed": true
          },
          "create": {
            "allowed": true
          },
          "update": {
            "allowed": true
          }
        },
        "searchable": {
          "filterable": true,
          "facet": true
        },
        "name": "displayName",
        "label": "Display Name",
        "dataType": "String",
        "length": 200
      },
    ...
    ],
    "name": "Person",
    "label": "Person",
    "many": false
  }
}
```

メタデータの一覧表示

［メタデータの一覧表示］REST API は、定義したビジネスエンティティまたはリレーションのリストを返します。ビジネスエンティティにタイムライン情報とセキュリティ情報が含まれている場合、応答にはその情報が含まれます。この API を使用して、あるエンティティから開始し、あるエンティティで終了する、または指定したエンティティから開始および終了するリレーションのリストを取得できます。

この API は GET メソッドを使用します。

［メタデータの一覧表示］URL

エンティティメタデータ用の［メタデータの一覧表示］URL の形式は、次のとおりです。

http://<host>:<port>/<context>/<database ID>/meta/entity

リレーションメタデータ用の［メタデータの一覧表示］URL の形式は、次のとおりです。

http://<host>:<port>/<context>/<database ID>/meta/relationship

［メタデータの一覧表示］URL に対して、次の HTTP GET 要求を行います。

GET http://<host>:<port>/<context>/<database ID>/meta/entity|relationship

クエリパラメータ

クエリパラメータを要求 URL に付加して、ビジネスエンティティリレーションの検索結果をフィルタリングできます。方向を指定して、ビジネスエンティティで開始および終了するリレーションを検索できます。

次の表にクエリパラメータを示します。

パラメータ	説明
start	オプション。リレーションが開始されるエンティティを指定します。 例えば、meta/relationship?start=Organization クエリは、Organization ビジネスエンティティから始まるすべてのリレーションを返します。
finish	オプション。リレーションが終了するエンティティを指定します。 例えば、meta/relationship?finish=Person クエリは、Person ビジネスエンティティで終わるすべてのリレーションを返します。

2 つのビジネスエンティティ間のすべてのリレーションを取得するには、両方のパラメータを指定します。例えば、meta/relationship?start=Organization&finish=Person クエリは、Organization ビジネスエンティティから始まるすべてのリレーションと Person ビジネスエンティティで終わるすべてのリレーションを返します。

サンプル API 要求

次のサンプル要求は、構成済みのビジネスエンティティのリストを取得します。

GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/meta/entity

次のサンプル要求は、構成済みのリレーションのリストを取得します。

GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/meta/relationship

次のサンプル要求は、Organization ビジネスエンティティから始まり Person ビジネスエンティティで終わるリレーションのリストを取得します。

GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/meta/relationship?start=Organization&finish=Person

サンプル API 応答

次の例は、構成されたリレーションのリストの抜粋を示しています。

```
{
  "link": [],
  "firstRecord": 1,
  "pageSize": 10,
  "item": [
    {
      "operations": {
        "read": {
          "allowed": true
```

```
    },
    "search": {
      "allowed": false
    },
    "create": {
      "allowed": true,
      "task": {
        "template": {
          "title": "Review changes in {taskRecord[0].label}",
          "priority": "NORMAL",
          "dueDate": "2018-04-24T09:31:48.167-04:00",
          "taskType": "AVOSBeUpdate",
          "comment": "This is urgent. Please review ASAP"
        },
        "comment": "AS_REQUIRED",
        "attachment": "OPTIONAL"
      }
    },
    "update": {
      "allowed": true,
      "task": {
        "template": {
          "title": "Review changes in {taskRecord[0].label}",
          "priority": "NORMAL",
          "dueDate": "2018-04-24T09:31:48.167-04:00",
          "taskType": "AVOSBeUpdate",
          "comment": "This is urgent. Please review ASAP"
        },
        "comment": "AS_REQUIRED",
        "attachment": "OPTIONAL"
      }
    },
    "merge": {
      "allowed": true,
      "task": {
        "template": {
          "title": "Review changes in {taskRecord[0].label}",
          "priority": "NORMAL",
          "dueDate": "2018-04-24T09:31:48.167-04:00",
          "taskType": "AVOSBeMerge",
          "comment": "This is urgent. Please review ASAP"
        },
        "comment": "AS_REQUIRED",
        "attachment": "OPTIONAL"
      }
    },
    "delete": {
      "allowed": true
    },
    "unmerge": {
      "allowed": true,
      "task": {
        "template": {
          "title": "Review changes in {taskRecord[0].label}",
          "priority": "NORMAL",
          "dueDate": "2018-04-24T09:31:48.167-04:00",
          "taskType": "AVOSBeUnmerge",
          "comment": "This is urgent. Please review ASAP"
        },
        "comment": "AS_REQUIRED",
        "attachment": "OPTIONAL"
      }
    }
  },
  "objectType": "ENTITY",
  "timeline": false,
  "object": {
    "link": [
      {
        "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/
                 CreditCard.json?action=meta",
```

```
    "rel": "entity"
  }
],
"field": [
  {
    "name": "issuingCompany",
    "label": "Issuing Company",
    "dataType": "String",
    "length": 100
  },
  {
    "name": "expirationYear",
    "label": "Expiration Year",
    "dataType": "String",
    "length": 4
  },
  {
    "allowedValues": [
      "Credit Card"
    ],
    "name": "accountType",
    "label": "Account Type",
    "dataType": "String",
    "length": 255
  },
  {
    "name": "accountNumber",
    "label": "Account Number",
    "dataType": "String",
    "length": 20
  },
  {
    "name": "securityCode",
    "label": "Security Code",
    "dataType": "String",
    "length": 4
  },
  {
    "name": "expirationMonth",
    "label": "Expiration Month",
    "dataType": "String",
    "length": 2
  },
  {
    "name": "cardholderName",
    "label": "Card Holder Name",
    "dataType": "String",
    "length": 100
  },
  {
    "name": "consolidationInd",
    "label": "Consolidation Ind",
    "dataType": "Integer",
    "length": 38,
    "readOnly": true,
    "system": true
  },
  {
    "name": "creator",
    "label": "Creator",
    "dataType": "String",
    "length": 50,
    "readOnly": true,
    "system": true
  },
  {
    "name": "interactionId",
    "label": "Interaction Id",
    "dataType": "Integer",
    "length": 38,
    "readOnly": true,
```

```
    "system": true
  },
  {
    "name": "updatedBy",
    "label": "Updated By",
    "dataType": "String",
    "length": 50,
    "readOnly": true,
    "system": true
  },
  {
    "name": "lastUpdateDate",
    "label": "Last Update Date",
    "dataType": "Date",
    "readOnly": true,
    "system": true
  },
  {
    "name": "lastRowidSystem",
    "label": "Last Rowid System",
    "dataType": "String",
    "length": 14,
    "readOnly": true,
    "system": true
  },
  {
    "name": "dirtyIndicator",
    "label": "Dirty Indicator",
    "dataType": "Integer",
    "length": 38,
    "readOnly": true,
    "system": true
  },
  {
    "name": "deletedBy",
    "label": "Deleted By",
    "dataType": "String",
    "length": 50,
    "readOnly": true,
    "system": true
  },
  {
    "name": "deletedInd",
    "label": "Deleted Indicator",
    "dataType": "Integer",
    "length": 38,
    "readOnly": true,
    "system": true
  },
  {
    "name": "hubStateInd",
    "label": "Hub State Ind",
    "dataType": "Integer",
    "length": 38,
    "readOnly": true,
    "system": true
  },
  {
    "name": "deletedDate",
    "label": "Deleted Date",
    "dataType": "Date",
    "readOnly": true,
    "system": true
  },
  {
    "name": "rowidObject",
    "label": "Rowid Object",
    "dataType": "String",
    "length": 14,
    "readOnly": true,
    "system": true
```

```
          },
          {
            "name": "cmDirtyInd",
            "label": "Content metadata dirty Ind",
            "dataType": "Integer",
            "length": 38,
            "readOnly": true,
            "system": true
          },
          {
            "name": "createDate",
            "label": "Create Date",
            "dataType": "Date",
            "readOnly": true,
            "system": true
          }
        ],
        "name": "CreditCard",
        "label": "Credit Card",
        "many": false
      }
    },
    ...
  ]
}
```

一致カラムのリスト

［一致カラムのリスト］REST API では、指定したビジネスエンティティの一致ルールセットのリスト、または指定した一致ルールセットの一致カラムのリストを返せます。一致カラムのリストを生成し、SearchMatch REST API で一致カラムを使用できます。

一致カラムと一致ルールセットの設定の詳細については、*Multidomain MDM の設定ガイド*を参照してください。

この API は GET メソッドを使用します。

要求 URL

［一致カラムのリスト］URL のコンテキストは、cmx です。ホストされた MDM 環境では、<テナント名>/cmx のように、コンテキストにテナント名を含めます。

［一致カラムのリスト］URL の形式は次のとおりです。

すべての一致ルールセットを返す URL

次の URL を使用して、特定のビジネスエンティティの一致ルールセットすべてをリストします。

http://<host>:<port>/<context>/queryTemplate/<database ID>/<business entity>

［一致カラムのリスト］URL に対して、次の HTTP GET 要求を行います。

GET http://<host>:<port>/<context>/queryTemplate/<database ID>/<business entity>

指定した一致ルールセットで使用される一致カラムを返す URL

次の URL を使用して、指定した一致ルールセットのすべての一致カラムをリストします。

http://<host>:<port>/<context>/queryTemplate/<database ID>/<business entity>/<match rule set>

サンプル API 要求

一致ルールセットの要求

次のサンプル要求は、Person ビジネスエンティティの一致ルールセットをリストします。

```
GET http://localhost:8080/cmx/queryTemplate/localhost-orcl-DS_UI1/Person
```

一致カラムの要求

次のサンプル要求は、一致ルールセット IDL2 で使用されている一致カラムをリストします。

```
GET http://localhost:8080/cmx/queryTemplate/localhost-orcl-DS_UI1/Person/IDL2
```

サンプル API 応答

次のサンプル応答は、IDL2 一致ルールセットに含まれる一致カラムを示します。

```
{
 "queryTemplates":[
 {
  "businessEntity": "Person",
  "matchRuleSet": "IDL2",
  "type": "extended",
  "searchFields": [
    {
      "name": "displayName",
      "mandatory": true
    },
    {
      "name": "BillAddresses.Address.addressLine1",
      "mandatory": false
    },
    {
      "name": "ShipAddresses.Address.addressLine2",
      "mandatory": false
    },
    {
      "name": "ShipAddresses.Address.addressLine1",
      "mandatory": false
    }
  ]
 }
]
}
```

レコードの読み取り

［レコードの読み取り］REST API は、ビジネスエンティティ内のルートレコードの詳細を返します。この API は、ルートレコードの子レコードの詳細を返すために使用できます。この API を使用し、レコードのコンテンツメタデータを表示できます。

この API は GET メソッドを使用します。

結果セットをソートして、昇順または降順で表示できます。より多くの複雑なパラメータが必要な場合は、POST メソッドを使用してください。例えば、データを取得して、子要素を複数のフィールドでソートしたい場合などです。

要求 URL

行 ID、またはソースシステムとソースキーを使用して、要求 URL にレコードを指定します。

［レコードの読み取り］URL は、次の形式にすることができます。

行 ID を指定した URL

行 ID を指定する場合は次の URL 形式を使用します。

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<rowId of the root record>
```

この URL に対して次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/<context>/<database ID>/<business entity>/<rowId of the root record>
```

ソースシステム名とソースキーを指定した URL

ソースシステム名とソースキーを指定する場合は、次の URL 形式を使用します。

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<system name>:<source key>
```

システム名とオブジェクトのグローバルビジネス識別子（GBID）を指定した URL

ソースシステム名と GBID を指定する場合は、次の URL 形式を使用します。

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<system name>:uid:<gbid>
```

GBID のみを指定した URL

GBID のみを指定する場合は次の URL 形式を使用します。

```
http://<host>:<port>/<context>/<database ID>/<business entity>/:uid:<gbid>
```

複数の GBID を指定した URL

複数の GBID を指定する場合は次の URL 形式を使用します。

```
http://<host>:<port>/<context>/<database ID>/<business entity>/:one:<gbid>,another:<gbid>
```

子ノードの詳細を返す URL

子ノードの詳細を返すには、次の URL 形式を使用します。

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<rowId of the record>?depth=n
```

子ノードの詳細を返す URL

子ノードの詳細を返すには、次の URL 形式を使用します。

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<rowId of the record>?children=<comma
separated list of child node names or paths>
```

例: `children= BillAddresses/Address,Email`

特定のノードの詳細を返す URL

特定のノードの詳細を返すには、次の URL 形式を使用します。

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<rowId of the record>/<node name>
```

特定のノードの子の詳細を返す URL

特定のノードの子の詳細を返すには、次の URL 形式を使用します。

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<rowId of the record>/<node name>?
children=<child node name>
```

レコードのコンテンツメタデータを返す URL

レコードのコンテンツメタデータを返すには、次の URL 形式を使用します。

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<rowId of the root record>?
contentMetadata=<content metadata type>
```

例えば、次の GET 要求で、子レコードの一致を取得できます。

```
GET http://<host>:<port>/<context>/<database ID>/<business entity>/<rowId of the root record>?
contentMetadata=MATCH
```

子要素をフィールドでソートする URL

子要素をフィールドでソートするには、次の URL 形式を使用します。

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<rowId of the root record>/<node name>?order=-<field name>
```

-をサフィックスとして使用すると、降順でソートされます。

クエリパラメータ

レコードの詳細をフィルタリングする場合、要求 URL にクエリパラメータを付け加えることができます。

次の表にクエリパラメータを示します。

パラメータ	説明
depth	返す子レベルの数。ルートノードとその直接の子を返すには 2 を指定し、ルートノード、直接の子、および孫を返すには 3 を指定します。デフォルトは 1 です。
effectiveDate	データの取得対象となる日付。
readSystemFields	結果でシステムフィールドを返すかどうかを示します。デフォルトは false です。
recordStates	レコードの状態。カンマ区切りの状態リストを指定します。サポートされるレコードの状態は、ACTIVE、PENDING、および DELETED です。デフォルトは ACTIVE です。
contentMetadata	レコードのメタデータ。カンマ区切りのリストを指定します。例えば、「XREF, PENDING_XREF, DELETED_XREF, HISTORY, XREF_HISTORY, and MATCH」のように指定します。 MATCH を選択すると、応答には_MTCH テーブルから取得した一致するレコードが含まれます。
historyDate	履歴データの取得対象となる日付。応答には、_HIST テーブルから取得した、指定した日付のレコードデータが含まれます。 historyDate を contentMetadata パラメータとともに使用して、履歴メタデータを取得できます。contentMetadata は XREF、BVT、または TRUST に設定します。 - XREF。応答には、_HXRF テーブルからの履歴相互参照データが含まれます。 - BVT。応答には、_HCTL テーブルからの履歴ベストバージョンオブトゥルースが含まれます。 - TRUST。応答には、_HCTL および_HVXR テーブルからの履歴信頼設定が含まれます。
children	子ノードの名前またはパスのカンマ区切りリスト。
suppressLinks	API 応答に親と子のリンクを含めるかどうかを示します。応答に親と子のリンクをいっさい含めない場合は、このパラメータを true に設定します。デフォルトは false です。 例えば、Person/1242?depth=10&suppressLinks=true クエリでは、レコードの詳細が最大 10 の子レベルまで表示され、応答に親と子のリンクは含まれません。

パラメータ	説明
order	フィールド名のカンマ区切りリスト。オプションでプレフィックス+または-を指定できます。プレフィックス+は結果を昇順でソートし、プレフィックス-は結果を降順でソートすることを指定します。デフォルトは+です。複数のパラメータを指定した場合、結果セットは、指定した最初のパラメータでまずソートされてから、次のパラメータでソートされます。 例えば、Person/1242/Names?order=-name クエリでは、結果は名前の降順で表示されます。 Person/1242/BillAddresses?order=rowidObject,-effStartDate クエリでは、請求先住所が行 ID で昇順にソートされ、次に有効開始日で降順にソートされて表示されます。

resolveLookup	指定されたビジネスエンティティのルックアップフィールド全体を取得します。パラメータを true に設定して、ルックアップフィールドをロードし、応答に含めます。デフォルトは false です。
	例えば、addressType フィールドは、Person ビジネスエンティティの子レベルのルックアップフィールドです。
	resolveLookup パラメータが false に設定されている場合、次の REST API 応答を受信する可能性があります。
	`{ "label": "LU Address Type", "addressType": "BILL" }`
	resolveLookups パラメータが true に設定されている場合、REST API 応答には追加の詳細が含まれ、次の REST API 応答を受信する可能性があります。
	`{ "label": "LU Address Type", "addressType": "BILL", "addressTypeDisp": "BILLING" }`

次のサンプルは、レコードの詳細をフィルタリングする方法を示しています。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/123/Phone/SFA:456/PhoneUse?
recordsToReturn=100&recordStates=ACTIVE,PENDING&contentMetadata=XREF
```

関連項目：

- 「UTC での日付と時刻の形式」 (ページ 34)

子要素のソート順を指定する POST 要求

複数のフィールドで結果セットを並べ替えるには POST 要求を使用します。POST 本文にパラメータまたはフィールドを含めます。

次のサンプル要求は、読み取り操作用に POST 要求を使用して、複数のフィールドでデータをソートする方法を示しています。

```
http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/ReadPerson.json
{
 parameters:
```

```
{
  coFilter: {
  object: {
    name:"Person",
    key: {
      rowid: 1242
    },
    order: "lastName",
    object:[
      {name:"Names", order:"-name"},
      {name:"Phone", order:"phoneNum, -phoneCountryCd",
      object:[{name:"PhonePermissions", order:"-column1"}]
      }
   ]}
  }
 }
}
```

注: ビジネスエンティティの各レベルで、各タイプの子に対して指定できるソート順は 1 つだけです。

ソート順についての注意事項

［レコードの読み取り］API は、各ビジネスエンティティ子ノードに対して 1 つ以上のフィールドによるソートをサポートします。次のセクションでは、ソート順を指定するときの注意事項について説明します。

- 孫に対してソート順が指定され、子に対してソート順が指定されていない場合、孫要素は指定のソート順でソートされますが、子要素は孫用に指定されているソート順ではソートされません。次にサンプル要求を示します。

  ```
  http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/1242/Phone/861/PhonePermissions?order=-column1
  ```

  このサンプル要求では、孫 PhonePermissions に対しては降順が指定されていますが、子要素 Phone にはソート順が指定されていません。Phone は、PhonePermissions のソート順でソートされません。

- 子に対してソート順が指定され、孫に対してはソート順が指定されていない場合、子は指定されているソート順でソートされますが、孫は子に対して指定されているソート順でソートされません。次にサンプル要求を示します。

  ```
  {parameters:
   {coFilter: {
     object: {
       name:"Person", key: { rowid: 1242 }, order: "lastName",
       object:[
         {name:"Names", order:"-name"},
         {name:"Phone", order:"-phoneCountryCd, -phoneNum", object:[{name:"PhonePermissions"}]},
      ]}
  }}}
  ```

  このサンプル要求では、子 Phone に対してはソート順が指定されていますが、孫 PhonePermissions には指定されていません。子 Phone は指定のソート順でソートされます。

- 子と孫に対してソート順が指定されている場合、両方がそれぞれのソート順でソートされます。次のサンプル要求は、Phone（子）と PhonePermissions（孫）に対してソート順を指定しています。

  ```
  {parameters:
   {coFilter: {
     object: {
       name:"Person", key: { rowid: 1242 }, order: "lastName",
       object:[
         {name:"Names", order:"-name"},
         {name:"Phone", order:"-phoneCountryCd, -phoneNum", object:[{name:"PhonePermissions", order:"-
  column1"}]},
      ]}
  }}}
  ```

- 子は子のカラムでのみソートされ、孫は孫のカラムでのみソートされます。次のサンプル要求では、Phone は PhoneType で、PhonePermissions はカラム 1 でソートされます。PhoneType は Phone（子）のカラムで、column1 は PhonePermissions（孫）のカラムです。

  ```
  http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/1242/Phone?order=-PhoneType
  http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/1242/Phone/861/phonePermissions?order=column1
  ```

- ビジネスエンティティの各レベルで、各タイプの子に対して指定できるソート順は 1 つだけです。次の要求では、親が別々の子 PhonePermissions に対して別々のソート順が指定されています。ただし、最初に降順が指定されているので、どちらの親（rowid 861 および rowid 862）の子 PhonePermissions も降順でソートされます。

```
{parameters:
 {coFilter: {
   object: {
     name:"Person", key: { rowid: 1242 }, order: "lastName",
     object:[
       {name:"Names", order:"-name"},
       {name:"Phone", key: { rowid:861 }, order:"+phoneCountryCd, -phoneNum", object:
[{name:"PhonePermissions", order:"-column1"}]},
       {name:"Phone", key: {rowid:862}, order:"phoneNum, -phoneCountryCd", object:
[{name:"PhonePermissions", order:"column1"}]}
     ]}
}}}
```

サンプル API 要求

次のサンプル要求は、Person ビジネスエンティティ内のルートレコードの詳細を返します。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/102
```

次のサンプル要求は、行 ID が 2 である子レコードの詳細を返します。子ベースオブジェクトは genderCd で、子レコードは深度 2 の位置にあります。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/102/genderCd/2?depth=2
```

次のサンプル要求は、システム名とソースキーを使用してルートレコードの詳細を返します。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/SFA:9000000000
```

次のサンプル要求は、ルートレコードとその相互参照レコードの詳細を返します。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/102?contentMetadata=XREF
```

次のサンプル要求は、ルートレコードの詳細を、名前の降順で返します。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/1242/Names?order=-name
```

次のサンプル要求は、請求先住所を、行 ID で昇順にソートした後、有効開始日で降順にソートして返します。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/1242/BillAddresses?order=rowidObject,-
effStartDate
```

サンプル API 応答

次のサンプルは、Person ビジネスエンティティ内のルートレコードの詳細を示しています。

```
{
    "link": [
        {
            "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/102",
            "rel": "self"
        },
        {
            "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/102?depth=2",
            "rel": "children"
        },
```

```
        {
            "href": "http://localhost:8080/cmx/request/hm_icons/person_small.jpeg?ors=localhost-orcl-DS_UI1",
            "rel": "icon"
        }
    ],
    "rowidObject": "102           ",
    "label": "DARWENT,JIMMY",
    "partyType": "Person",
    "statusCd": "A ",
    "lastName": "DARWENT",
    "middleName": "N",
    "firstName": "JIMMY",
    "displayName": "JIMMY N DARWENT",
    "genderCd": {
        "link": [
            {
                "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/102",
                "rel": "parent"
            },
            {
                "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/102/genderCd",
                "rel": "self"
            },
            {
                "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/102/genderCd?depth=2",
                "rel": "children"
            }
        ],
        "genderCode": "M"
    },
    "generationSuffixCd": {
        "link": [
            {
                "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/102",
                "rel": "parent"
            },
            {
                "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/102/generationSuffixCd?
depth=2",
                "rel": "children"
            },
            {
                "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/102/generationSuffixCd",
                "rel": "self"
            }
        ],
        "generationSuffixCode": "I"
    }
}
```

次のサンプルでは、genderCd ベースオブジェクト内の子レコードの詳細が示されています。

```
{
    "link": [
        {
            "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/102",
            "rel": "parent"
        },
        {
            "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/102/genderCd/2?depth=2",
            "rel": "children"
        },
        {
            "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/102/genderCd/2",
            "rel": "self"
        }
    ],
    "rowidObject": "2             ",
    "label": "LU Gender",
    "genderDisp": "MALE",
```

```
    "genderCode": "M"
}
```

レコードの作成

［レコードの作成］REST API は、指定されたビジネスエンティティのレコードを作成します。要求本文でレコードデータを送信します。［昇格］API を使用して、ビジネスエンティティのレコードの昇格と追加を行います。

この API は、POST メソッドを使用してレコードを作成します。

要求 URL

［レコードの作成］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/<business entity>?systemName=<name of the source system>
```

注: ソースシステムの名前は URL の必須パラメータです。

［レコードの作成］URL に対して、次の HTTP POST 要求を行います。

```
POST http://<host>:<port>/<context>/<database ID>/<business entity>?systemName=<name of the source system>
```

要求と一緒に送信するデータのメディアタイプを指定するには、Content-Type ヘッダーを追加します。

```
POST http://<host>:<port>/<context>/<database ID>/<business entity>?systemName=<name of the source system>
Content-Type: application/<json/xml>
```

URL パラメータ

ソースシステムの名前は要求 URL の必須パラメータです。

次の表に、URL に使用できるパラメータを示します。

パラメータ	説明
systemName	ソースシステムの名前。
interactionId	相互作用の ID。複数の要求をグループ化し、単一の相互作用にすることができます。すべての変更は相互作用 ID を使用して実行されます。
ignoreWarnings	警告レベルの検証メッセージを無視するかどうかを指定します。デフォルトは false です。警告メッセージを無視するには、true に設定します。
startDate と endDate	レコードが有効である期間を指定します。これらのパラメータは、タイムラインが有効になったベースオブジェクトに指定します。
validateOnly	書き込みビジネスエンティティサービスが着信データを検証するかどうかを示します。デフォルトは false です。
recordState	レコードの状態。このパラメータは、レコードの初期状態を指定するために使用します。ACTIVE または PENDING を使用します。デフォルトは ACTIVE です。
taskComment	API によってトリガされたワークフロータスクにコメントを追加します。
taskAttachments	タスク添付が有効になっている場合は、API によってトリガされたワークフロータスクにファイルを添付します。

関連項目：

- 「UTC での日付と時刻の形式」 (ページ 34)

要求本文

レコードのデータを REST 要求本文で送信します。データを送信するには、JSON 形式または XML 形式を使用します。ビジネスエンティティの構造の取得と必須パラメータ値の指定を行うには、要求本文内で［メタデータの取得］API を使用します。

ビジネスエンティティレコードを作成するために REST API 要求を送信した後、Informatica MDM Hub はフィールドの信頼スコアを計算します。信頼スコアが計算された空のフィールドに対して REST API 要求を送信できます。要求の本文には二重引用符を指定する必要があります。

次の JSON コードサンプルは、計算された信頼スコアを使用して空の［性別コード］フィールドを要求します。

```
{
    "firstName": "Sasha",
    "genderCd": ""
}
```

二重引用符の代わりに NULL を使用すると、Informatica MDM Hub はフィールドの信頼スコアを計算しません。

応答ヘッダーと応答本文

応答が正常な場合、API は応答ヘッダー内の interactionId および processId と、応答本文内のレコード詳細を返します。

プロセスが相互作用 ID を生成し、それをレコードの作成に使用する場合、API はその相互作用 ID を返します。プロセスが、レコードを直接データベースに保存せず、ワークフローを開始する場合、API はワークフロープロセスの ID であるプロセス ID を返します。

次の例は、相互作用 ID とプロセス ID が含まれる応答ヘッダーを示しています。

```
BES-interactionId: 72200000242000
BES-processId: 15948
```

応答本文には、生成された行 ID とともにレコードが含まれています。

サンプル API 要求

次のサンプル要求は、Person ビジネスエンティティ内にレコードを作成します。この要求は、API によってトリガされたワークフロータスクにコメントと添付ファイルを追加します。

```
POST http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person?systemName=Admin&taskComment=Read my
comment&taskAttachments=TEMP_SVR1.290T8,TEMP_SVR1.290T9
Content-Type: application/json

{
   "firstName":"John",
   "lastName":"Smith",
   "Phone":{
      "item":[
         {
            "phoneNumber":"111-11-11"
         }
      ]
   }
}
```

POST REST API 要求を使用して新しいレコードを作成する場合、POST リクエスト本文の sourceKey 属性を設定できます。sourceKey 属性は、XREF テーブルの pkey_src_object 値として使用されます。例えば、次の POST 要求を送信して、新しい Party レコードと、ソースシステムの SFA と pkey_src_object 123 を使用した Party XREF レコードを作成したい場合があります。

```
 POST http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person?systemName=SFA
{
    "firstName": "John",
    "middleName": "jr",
    "lastName": "Smith",
    "genderCd": {
        "genderCode": "M"
    },
    "key": {
        "sourceKey": "123"
    }
}
```

サンプル API 応答

次のサンプル応答は、レコードが正常に作成された後の応答ヘッダーと応答本文を示しています:

```
BES-interactionId: 72200000242000
BES-processId: 15948
Content-Type: application/json
{
    "Person": {
        "key": {
            "rowid": "2198246",
            "sourceKey": "72200000241000"
        },
        "rowidObject": "2198246",
        "Phone": {
            "item": [
                {
                    "key": {
                        "rowid": "260961",
                        "sourceKey": "72200000243000"
                    },
                    "rowidObject": "260961"
                }
            ]
        }
    }
}
```

レコードの更新

［レコードの更新］REST API は、指定されたルートレコードとその子レコードを更新します。要求 URL でレコードの ID を送信します。要求の本文で変更の要約を送信します。

変更後、レコードが保留中状態の場合は、［昇格］API を使用して変更を昇格させます。例えば、更新によってレビューワークフローがトリガされた場合、レビューが完了するまでレコードは保留中状態になります。

この API は POST メソッドを使用します。

ルートレコードの場合、要求本文に変更サマリではなく変更されたフィールドが含まれる、簡易化された PUT バージョンを使用することも選択できます。ルートレコードと関連する子レコードや孫レコードを更新するには、POST バージョンを使用します。または、PKEY を持つ PUT バージョンを使用したり、子レコードの rowidObject を指定することもできます。

要求 URL

［レコードの更新］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<rowId>?systemName=<name of the source system>
```

注: ソースシステムの名前は URL の必須パラメータです。

［レコードの更新］URL に対して、次の HTTP POST 要求を行います。

```
POST http://<host>:<port>/<context>/<database ID>/<business entity>/<rowId>?systemName=<name of the source system>
```

要求と一緒に送信するデータのメディアタイプを指定するには、Content-Type ヘッダーを追加します。

```
PUT http://<host>:<port>/<context>/<database ID>/<business entity>/<rowId>?systemName=<name of the source system>
Content-Type: application/<json/xml>
```

クエリパラメータ

ソースシステムの名前は必須クエリパラメータです。

要求には以下のクエリパラメータを使用できます。

パラメータ	説明
systemName	ソースシステムの名前。
interactionId	相互作用の ID。複数の要求をグループ化し、単一の相互作用にすることができます。すべての変更は相互作用 ID を使用して実行されます。
ignoreWarnings	警告レベルの検証メッセージを無視するかどうかを指定します。デフォルトは false です。警告メッセージを無視するには、true に設定します。
startdate と enddate	レコードが有効である期間を指定します。これらのパラメータは、タイムラインが有効になったベースオブジェクトに指定します。
validateOnly	着信データを検証するかどうかを指定します。デフォルトは false です。
recordState	レコードの状態を設定します。ACTIVE、PENDING、または DELETED を使用します。 例えば、recordState=ACTIVE を設定し、要求を論理的に削除されたレコードで実行すると、要求はレコードをアクティブな状態に復元します。
taskComment	API によってトリガされたワークフロータスクにコメントを追加します。
taskAttachments	タスク添付が有効になっている場合は、API によってトリガされたワークフロータスクにファイルを添付します。

関連項目：

- 「UTC での日付と時刻の形式」 (ページ 34)

要求本文

更新するデータを REST 要求本文で送信します。データを送信するには、JSON 形式または XML 形式を使用します。

新しいパラメータ値を指定します。$original パラメータを使用し、更新するパラメータの古い値を指定します。

子レコードでは、次のプロパティも使用できます。

プロパティ/要素	タイプ	説明
MATCH	オブジェクト	一致テーブルから子レコードの一致候補を追加または削除する場合は、子レコードに MATCH オブジェクトを追加します。
MERGE	オブジェクト	子レコードをマージするか、マージから候補を削除する場合は、子レコードに MERGE オブジェクトを追加します。

次の JSON コードサンプルは、ルートレコードの名を Bob に変更します。

```
{
  rowidObject: "123",
  firstName: "Bob",
  lastName: "Smith",
    $original: {
      firstName: "John"
                }
}
```

次の JSON コードサンプルは、Address 子レコードの一致候補を削除して、2 つの Telephone 子レコードのマージを定義します。

```
{
  rowidObject: "123",
  firstName: "Bob",
  lastName: "Smith",
    $original: {
      firstName: "John"
                }
  Address: {     // remove A3 from the matches for A2 in the Address_MTCH table
    item: [
      {
        rowidObject: "A2",
        MATCH: {
          item: [   // to remove matched child records for A2, specify null
            null
          ],
          $original: {
            item: [{key: {rowid: 'A3'}}]
          }
        }
      }
    ]
  }
  Telephone: {   // override the matches for the telephone child records
    item:[
      {
        rowid: "T1",
        MERGE: {
          item: [   // to remove merge candidates for T1, specify null
            null,
            null
          ],
          $original: {
            item: [
              {rowid: "T2"},
              {rowid: "T3"}
            ]
          }
        }
      },
      {
        rowid: "T4",
        MERGE: {
```

```
          item: [   // to add or override matches, specify matched records
            {rowid: "T2"}
          ],
        $original: {
          item: [
            null
          ]
        }
      }
    ]
  }
}
```

ビジネスエンティティレコードを更新するために REST API 要求を送信した後、Informatica MDM Hub はフィールドの信頼スコアを計算します。信頼スコアが計算された空のフィールドに対して REST API 要求を送信できます。要求の本文には二重引用符を指定する必要があります。

次の JSON コードサンプルは、計算された信頼スコアを使用して空の［性別コード］フィールドを要求します。

```
{
    "firstName": "Sasha",
    "genderCd": ""
}
```

二重引用符の代わりに NULL を使用すると、Informatica MDM Hub はフィールドの信頼スコアを計算しません。

応答ヘッダー

応答が正常な場合、API は応答ヘッダー内の interactionId および processId と、応答本文内のレコード詳細を返します。

プロセスが相互作用 ID を生成し、それをレコードの作成に使用する場合、API はその相互作用 ID を返します。プロセスが、レコードを直接データベースに保存せず、ワークフローを開始する場合、API はワークフロープロセスの ID であるプロセス ID を返します。

次の例は、相互作用 ID とプロセス ID が含まれる応答ヘッダーを示しています。

```
BES-interactionId: 72200000242000
BES-processId: 15948
```

応答本文には、生成された行 ID を持つレコードが含まれています。

サンプル API 要求

次のサンプル要求は、ビジネスエンティティ内のルートレコードとその子レコードを更新します。Person はビジネスエンティティで、Phone は子ベースオブジェクトです。この要求は、API によってトリガされたワークフロータスクにコメントと添付ファイルを追加します。

```
POST  http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/233?systemName=Admin
```

```
{
    rowidObject: "233",
    firstName: "BOB",
    lastName: "LLOYD",
    Phone: {
        item: [
            {
                rowidObject: "164",
                phoneNumber: "777-77-77",
                $original: {
                    phoneNumber: "(336)366-4936"
                }
            }
```

```
        ]
    },
    $original: {
        firstName: "DUNN"
    }
}
```

サンプル API 応答

次のサンプル応答は、レコードが正常に更新された後の応答ヘッダーと応答本文を示しています:

```
BES-interactionId: 72300000001000
BES-processId: 16302
{
    Person: {
        key: {
            rowid: "233",
            sourceKey: "SYS:233"
        },
        rowidObject: "233",
        preferredPhone: {
            key: {}
        }
    }
}
```

レコードの削除

［レコードの削除］REST API は、ビジネスエンティティのルートレコードを削除します。この API を使用して、ルートレコードの子レコードを削除します。

この API は、DELETE メソッドを使用してレコードを削除します。

要求 URL

［レコードの削除］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<rowID of the root record>?systemName=Admin
```

注: ソースシステムの名前は URL の必須パラメータです。

［レコードの削除］URL に対して、次の HTTP DELETE 要求を行います。

```
DELETE http://<host>:<port>/<context>/<database ID>/<business entity>/<rowID of the record>?systemName=Admin
```

ルートレコードの子レコードを削除するには、次の URL 形式を使用します。

```
DELETE http://<host>:<port>/<context>/<database ID>/<business entity>/<rowID of the record>/<child base object>/<rowID of the child record>?systemName=Admin
```

クエリパラメータ

ソースシステムの名前は必須 URL パラメータです。ソースシステムを指定するには、systemName パラメータを使用します。

サンプル API 要求

次のサンプル要求は、Person ビジネスエンティティ内のルートレコードを削除します。

```
DELETE http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/292258?systemName=Admin
```

サンプル API 応答

次のサンプル応答では、Person ビジネスエンティティ内のルートレコードが正常に削除された後、応答が示されています。

```
{
    "Person": {
        "key": {
            "rowid": "292258",
            "sourceKey": "WRK50000_7016"
        },
        "rowidObject": "292258"
    }
}
```

リストレコード

［レコードの一覧表示］REST API は、ルックアップ値または外部キー値のリストを返します。ルックアップでは、参照データが表示されるほか、指定されたカラムに入り得る値のリストが返されます。

この API は GET メソッドを使用します。

この API は、ルックアップコード値とルックアップコードの説明を取得するためにも使用できます。ルックアップのソート順を指定できます。より多くの複雑なパラメータが必要な場合は、POST メソッドを使用してください。

要求 URL

［レコードの一覧表示］REST URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/<lookup table>?action=list
```

この URL に対して次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/<context>/<database ID>/<lookup table>?action=list
```

ルックアップ値のコードを表示するには、次の URL 形式を使用します。

```
http://<host>:<port>/<context>/<database ID>/<lookup table>?action=list&idlabel=<lookup code>%3A<lookup display name>
```

注: ルックアップ値を表示するには、［メタデータの取得］API を使用して正確な URL を取得します。

クエリパラメータ

クエリパラメータを要求 URL に追加して、結果をフィルタリングできます。

次の表にクエリパラメータを示します。

パラメータ	説明
suppressLinks	API 応答に親と子のリンクを含めるかどうかを示します。応答に親と子のリンクをいっさい含めない場合は、このパラメータを true に設定します。デフォルトは false です。 例えば、Person/1242?depth=10&suppressLinks=true クエリでは、レコードの詳細が最大 10 の子レベルまで表示され、応答に親と子のリンクは含まれません。
order	ルックアップ値を昇順または降順で表示するのに使用します。+をプレフィックスとして使用すると昇順、-をプレフィックスとして使用すると降順でソートされます。デフォルトでは、プレフィックスを指定しない場合、結果セットは昇順でソートされます。 例えば、LUNamePrefix?action=list&order=-namePrefixDisp クエリでは、名前のプレフィックスが、プレフィックスの表示名で降順にソートされて表示されます。

resolveLookup	指定されたビジネスエンティティのルックアップフィールド全体を取得します。パラメータを true に設定して、ルックアップフィールドをロードし、応答に含めます。デフォルトは false です。
	例えば、addressType フィールドは、Person ビジネスエンティティの子レベルのルックアップフィールドです。
	resolveLookup パラメータが false に設定されている場合、次の REST API 応答を受信する可能性があります。
	`{ "label": "LU Address Type", "addressType": "BILL" }`
	resolveLookups パラメータが true に設定されている場合、REST API 応答には追加の詳細が含まれ、次の REST API 応答を受信する可能性があります。
	`{ "label": "LU Address Type", "addressType": "BILL", "addressTypeDisp": "BILLING" }`

ソート順を指定する POST 要求

ルックアップ値のソート順を指定するには、POST 要求を使用します。POST 本文にパラメータまたはフィールドを含めます。

次の例は、一覧表示操作での POST 要求の使用方法を示しています。

```
http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/ListCO.json
{
 parameters:
 {
   coFilter: {
```

```
   object: {
     name:"LUCountry",
     order: "-countryNameDisp"
   }
  }
 }
}
```

サンプル API 要求

次のサンプル要求は、ルックアップ値を表示します。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/LUGender?action=list
```

次のサンプル要求は、性別ルックアップ値のコードを表示します。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/LUGender?action=list&idlabel=genderCode%3AgenderDisp
```

次のサンプル要求は、名前のプレフィックスを、プレフィックスの表示名で降順にソートして表示します。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/LUNamePrefix?action=list&order=-namePrefixDisp
```

サンプル API 応答

次のサンプル応答では、ルックアップ値のリストが示されています。

```
{
    "firstRecord": 1,
    "pageSize": 2147483647,
    "searchToken": "SVR1.AU5LC",
    "item": [
        {
            "rowidObject": "1             ",
            "genderDisp": "UNKNOWN",
            "genderCode": "N"
        },
        {
            "rowidObject": "2             ",
            "genderDisp": "MALE",
            "genderCode": "M"
        },
        {
            "rowidObject": "3             ",
            "genderDisp": "FEMALE",
            "genderCode": "F"
        }
    ]
}
```

次のサンプル応答では、ルックアップ値のコードが示されています。

```
{
    "item": [
        {
            "id": "F",
            "label": "FEMALE"
        },
        {
            "id": "M",
            "label": "MALE"
        },
        {
            "id": "N",
            "label": "UNKNOWN"
        }
    ],
    "firstRecord": 1,
    "recordCount": 0,
    "pageSize": 2147483647,
```

```
    "searchToken": "SVR1.AU5LD"
}
```

検索レコード

［レコードの検索］REST API は、検索可能なルートレコード内とすべての子レコード内でインデックス付きの値を検索します。フィルタおよびファセットを使用して、検索結果のサブセットを表示できます。ファセットは検索結果をカテゴリにグループ化し、フィルタは検索結果を絞り込みます。この API は、検索可能と設定されているフィールドをすべて検索し、検索条件に一致するレコードを返します。

この API は、GET メソッドを使用して検索可能フィールドのインデックスを検索します。

要求 URL

［レコードの検索］URL の形式は次のとおりです。

基本検索の URL

基本検索の場合は次の URL を使用します。

```
http://<host>:<port>/<context>/<database ID>/<business entity>?q=<string value>
```

［レコードの検索］URL に対して、次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/<context>/<database ID>/<business entity>?q=<string value>
```

フィールド検索の URL

フィールド検索の場合は次の URL を使用します。

```
http://<host>:<port>/<context>/<database ID>/<business entity>?fq=<business entity field name>='<business entity field value>'
```

ビジネスエンティティフィールドに-120 などの負の数値を指定すると、返されるレコードのランキングが影響を受ける可能性があります。

ファセット検索の URL

ファセットを使用した基本検索の場合は次の URL を使用します。

```
http://<host>:<port>/<context>/<database ID>/<business entity>?q=<string value>&facets=<field name>
```

ファセットを使用したフィールド検索の場合は次の URL を使用します。

```
http://<host>:<port>/<context>/<database ID>/<business entity>?fq=<business entity field name>='<business entity field value>'&facets=<field name>
```

フィルタ検索の URL

フィルタを使用した基本検索の場合は次の URL を使用します。

```
http://<host>:<port>/<context>/<database ID>/<business entity>?q=<string value>&filters=<field name1>=<field value1> AND <field name2>=<field value2> ...
```

検索で q または fq パラメータを使用します。

URL **エンコード**

URL にはスペースや一重引用符などの文字が含まれるため、URL エンコードを使用します。

次の例は、URL エンコードが行われた［レコードの検索］URL を示しています。

```
http://<host>:<port>/<context>/<database ID>/<business entity>?q=<field name>%3D%27<value of the field>
```

クエリパラメータ

q または fq クエリパラメータは、検索の文字列値を指定するために使用します。q クエリパラメータと fq クエリパラメータは相互に排他的です。fq クエリパラメータは、フィールド検索に使用します。複数の条件がある場合は、AND 論理演算子を使用します。

次の表に、URL に使用できるパラメータを示します。

パラメータ	説明
q	文字列値または検索用語を指定します。このクエリでは、レコード内の検索用語の出現箇所が検索されます。基本検索で使用します。 例えば、Person?q=STEVE クエリでは、STEVE という用語のあるレコードが検索されます。 2 つ以上の語句を一緒に検索するには、二重引用符にそれらの語句を含めます。検索結果に特定の語句が含まれるようにするには、各語句の前に文字「+」を入れます。フィールド値にスペースが含まれている場合、フィールド値を一重引用符で囲みます。 WILLIAM JOHN LAWSON と完全に一致するものを検索するには、次のクエリを使用します。 Person?q="WILLIAM JOHN LAWSON" WILLIAM、JOHN、または LAWSON を検索するには、次のクエリを使用します。 Person?q=WILLIAM JOHN LAWSON WILLIAM、JOHN、および LAWSON を検索するには、次のクエリを使用します。 Person?q=WILLIAM JOHN LAWSON&queryOperator=AND 検索用語にアンパサンド（&）特殊文字が含まれている場合は、その文字を%26としてエンコードします。そうしない場合、検索要求は期待される結果を返しません。 次のクエリを使用して、Johnson&Johnson を検索します。 Organization?q=Johnson%26Johnson&queryOperator=AND
fq	特定のフィールドの文字列値または検索用語を指定します。このクエリでは、レコードの特定部分の用語のみが検索されます。インデックス付きフィールドに基づいて検索対象を絞り込む場合に使用します。 例えば、Person?fq=displayname=STEVE クエリでは、表示名が STEVE のレコードが検索されます。 検索用語にアンパサンド（&）特殊文字が含まれている場合は、その文字を%26としてエンコードします。そうしない場合、検索要求は期待される結果を返しません。 例えば、次のクエリを使用して、Mac&Cheese を検索します。 Product?fq=fname=Mac%26Cheese&queryOperator=AND
ファセット	検索結果のグループ化の基準となるファセットまたはカテゴリとして扱われるフィールドを指定します。検索可能フィールドのみを指定します。q および fq パラメータとともに使用します。構文は、&facets=FieldName1,FieldName2,FieldNameN です。 例えば、Person?q=STEVE&facets=department クエリでは、表示名が STEVE の人物が検索され、検索結果が部門でグループ化されます。この検索では、表示名が STEVE の人物のレコードが表示され、これらのレコードが部門でグループ化されます。

パラメータ	説明
フィルタ	検索結果を絞り込むことができるフィールドを指定します。フィルタ可能フィールドのみを指定します。q および fq パラメータとともに使用します。 例えば、 `Person?fq=displayname=STEVE&filters=birthdate='1980-11-27T08:00:00Z'` クエリでは、表示名が STEVE の人物が検索され、検索結果が生年月日でフィルタリングされます。この検索では、表示名が STEVE で誕生日が 1980 年 11 月 27 日の人物のレコードが表示されます。 **注**: 日付は、一重引用符で囲んで指定します。
depth	返す子レベルの数を指定します。ルートノードとその直接の子を返すには 2 を指定し、ルートノード、直接の子、および孫を返すには 3 を指定します。ルールノードのみを返すには、1 を指定します。デフォルトでは、depth は指定されません。 depth を指定しないと、ルートノードと検索条件と一致する子のみが検索結果として返されます。 例えば、`Person?q=STEVE&depth=2` クエリでは、STEVE という用語のあるレコードが検索され、ルートレコードとその直接の子に関する情報が返されます。
queryOperator	検索用語内の文字列のいずれかまたはそのすべてを検索するかどうかを指定します。 パラメータは次のいずれかの値を使用します。 - `OR`。f または fq パラメータに表示されたいずれかの文字列を検索します。 - `AND`。f または fq パラメータに表示されたすべての文字列を検索します。 このパラメータを指定しない場合、デフォルトは `OR` です。 例えば、`Person?q=WILLIAM JOHN LAWSON&queryOperator=AND` クエリは、WILLIAM、JOHN、および LAWSON が含まれるレコードを検索します。
suppressLinks	API 応答に親と子のリンクを含めるかどうかを示します。応答に親と子のリンクをいっさい含めない場合は、このパラメータを true に設定します。デフォルトは false です。 例えば、`Person?q=STEVE&suppressLinks=true` クエリは、用語 STEVE が含まれているレコードを検索して、親と子のリンクを含まない応答を返します。
readSystemFields	結果でシステムフィールドを返すかどうかを示します。デフォルトは false です。
order	フィールド名のカンマ区切りリスト。オプションでプレフィックス+または-を指定できます。プレフィックス+は結果を昇順でソートし、プレフィックス-は結果を降順でソートすることを指定します。デフォルトは+です。 子フィールドを使用して結果をソートする場合は、フィールドの完全な名前を使用します。例えば、`BillAddresses.Address.cityName` を使用します。 複数のパラメータを指定した場合、結果セットは、指定した最初のパラメータでまずソートされてから、次のパラメータでソートされます。例えば、`Person?order=displayName,-BillAddresses.Address.cityName` クエリは、表示名で昇順に結果をソートしてから、都市名で降順にソートします。
maxRecordsToSort	ソートする検索結果の最大数。デフォルトは 1000 です。
resolveLookup	指定されたビジネスエンティティのルックアップフィールド全体を取得します。パラメータを true に設定して、ルックアップフィールドをロードし、応答に含めます。デフォルトは false です。

filters **パラメータでの範囲の指定**:
filters パラメータを使用して、特定の範囲内に検索結果を絞り込むことができます。数値データ型および日付データ型のフィルタ可能フィールドの範囲を指定できます。

整数データ型では、次の形式を使用します。

```
fieldName1=[fromValue,toValue]
```

範囲は、fromValue から toValue です。fromValue が toValue よりも小さいことを確認してください。 例えば、`filters=age=[35,45]`クエリでは、検索結果が絞り込まれて、35～45 歳の年齢層のレコードが検索されます。

日付データ型では、次の形式を使用します。

```
fieldName1=[fromDate,toDate]
```

範囲は、fromDate から toDate です。例えば、`filters=birthdate=[2000-06-12T12:30:00Z,2015-06-12T12:30:00Z]`クエリでは、2000 年 6 月 12 日～2015 年 6 月 12 日の誕生日が指定されます。

注: 完全一致の日付フィルタを指定する場合、一重引用符で囲みます。日付範囲を指定する場合は、引用符を使用しないでください。

関連項目：

- 「UTC での日付と時刻の形式」 (ページ 34)

サンプル API 要求

q パラメータを使用した要求

次のサンプル要求は、Person ビジネスエンティティから STEVE という名前のレコードを検索します。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person?q=STEVE
```

fq パラメータを使用した要求

次のサンプル要求は、Person ビジネスエンティティから表示名が STEVE のレコードを検索します。displayName フィールドはインデックス付きフィールドです。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person?fq=displayName=STEVE
```

ソートオプションのある要求

次のサンプル要求は、Person ビジネスエンティティから表示名が STEVE のレコードを検索し、結果を昇順でソートします。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person?
fq=displayName=STEVE&order=BillAddresses.Address.cityName
```

fq パラメータと AND 論理演算子を使用した要求

次のサンプル要求は、Person ビジネスエンティティから表示名が STEVE で税金 ID が DM106 のレコードを検索します。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person?fq=displayName=STEVE AND taxId=DM106
```

ファセットを使用した要求

次のサンプル要求は、Person ビジネスエンティティから表示名が STEVE のレコードを検索し、部門にグループ化して結果を絞り込みます。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person?fq=displayName=STEVE&facets=department
```

フィルタ（完全一致のフィルタ）を使用した要求

次のサンプル要求は、Person ビジネスエンティティから表示名が STEVE のレコードを検索し、指定された都市や国でフィルタリングします。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person?fq=displayName=STEVE&filters=cityName=Canberra
AND country=Australia
```

フィルタ範囲を使用した要求

次のサンプル要求は、Person ビジネスエンティティから表示名が STEVE のレコードを検索し、35〜45 歳の年齢層でフィルタリングします。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person?fq=displayName=STEVE&filters=age=[35,45] AND
cityName=Canberra
```

サンプル API 応答

次のサンプル応答では、STEVE という名前での検索結果が示されています。


```
{
    "firstRecord": 1,
    "recordCount": 2,
    "pageSize": 10,
    "item": [
        {
            "Person": {
                "link": [
                    {
                        "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/1443",
                        "rel": "self"
                    },
                    {
                        "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/1443?depth=2",
                        "rel": "children"
                    }
                ],
                "rowidObject": "1443          ",
                "label": "CRAIG,STEVE",
                "partyType": "Person",
                "lastName": "CRAIG",
                "firstName": "STEVE",
                "taxID": "stevecraig                    ",
                "displayName": "STEVE CRAIG"
            }
        },
        {
            "Person": {
                "link": [
                    {
                        "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/285",
                        "rel": "self"
                    },
                    {
                        "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/285?depth=2",
                        "rel": "children"
                    }
                ],
                "rowidObject": "285           ",
                "label": "PEARSON,STEVE",
                "partyType": "Person",
                "lastName": "PEARSON",
                "firstName": "STEVE",
                "displayName": "STEVE PEARSON"
            }
        }
    ]
}
```


提案元

［提案元］REST API は、データベースに存在するデータに基づいて、検索文字列の関連用語のリストを返します。この API を使用して、ユーザーインタフェースのフィールドに入力した文字を受け入れ、入力内容をオー

トコンプリートする提案を返します。提案のリストから文字列を見つけて選択できます。［提案元］API は、検索可能フィールドに使用します。

この API は GET メソッドを使用します。

注: API を使用して、検索可能フィールドにオートコンプリートの提案のリストを表示するには、フィールドの suggester プロパティを true に設定し、データの再インデックス処理を行います。

要求 URL

［提案元］REST URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/<business entity>?suggest=<string>
```

この URL に対して次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/<context>/<database ID>/<business entity>?suggest=<string>
```

クエリパラメータ

次の表にクエリパラメータを示します。

パラメータ	説明
suggest	必須。提案を行う文字列を指定します。
recordsToReturn	返す行の数を指定します。
buildIndex	オプション。インデックスを構築するかどうかを示します。システムを再起動する場合、このパラメータを true に設定して明示的にインデックスを構築し、収集を行います。このパラメータは、今後のリリースで廃止される可能性があります。

サンプル API 要求

次のサンプル要求は、ユーザーインタフェースで使用できる提案のリストを返します。

```
GET http://localhost:8080/cmx/cs/localhost-infa-DS_UI1/Person.json?suggest=Abhinav
```

サンプル API 応答

次のサンプル応答では、提案のリストが示されています。

```
{
term: [2]
"abhinav goel"
"abhinav gupta"

}
```

SearchQuery

SearchQuery REST API は、クエリに指定されたフィールド値の完全一致のレコードを検索します。SearchQuery API を使用して、特定のビジネスエンティティのすべてのレコードを取得するか、特定のフィールド値に基づいたレコードを取得できます。指定された値を検索可能なすべてのフィールドに渡って検索するレコード検索 API とは異なり、SearchQuery API は、指定された値を特定のフィールド内で検索します。

クエリ結果をフィルタして、ルートビジネスエンティティレコードと子レコード内の特定の値を表示できます。クエリでは、AND、IN、および RANGE の演算子を使用できます。

クエリ結果に特定のフィールドを含めるには、フィールドを指定するか、フィールドが含まれるビジネスエンティティビューを指定します。クエリ結果をソートして、昇順または降順でレコードを表示できます。

この API は GET メソッドを使用します。

要求 URL

SearchQuery URL のコンテキストは、cmx/cs です。ホストされた MDM 環境では、<テナント名>/cmx/cs のように、コンテキストにテナント名を含めます。

SearchQuery URL の形式は次のとおりです。

特定のビジネスエンティティタイプのすべてのレコードを返す URL

次の URL を使用して、指定したビジネスエンティティタイプのすべてのレコードを検索します。

```
http://<host>:<port>/<context>/<database ID>/<business entity>?action=query
```

SearchQuery URL に対して次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/<context>/<database ID>/<business entity>?action=query
```

指定したフィールド値に一致するレコードのすべての詳細を返す URL

次の URL を使用して、指定したフィールド値に一致するレコードを検索します。

```
http://<host>:<port>/<context>/<database ID>/<business entity>?action=query&filter=<business entity field name 1>='<business entity field value 1>' AND <business entity field name 2>='<business entity field value 2>'...AND <business entity field name n>='<business entity field value n>'
```

指定したフィールド値に一致するレコードの特定の詳細を返す URL

次の URL を使用してレコードを検索し、検索結果に特定のレコードフィールドを表示します。

```
http://<host>:<port>/<context>/<database ID>/<business entity>?action=query&filter=<business entity field name 1>='<business entity field value 1>' AND <business entity field name 2>='<business entity field value 2>'...AND <business entity field name n>='<business entity field value n>'&outputView=<business entity view>
```

クエリパラメータ

フィールドと値のペアのリストとして、クエリを定義します。

次の表に、URL に使用できるクエリパラメータを示します。

パラメータ	説明
action	必須。クエリ結果に、指定したビジネスエンティティタイプのすべてのレコードを返します。query に設定し、filter パラメータを設定してこのパラメータを使用します。filter パラメータを指定せずに使用すると、指定したビジネスエンティティタイプのすべてのレコードがクエリによって検索されます。 例えば、次のクエリを使用すると、すべての Person ビジネスエンティティレコードが検索されます。 `Person?action=query`
filter	必須。演算子で区切られた、フィールド値ペアのリストを指定します。有効な演算子は AND、IN、および Range です。 例えば、次のクエリを使用すると、名が STEVE で姓が SMITH の Person レコードが検索されます。 `Person?action=query&filter=firstName='STEVE' AND lastName='SMITH'`

パラメータ	説明
depth	返す子レコードレベルの数を指定します。例えば、以下のレベルを指定できます。 - 1. ルートレコードを返します。 - 2. ルートレコードと、その直接の子レコードを返します。 - 3. ルートレコード、直接の子レコード、および孫レコードを返します。 例えば、次のクエリを使用すると、名が STEVE のレコードが検索され、ルートレコードとその直接の子レコードに関する情報が返されます。 `Person?action=query&filter=firstName='STEVE' AND lastName='SMITH'&depth=2`
suppressLinks	API 応答に親と子のリンクを含めるかどうかを示します。応答に親と子のリンクをいっさい含めない場合は、このパラメータを true に設定します。デフォルトは false です。 例えば、次のクエリを使用すると、名が STEVE のレコードが検索され、親と子のリンクが表示されない応答が返されます。 `Person?action=query&filter=firstName='STEVE'&suppressLinks=true`
readSystemFields	結果でシステムフィールドを返すかどうかを示します。デフォルトは false です。
fields	クエリ結果に表示するフィールドを指定します。
outputView	クエリ結果を表示するために使用するビジネスエンティティビューを指定します。クエリ結果のビジネスエンティティビューを設定する場合は、クエリ結果に表示するフィールドを含めます。
順序	クエリ結果のソート順を指定します。プラス（+）文字をプレフィックスとして使用すると昇順、マイナス（-）文字をプレフィックスとして使用すると降順でソートされます。デフォルトでは、クエリ結果は昇順になります。 複数のパラメータを指定した場合、結果セットは、指定した最初のパラメータでまずソートされてから、次のパラメータでソートされます。

resolveLookup	指定されたビジネスエンティティのルックアップフィールド全体を取得します。パラメータを true に設定して、ルックアップフィールドをロードし、応答に含めます。デフォルトは false です。
	例えば、addressType フィールドは、Person ビジネスエンティティの子レベルのルックアップフィールドです。
	resolveLookup パラメータが false に設定されている場合、次の REST API 応答を受信する可能性があります。
	`{ "label": "LU Address Type", "addressType": "BILL" }`
	resolveLookups パラメータが true に設定されている場合、REST API 応答には追加の詳細が含まれ、次の REST API 応答を受信する可能性があります。
	`{ "label": "LU Address Type", "addressType": "BILL", "addressTypeDisp": "BILLING" }`

演算子

フィルタパラメータ内で次の演算子を使用できます。

AND

フィルタパラメータのリストに含まれるすべてのフィールド値を持つレコードが検索されます。

例えば、次のクエリを使用すると、名が STEVE で姓が SMITH のレコードが検索されます。

```
Person?action=query&filter=firstName='STEVE' AND lastName='SMITH'
```

IN

フィルタパラメータのリストに含まれる値のいずれかを持つレコードが検索されます。

例えば、次のクエリを使用すると、名が STEVE または JOHN のレコードが検索されます。

```
Person?action=query&filter=firstName IN [STEVE,JOHN]
```

範囲

指定した範囲内のレコードが検索されます。数値および日付のデータ型のフィールドに範囲を指定できます。

整数データ型では、次の形式を使用します。

```
<business entity field name>=[fromValue,toValue]
```

範囲は、fromValue から toValue です。fromValue が toValue よりも小さいことを確認してください。

例えば、次のクエリを使用すると、年齢グループが 35 から 45 のレコードが検索されます。

```
Person?action=query&filter=firstName IN [STEVE,JOHN] AND age=[35,45]
```

日付データ型では、次の形式を使用します。

```
<business entity field name>=[fromDate,toDate]
```

範囲は、fromDate から toDate です。

例えば、次のクエリを使用すると、誕生日が 2000 年 6 月 12 日から 2015 年 6 月 12 日のレコードが検索されます。

```
Person?action=query&filter=birthDate=[2000-06-12T12:30:00Z,2015-06-12T12:30:00Z]
```

! =

指定されたフィールド値または範囲に一致しないレコードを検索します。

例えば、次のクエリを使用すると、名が ADAM 以外のレコードが検索されます。

```
Person?action=query&filter=firstName!='ADAM'
```

例えば、次のクエリを使用すると、名が ADAM 以外で、生年月日が 2017 年 11 月 16 日より前と 2020 年 11 月 16 日より後のレコードが検索されます

```
Person?action=query&filter=firstName!='ADAM' AND birthdate!=[2017-11-16T00:00:00,2020-11-16T00:00:00]
```

ワイルドカード

完全な検索文字列の代わりに、テキストとアスタリスク（*）ワイルドカード演算子を使用してテキストパターンを指定できます。アスタリスクワイルドカード演算子を使用すると、目的のレコードが見つかる可能性が高くなります。アスタリスクワイルドカード演算子は、正確に一致するテキストがわからない場合や、類似テキストを検索する場合に便利です。

次の表に、検索文字列の例とそれぞれの働きについて説明します。

クエリ文字列の例	クエリの動作
John*	John で始まる値が含まれるレコードをクエリします。例えば、Johnson や Johnny が該当します。
Jo*n	Jo で始まり n で終わるレコードをクエリします。例えば、Johansson や Jordan が該当します。
*	すべてのレコードを返します。

サンプル API 要求

このトピックでは、サンプル要求クエリについて説明します。

次のサンプル要求は、名が STEVE で姓が SMITH のレコードを、Person ビジネスエンティティにクエリで問い合わせます。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person?action=query&filter=firstName='STEVE' AND
lastName='SMITH'&outputView=PersonAddressView
```

次のサンプル要求は、対応する元の行 ID を持つレコードを、Person ビジネスエンティティにクエリで問い合わせます。この例では、この Person の一意の識別子 ROWID_OBJECT として使用される値「1」に基づいて、1 つのレコードのみが返されます。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person?
action=query&filter=origRowidObject='1'&outputView=PersonAddressView
```

次のサンプル要求は、対応する行 ID を持つ存続レコードを、Person ビジネスエンティティにクエリで問い合わせます。この例では、この Person の一意の識別子 ROWID_OBJECT として使用される値「1」に基づいて、1 つのレコードのみが返されます。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person?
action=query&filter=rowidObject='1'&outputView=PersonAddressView
```

サンプル API 応答

次のサンプル応答は、名が STEVE で姓が SMITH の Person レコードに対するクエリ結果を示します。

```
{
  "link": [],
  "firstRecord": 1,
  "pageSize": 10,
  "searchToken": "SVR1.1T8UU",
  "facet": [],
  "item": [
    {
      "Person": {
        "rowidObject": "268           ",
        "label": "Person: SMITH, STEVE,268           ",
        "partyType": "Person",
        "lastName": "SMITH",
        "displayName": "STEVE SMITH",
        "firstName": "STEVE"
      }
    },
    {
      "Person": {
        "rowidObject": "448           ",
        "label": "Person: SMITH, STEVE,448           ",
        "partyType": "Person",
        "lastName": "SMITH",
        "displayName": "STEVE SMITH",
```

```
        "firstName": "STEVE"
      }
    }
  ]
}
```

SearchQuery 結果の CSV ファイルへのエクスポート

SearchQuery 要求の結果を CSV ファイルにエクスポートするには、要求 URL のパスで、ビジネスエンティティの名前を.CSV ファイルとして指定します。要求 URL では、すべてのクエリパラメータを使用できます。

例えば、次の要求 URL を使用して、指定したフィールド値に一致するレコードの検索結果をエクスポートします。

```
http://<host>:<port>/<context>/<database ID>/<business entity>.CSV?action=query&filter=<business entity field name 1>='<business entity field value 1>' AND <business entity field name 2>='<business entity field value 2>'...AND <business entity field name n>='<business entity field value n>'
```

サンプル API 要求

次のサンプル要求は、名 STEVE と姓 SMITH のレコードをクエリし、クエリ結果を CSV 形式で返します。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person.CSV?action=query&filter=firstName='STEVE' AND lastName='SMITH'&fields=recordID,firstName,lastName
```

サンプル API 応答

次のサンプルは、名 STEVE と姓 SMITH のクエリに対するクエリ結果を、CSV 形式で示します。

```
recordID,firstName,lastName
00023,Steve,Smith
00048,Steve,Smith
```

SearchMatch

SearchMatch REST API は、あいまい一致のレコードを検索します。この API は、MDM Hub で一致カラムとして設定された特定のビジネスエンティティフィールドに基づいて、一致レコードを識別します。SearchMatch API を使用する前に、［一致カラムのリスト］API を使用して、ビジネスエンティティに対する一致カラムを識別します。

必要に応じて、一致カラムの代わりに一致ルールセットを指定できます。クエリに一致ルールセットを使用するには、一致ルールセットで［ルールによる検索を有効にする］オプションが有効になっていることを確認します。一致カラムと一致ルールセットの設定の詳細については、*Multidomain MDM の設定ガイド*を参照してください。

クエリでは、AND、IN、および RANGE の演算子を使用できます。

クエリ結果に特定のフィールドを含めるには、フィールドを指定するか、フィールドが含まれるビジネスエンティティビューを指定します。クエリ結果の表示でレコードを昇順にソートするか降順にソートするかを指定できます。

API は、GET メソッドを使用して、ビジネスエンティティフィールドを問い合わせ、あいまい一致のレコードと一緒に、それぞれの一致スコアと関連する一致ルールを返します。

要求 URL

SearchMatch URL のコンテキストは、cmx/cs です。ホストされた MDM 環境では、<テナント名>/cmx/cs のように、コンテキストにテナント名を含めます。

SearchMatch URL の形式は次のとおりです。

一致カラムとして設定されている特定のフィールドの値に基づいて、一致したレコードを返す URL

次の URL を使用して、指定したフィールド値に一致するレコードを検索します。

```
http://<host>:<port>/<context>/<database ID>/<business entity>?action=match&fuzzyFilter=<business entity field name 1>='<business entity field value 1>',<business entity field name 2>='<business entity field value 2>',...<business entity field name n>='<business entity field value n>'
```

SearchMatch URL に対して次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/<context>/<database ID>/<business entity>?action=match&fuzzyFilter=<business entity field name 1>='<business entity field value 1>',<business entity field name 2>='<business entity field value 2>',...<business entity field name n>='<business entity field value n>'
```

一致ルールセットに基づいて一致したレコードを返す URL

次の URL を使用して、指定した一致ルールセットに基づいて、一致したレコードを検索します。

```
http://<host>:<port>/<context>/<database ID>/<business entity>?action=match&fuzzyFilter=<business entity field name 1>='<business entity field value 1>',<business entity field name 2>='<business entity field value 2>',...<business entity field name n>='<business entity field value n>'&matchRuleSet=<match rule set name>
```

クエリパラメータ

fuzzyFilter パラメータを使用して、クエリするフィールド値を指定します。fuzzyFilter パラメータを action パラメータと一緒に使用します。

次の表に、URL に使用できるクエリパラメータを示します。

パラメータ	説明
action	必須。指定したビジネスエンティティに一致するレコードを返します。match に設定し、fuzzyFilter パラメータを設定してこのパラメータを使用します。 例えば、次のクエリを使用すると、名が STEVE の個人が検索されます。 `Person?action=match&fuzzyFilter=STEVE`
fuzzyFilter	必須。特定のビジネスエンティティタイプのレコードに対するクエリで使用する、フィールド名とフィールド値のペアの、カンマ区切りリストを指定します。 例えば、次のクエリを使用すると、名が STEVE で住所が Toronto のレコードが検索されます。 `Person?action=match&fuzzyFilter=firstName=STEVE,Address.Address.City=TORONTO`
matchRuleSet	一致したレコードを識別するために使用する、一致ルールセットを指定します。 特定の一致ルールセットがない場合は、[NONE] を指定します。自動および手動マージの一致ルールが使用されます。
filter	あいまい検索の結果のフィルタに使用するフィールド値を指定します。 例えば、次のクエリを使用すると、Toronto に住んでいて名が STEVE のレコードが検索されます。 `Person?action=match&fuzzyFilter=firstName='STEVE',lastName="SMITH'&filter=city=Toronto`
depth	返す子レコードレベルの数を指定します。例えば、以下のレベルを指定できます。 - 1. ルートレコードを返します。 - 2. ルートレコードと、その直接の子レコードを返します。 - 3. ルートレコード、直接の子レコード、および孫レコードを返します。 例えば、次のクエリを使用すると、名が STEVE のレコードが検索され、ルートレコードとその直接の子レコードに関する情報が返されます。 `Person?action=match&fuzzyFilter=firstName='STEVE'&filter=city=Toronto`

パラメータ	説明
suppressLinks	API 応答に親と子のリンクを含めるかどうかを示します。応答に親と子のリンクをいっさい含めない場合は、このパラメータを true に設定します。デフォルトは false です。 例えば、次のクエリを使用すると、名が STEVE のレコードが検索され、親と子のリンクが表示されない応答が返されます。 `Person?action=match&fuzzyFilter=firstName='STEVE'&suppressLinks=true`
readSystemFields	結果でシステムフィールドを返すかどうかを示します。デフォルトは false です。
fields	クエリ結果に表示するフィールドを指定します。
outputView	クエリ結果を表示するために使用するビジネスエンティティビューを指定します。クエリ結果のビジネスエンティティビューを設定する場合は、クエリ結果に表示するフィールドを含めます。

resolveLookup	指定されたビジネスエンティティのルックアップフィールド全体を取得します。パラメータを true に設定して、ルックアップフィールドをロードし、応答に含めます。デフォルトは false です。
	例えば、addressType フィールドは、Person ビジネスエンティティの子レベルのルックアップフィールドです。
	resolveLookup パラメータが false に設定されている場合、次の REST API 応答を受信する可能性があります。
	`{ "label": "LU Address Type", "addressType": "BILL" }`
	resolveLookups パラメータが true に設定されている場合、REST API 応答には追加の詳細が含まれ、次の REST API 応答を受信する可能性があります。
	`{ "label": "LU Address Type", "addressType": "BILL", "addressTypeDisp": "BILLING" }`

フィルタパラメータ内で次の演算子を使用できます。

AND

フィルタパラメータのリストに含まれるすべてのフィールド値を持つレコードが検索されます。

例えば、次のクエリを使用すると、名が STEVE で姓が SMITH のレコードが検索されます。

`Person?action=match&fuzzyFilter=firstName='STEVE',lastName='SMITH'&filter=city=Toronto AND gender=Male`

IN

フィルタパラメータのリストに含まれる値のいずれかを持つレコードが検索されます。

例えば、次のクエリを使用すると、Toronto または Ottawa の都市に住んでいて、名が STEVE または姓が JOHN のレコードが検索されます。

`Person?action=match&fuzzyFilter=firstName='STEVE',lastName='SMITH'&filter=city in [Toronto,Ottawa]`

範囲

指定した範囲内のレコードが検索されます。数値および日付のデータ型のフィールドに範囲を指定できます。

整数データ型では、次の形式を使用します。

```
<business entity field name>=[fromValue,toValue]
```

範囲は、fromValue から toValue です。fromValue が toValue よりも小さいことを確認してください。

例えば、次のクエリを使用すると、年齢グループが 35 から 45 のレコードが検索されます。

```
Person?action=match&fuzzyFilter=firstName='STEVE',lastName='SMITH'&filter=age=[35,45]
```

日付データ型では、次の形式を使用します。

```
<business entity field name>=[fromDate,toDate]
```

範囲は、fromDate から toDate です。

例えば、次のクエリを使用すると、誕生日が 2000 年 6 月 12 日から 2015 年 6 月 12 日のレコードが検索されます。

```
Person?action=match&fuzzyFilter=firstName='STEVE',lastName='SMITH'&filter=birthDate=[2000-06-12T12:30:00Z,
2015-06-12T12:30:00Z]
```

サンプル API 要求

次のサンプル要求は、一致ルールセット IDL2 を使用して、名が STEVE のレコードを、Person ビジネスエンティティにクエリで問い合わせます。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person?
action=match&fuzzyFilter=firstName=STEVE&matchRuleSet=IDL2
```

サンプル API 応答

次のサンプル応答は、一致ルールセット IDL2 に基づいて、名が STEVE のレコードに対するクエリ結果を示します。

```
{
  "link": [],
  "firstRecord": 1,
  "recordCount": 3,
  "pageSize": 10,
  "searchToken": "SVR1.17LJ2",
  "matchedEntity": [
    {
      "businessEntity": {
        "Person": {
          "rowidObject": "145           ",
          "label": "SAMUEL,STEVE",
          "partyType": "Person",
          "lastName": "SAMUEL",
          "displayName": "MR STEVE SAMUEL                                    ",
          "statusCd": "A ",
          "firstName": "STEVE",
          "genderCd": {
            "genderCode": "M"
          },
          "namePrefixCd": {
            "namePrefixCode": "MR"
          }
        }
      },
      "matchRule": "IDL2|1",
     "matchScore": "93",
      "link": [
        {
```

```
          "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI2/meta/matchRule/Person/IDL2|1.json",
          "rel": "matchRule"
        }
      ]
    },

    {
      "businessEntity": {
        "Person": {
          "rowidObject": "268            ",
          "label": "SMITH,STEVE",
          "partyType": "Person",
          "lastName": "SMITH",
          "displayName": "STEVE SMITH",
          "firstName": "SAM"
        }
      },
      "matchRule": "IDL2|1",
      "matchScore": "98",
      "link": [
        {
          "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI2/meta/matchRule/Person/IDL2|1.json",
          "rel": "matchRule"
        }
      ]
    },
    {
      "businessEntity": {
        "Person": {
          "rowidObject": "448            ",
          "label": "SMITH,STEVEN",
          "partyType": "Person",
          "lastName": "SMITH",
          "displayName": "SAM STEVEN",
          "firstName": "STEVEN"
        }
      },
      "matchRule": "IDL2|1",
      "matchScore": "98",
      "link": [
        {
          "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI2/meta/matchRule/Person/IDL2|1.json",
          "rel": "matchRule"
        }
      ]
    }

  ],
  "facet": []
}
```


SearchMatch 結果の CSV ファイルへのエクスポート

SearchMatch 要求の結果を CSV ファイルとしてエクスポートするには、要求 URL のパスで、ビジネスエンティティの名前を.CSV ファイルとして指定します。要求 URL では、すべてのクエリパラメータを使用できます。

例えば、次の要求 URL を使用して、一致ルールセットに基づいた一致レコードの検索結果をエクスポートします。

```
http://<host>:<port>/<context>/<database ID>/<business entity>.CSV?action=match&fuzzyFilter=<business entity field name 1>='<business entity field value 1>',<business entity field name 2>='<business entity field value 2>',...<business entity field name n>='<business entity field value n>'&matchRuleSet=<match rule set name>
```

サンプル API 要求

次のサンプル要求は、名 STEVE と姓 SMITH に一致するレコードを検索し、クエリ結果を CSV 形式で返します。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person.CSV?
action=match&fuzzyFilter=firstName=STEVE,lastName=SMITH&fields=recordID,firstName,lastName
```

サンプル API 応答

次のサンプルは、名 STEVE と姓 SMITH に一致するレコードのクエリ結果を、CSV 形式で表します。

```
recordID,firstName,lastName
00023,Steve,Smith
00035,Steven,Smith
00048,Steve,Smith
00079,Steve,Smithson
```

BPM メタデータの取得

［BPM メタデータの取得］REST API は、タスクタイプ、および BPM ワークフローツールが Get Task Lineage サービスと管理サービスを実行できるかどうかを指定する 2 つのインジケータを返します。

この API は GET メソッドを使用します。

要求 URL

［BPM メタデータの取得］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/BPMMetadata
```

［BPM メタデータの取得］URL に対して、次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/<context>/<database ID>/BPMMetadata
```

サンプル API 要求

次のサンプル要求は、タスクタイプと BPM ワークフローツールについての情報を返します。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/BPMMetadata
```

サンプル API 応答

次のサンプル応答では、タスクタイプと、BPM ワークフローツールの 2 つのインジケータの値が示されています。

```
{
    "parameters": {
        "doesSupportAdministration": true,
        "doesSupportLineage": true,
        "doesSupportAttachments": true,
        "maximumAttachmentFileSizeInMb": 20,
        "taskTypes": {
            "taskTypes": [
                {
                    "name": "Merge",
                    "label": "Merge"
                },
                {
                    "name": "FinalReview",
                    "label": "FinalReview"
                },
                {
                    "name": "Update",
                    "label": "Update"
                },
```

```
                {
                    "name": "Notification",
                    "label": "Notification"
                },
                {
                    "name": "ReviewNoApprove",
                    "label": "ReviewNoApprove"
                },
                {
                    "name": "Unmerge",
                    "label": "Unmerge"
                }
            ]
        }
    }
}
```

タスクの一覧表示

［タスクの一覧表示］API は、ワークフロータスクのリストを返します。ワークフローは、ビジネスプロセスにおけるアクティビティ、およびアクティビティを通して実行するパスを定義します。各アクティビティはタスクと呼ばれます。

この API は、格納されているタスクのリストとページ化されているタスクのリストを、GET メソッドを使用して返します。

要求 URL

［タスクの一覧表示］URL の形式は次のとおりです。

`http://<host>:<port>/<context>/<database ID>/task`

［タスクの一覧表示］URL に対して、次の HTTP GET 要求を行います。

`GET http://<host>:<port>/<context>/<database ID>/task`

要求には HTTP ヘッダーを追加できます。

クエリパラメータ

タスクのリストをフィルタリングするには、タスクデータフィールドをクエリパラメータとして使用します。

以下のクエリパラメータを使用できます。

パラメータ	説明
taskType	レコードで実行できる一連のアクションです。タスクタイプの指定には名前属性を使用します。タスクタイプの詳細については、『*Multidomain MDM Data Director の実装ガイド*』を参照してください。
taskId	タスクの ID。
processId	タスクを含むワークフロープロセスの ID。
owner	タスクを実行するユーザー。
title	タスクの簡単な説明。

パラメータ	説明
status	ワークフロー内のタスクの状態。以下の 2 つの値のいずれかを使用します。 - Open: タスクがまだ開始されていないか、または進捗中である。 - Closed: タスクが完了しているか、またはキャンセルされている。
priority	タスクの重要度。high、normal、low のうちのいずれかの値を使用します。
creator	タスクを作成するユーザー。
createDateBefore と createDateAfter	日付範囲。タスクは createDate フィールドでフィルタリングできます。
dueDateBefore と dueDateAfter	日付範囲。タスクは dueDate フィールドでフィルタリングできます。
scope	ユーザーロールに基づいて利用可能なタスクを表示します。admin 値を使用して、ロールに基づいてタスクのリストをフィルタリングします。 **注**: タスク管理者ロールを割り当てられたユーザーはすべてのタスクを管理できますが、他のユーザーロールを割り当てられたユーザーは自分のロールで使用可能なタスクのみを管理できます。
q	検索語句を指定します。クエリは、ビジネスエンティティレコードに関連するタスク内の検索語句の出現を検索します。 例えば、クエリ q=KAREN は、検索語句 KAREN を含むタスクを検索します。

クエリパラメータは要求 URL で名前と値のペアとして使用します。

次のサンプルは、クエリパラメータを使用してタスクをフィルタリングする方法を示しています。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/task?
scope=admin&recordsToReturn=100&owner=sergey&status=OPEN&q=KAREN
```

関連項目：

- 「UTC での日付と時刻の形式」 (ページ 34)

パラメータのソート

REST API 応答内のソート順を決定するには、これらの一般的なソートパラメータを使用し、タスクフィールドをカンマで区切ったリストを指定します。ソート順はフィールドごとに指定できます。降順を指定するにはダッシュ記号（-）を使用します。デフォルトのソート順は昇順です。

次のサンプルは、結果をソートする方法を示しています。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/task?recordsToReturn=100&owner=sergey&status=OPEN&sort=-
priority
```

サンプル API 要求

次のサンプル要求はタスクのリストを取得します。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/task
```

この要求には本文は含まれません。

サンプル API 応答

次のサンプル応答では、JSON 形式のタスクのリストが示されています。


```
{
    "firstRecord": 1,
    "recordCount": 10,
    "pageSize": 10,
    "task": [
        {
            "link": [
                {
                    "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/task/urn:b4p2:15443",
                    "rel": "self"
                }
            ],
            "taskType": {
                "name": "ReviewNoApprove",
                "label": "Review No approve",
                "taskKind": "REVIEW",
                "taskAction": [
                    {
                        "name": "Escalate",
                        "label": "Escalate",
                        "nextTaskType": "AVOSBeFinalReview",
                        "comment": "AS_REQUIRED",
                        "attachment": "NEVER",
                        "manualReassign": false,
                        "closeTaskView": true,
                        "cancelTask": false
                    },
                    {
                        "name": "Reject",
                        "label": "Reject",
                        "nextTaskType": "AVOSBeUpdate",
                        "comment": "MANDATORY",
                        "attachment": "MANDATORY",
                        "manualReassign": false,
                        "closeTaskView": true,
                        "cancelTask": false
                    },
                    {
                        "name": "Disclaim",
                        "label": "Disclaim",
                        "nextTaskType": "AVOSBeReviewNoApprove",
                        "comment": "AS_REQUIRED",
                        "attachment": "NEVER",
                        "manualReassign": false,
                        "closeTaskView": true,
                        "cancelTask": false
                    }
                ],
                "pendingBVT": true,
                "updateType": "PENDING"
            },
            "taskId": "urn:b4p2:15443",
            "title": "Review changes in SMITH,SMITH",
            "dueDate": "2015-07-15T21:45:59-07:00",
            "status": "OPEN",
            "priority": "NORMAL",
            "businessEntity": "Person"
        },
```

```
{
    "link": [
        {
            "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/task/urn:b4p2:15440",
            "rel": "self"
        }
    ],
    "taskType": {
        "name": "ReviewNoApprove",
        "label": "Review No approve",
        "taskKind": "REVIEW",
        "pendingBVT": true,
        "updateType": "PENDING"
    },
    "taskId": "urn:b4p2:15440",
    "title": "Review changes in SMITH,JOHN",
    "dueDate": "2015-07-15T21:37:50-07:00",
    "status": "OPEN",
    "priority": "NORMAL",
    "businessEntity": "Person"
},
{
    "link": [
        {
            "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/task/urn:b4p2:15437",
            "rel": "self"
        }
    ],
    "taskType": {
        "name": "ReviewNoApprove",
        "label": "Review No approve",
        "taskKind": "REVIEW",
        "taskAction": [
            {
                "name": Reject",
                "label": "Reject",
                "nextTaskType:AVOSBeUpdate",
                "comment": "AS_REQUIRED",
                "attachment": "MANDATORY",
                "manualReassign": false,
                "closeTaskView": true,
                "cancelTask": false
            }
        ],
        "pendingBVT": true,
        "updateType": "PENDING"
    },
    "taskId": "urn:b4p2:15437",
    "title": "Review changes in SMITH,JOHN",
    "dueDate": "2015-07-15T21:34:32-07:00",
    "status": "OPEN",
    "priority": "NORMAL",
    "businessEntity": "Person"
},
{
    "link": [
        {
            "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/task/urn:b4p2:14820",
            "rel": "self"
        }
    ],
    "taskType": {
        "name": "ReviewNoApprove",
        "label": "Review No approve",
        "taskKind": "REVIEW",
        "pendingBVT": true,
        "updateType": "PENDING"
    },
    "taskId": "urn:b4p2:14820",
    "title": "Review changes in STAS,STAS",
    "dueDate": "2015-07-14T10:40:51-07:00",
```

```
    "status": "OPEN",
    "priority": "NORMAL",
    "businessEntity": "Person"
},
{
    "link": [
        {
            "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/task/urn:b4p2:14809",
            "rel": "self"
        }
    ],
    "taskType": {
        "name": "ReviewNoApprove",
        "label": "Review No approve",
        "taskKind": "REVIEW",
        "pendingBVT": true,
        "updateType": "PENDING"
    },
    "taskId": "urn:b4p2:14809",
    "title": "Review changes in ,93C8ORSCOFSA687",
    "dueDate": "2015-07-14T08:28:15-07:00",
    "status": "OPEN",
    "priority": "NORMAL",
    "businessEntity": "Person"
},
{
    "link": [
        {
            "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/task/urn:b4p2:14609",
            "rel": "self"
        }
    ],
    "taskType": {
        "name": "ReviewNoApprove",
        "label": "Review No approve",
        "taskKind": "REVIEW",
        "pendingBVT": true,
        "updateType": "PENDING"
    },
    "taskId": "urn:b4p2:14609",
    "title": "Review changes in A8,A8",
    "dueDate": "2015-07-13T08:40:11-07:00",
    "status": "OPEN",
    "priority": "NORMAL",
    "businessEntity": "Person"
},
{
    "link": [
        {
            "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/task/urn:b4p2:14425",
            "rel": "self"
        }
    ],
    "taskType": {
        "name": "ReviewNoApprove",
        "label": "Review No approve",
        "taskKind": "REVIEW",
        "pendingBVT": true,
        "updateType": "PENDING"
    },
    "taskId": "urn:b4p2:14425",
    "title": "Review changes in A7,A7",
    "dueDate": "2015-07-10T14:11:02-07:00",
    "status": "OPEN",
    "priority": "NORMAL",
    "businessEntity": "Person"
},
{
    "link": [
        {
            "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/task/urn:b4p2:14422",
```

```
                    "rel": "self"
                }
            ],
            "taskType": {
                "name": "ReviewNoApprove",
                "label": "Review No approve",
                "taskKind": "REVIEW",
                "pendingBVT": true,
                "updateType": "PENDING"
            },
            "taskId": "urn:b4p2:14422",
            "title": "Review changes in A6,A6",
            "dueDate": "2015-07-10T13:54:09-07:00",
            "status": "OPEN",
            "priority": "NORMAL",
            "businessEntity": "Person"
        },
        {
            "link": [
                {
                    "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/task/urn:b4p2:14415",
                    "rel": "self"
                }
            ],
            "taskType": {
                "name": "ReviewNoApprove",
                "label": "Review No approve",
                "taskKind": "REVIEW",
                "pendingBVT": true,
                "updateType": "PENDING"
            },
            "taskId": "urn:b4p2:14415",
            "title": "Review changes in A5,A5",
            "dueDate": "2015-07-10T13:51:12-07:00",
            "status": "OPEN",
            "priority": "NORMAL",
            "businessEntity": "Person"
        },
        {
            "link": [
                {
                    "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/task/urn:b4p2:14355",
                    "rel": "self"
                }
            ],
            "taskType": {
                "name": "Notification",
                "label": "Notification",
                "taskKind": "REVIEW",
                "pendingBVT": false,
                "updateType": "ACTIVE"
            },
            "taskId": "urn:b4p2:14355",
            "title": "Review changes in A4,A4",
            "dueDate": "2015-07-10T10:31:57-07:00",
            "status": "OPEN",
            "priority": "NORMAL",
            "businessEntity": "Person"
        }
    ]
}
```


タスクの読み取り

［タスクの読み取り］REST API は、タスクタイプ、優先順位、ステータスなどのタスク詳細を返します。

この API は GET メソッドを使用します。

要求 URL

［タスクの読み取り］URL の形式は次のとおりです。

http://<host>:<port>/<context>/<database ID>/task/<taskId>

注: ［タスクの一覧表示］API は、タスクの ID を取得するために使用します。

［タスクの読み取り］URL に対して、次の HTTP GET 要求を行います。

GET http://<host>:<port>/<context>/<database ID>/task/<taskId>

サンプル API 要求

次のサンプル要求はタスクの詳細を返します。

GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/task/urn:b4p2:16605

サンプル API 応答

次のサンプル応答ではタスクの詳細が示されています。


```
{
    "taskType": {
        "name": "ReviewNoApprove",
        "label": "Review No Approve",
        "taskKind": "REVIEW",
        "taskAction": [
            {
                "name": "Escalate",
                "label": "Escalate",
                "nextTaskType": "AVOSBeFinalReview",
                "comment": "AS_REQUIRED",
                "attachment": "NEVER",
                "manualReassign": false,
                "closeTaskView": true,
                "cancelTask": false
            },
            {
                "name": "Reject",
                "label": "Reject",
                "nextTaskType": "AVOSBeUpdate",
                "comment": "MANDATORY",
                "attachment": "MANDATORY",
                "manualReassign": false,
                "closeTaskView": true,
                "cancelTask": false
            },
            {
                "name": "Disclaim",
                "label": "Disclaim",
                "nextTaskType": "AVOSBeReviewNoApprove",
                "comment": "AS_REQUIRED",
                "attachment": "NEVER",
                "manualReassign": false,
                "closeTaskView": true,
                "cancelTask": false
            }
        ],
        "pendingBVT": true,
        "updateType": "PENDING"
    },
    "taskId": "urn:b4p2:16605",
    "processId": "16603",
    "title": "Review changes in HERNANDEZ,ALEJANDRO",
    "dueDate": "2015-07-23T01:18:39.125-07:00",
    "status": "OPEN",
    "priority": "NORMAL",
    "taskRecord": [
```

```
        {
            "businessEntity": {
                "key": {
                    "rowid": "114",
                    "sourceKey": "SYS:114"
                },
                "name": "Person"
            }
        },
        {
            "businessEntity": {
                "key": {
                    "rowid": "114           ",
                    "sourceKey": "SYS:114",
                    "rowidXref": "4680363"
                },
                "name": "Person.XREF"
            }
        }
    ],
    "creator": "avos",
    "createDate": "2015-07-16T01:18:46.148-07:00",
    "attachments": [
        {
            "id": urn.b4p2:22203::file1.txt",
            "name": "file1.txt",
            "contentType": "text/plain",
            "creator": "admin",
            "createDate": "2018-02-26T14:31:05.590-05:00"
        }
    ],
    "businessEntity": "Person",
    "interactionId": "72340000003000",
    "orsId": "localhost-orcl-DS_UI1"
}
```

タスクの作成

［タスクの作成］REST API は、タスクを作成し、ワークフローを開始します。

この API は、POST メソッドを使用してタスクを作成し、そのタスクが含まれるワークフロープロセスの ID を返します。

要求 URL

［タスクの作成］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/task
```

［タスクの作成］URL に対して、次の HTTP POST 要求を行います。

```
POST http://<host>:<port>/<context>/<database ID>/task
```

要求と一緒に送信するデータのメディアタイプを指定するには、Content-Type ヘッダーを追加します。

```
POST http://<host>:<port>/<context>/<database ID>/task
Content-Type: application/<json/xml>
```

要求本文

タスクを作成するときにはタスク属性を指定します。要求でタスクデータを送信するには、JSON 形式または XML 形式を使用します。

次の表に、要求本文内のタスクパラメータを示します。

パラメータ	説明
taskType	レコードで実行できる一連のアクションです。タスクタイプの指定には名前属性を使用します。タスクタイプの詳細については、『*Multidomain MDM Data Director の実装ガイド*』を参照してください。
owner	作成者によってタスクが割り当てられるユーザー。
title	タスクの簡単な説明。
comments	タスクについてのコメント。
添付ファイル	タスクの添付ファイル。
dueDate	所有者がそのタスクをいつ完了する必要があるかを示す日付。
status	ワークフロー内のタスクの状態。以下の 2 つの値のいずれかを使用します。 - Open: タスクがまだ開始されていないか、または進捗中である。 - Closed: タスクが完了しているか、またはキャンセルされている。
priority	タスクの重要度。high、normal、low のうちのいずれかの値を使用します。デフォルトは normal です。
creator	タスクを作成するユーザー。
createDate	タスクを作成する日。
orsId	Hub コンソールのデータベースツールに登録されている、オペレーショナル参照ストア（ORS）の ID。
processId	ActiveVOS(R)のタスクタイプ ID。詳細については、『*Multidomain MDM Data Director の実装ガイド*』を参照してください。
taskRecord	タスクに関連付けられたビジネスオブジェクトルートレコードまたは相互参照レコード。行 ID、またはソースシステムとソースキーを使用してレコードを指定します。
businessEntity	taskRecord が属するビジネスエンティティの名前。
interactionId	相互作用の ID。相互作用 ID を使用して、タスクとレコード間のタスクコンテキストリレーションを保持します。
groups	指定したユーザーグループのすべてのユーザーにタスクを割り当てます。MDM Hub コンソールでユーザーグループを定義します。グループを配列として指定します。

次のサンプルコードは、行 ID を使用して taskRecord を指定しています。

```
taskRecord: [{
       businessEntity:{
         name: "Person",
         key:{
            rowid: "233",
         }
```

```
      }
    }]
```

要求本文の形式は次のとおりです。

```
{
    taskType: {name:"name of the task"},
    owner: "user who performs the task",
    title: "title of the task",
    comments: "description of the task",
    attachments: [
        {
            id: "TEMP_SVR1.1VDVS"
        }
    ],
    dueDate: "date to complete the task",
    status: "status of the task",
    priority: "priority of the task",
    creator: "use who creates the task",
    createDate: "date on which the task is created",
    updatedBy: "user who last updated the task",
    lastUpdateDate: "date on which the task was last updated",
    businessEntity: "name of the business entity",
    interactionID: "ID of an interaction",
    groups: ["group name A", "group name B", ...],
    orsId: "database ID",
    processId: "ActiveVOS task type ID",
    taskRecord: [{
      businessEntity:{
        name: "name of the business entity",
        key:{
           rowid: "rowId of the record", //Use the rowId or the source system and source key to identify the
record.
        }
      }
    }]
}
```

関連項目：

- 「UTC での日付と時刻の形式」 (ページ 34)

サンプル API 要求

次のサンプル要求は、ルートレコードのタスクを作成します。

```
POST http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/task
```

```
{
    taskType: {name:"UpdateWithApprovalWorkflow"},
    taskId: "",
    owner: "manager",
    title: "Smoke test task",
    comments: "Smoke testing",
    dueDate: "2015-06-15T00:00:00",
    status: "OPEN",
    priority: "NORMAL",
    creator: "admin",
    createDate: "2015-06-15T00:00:00",
    updatedBy: "admin",
    lastUpdateDate: "2015-06-15T00:00:00",
    businessEntity: "Person",
    orsId: "localhost-orcl-DS_UI1",
    processId: "IDDUpdateWithApprovalTask",
    taskRecord: [{
      businessEntity:{
        name: "Person",
        key:{
           rowid: "123"
```

```
        }
      }
    }]
}
```

サンプル API 応答

次のサンプルでは、タスクが正常に作成された場合の応答が示されています。この API は、タスクが含まれるワークフロープロセスの ID を返します。

```
{
    "parameters": {
        "processId": "15827"
    }
}
```

タスクの更新

［タスクの更新］REST API は、1 つのタスクを更新します。

この API は、一部のタスクフィールドの更新には PATCH メソッドを使用し、タスク全体の更新には PUT メソッドを使用します。タスクの ID を URL パラメータとして指定します。

要求 URL

［タスクの更新］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/task/<taskId>
```

注: ［タスクの一覧表示］API は、タスクの ID を取得するために使用します。

タスク全体を更新するには、［タスクの更新］URL に対して次の HTTP PUT 要求を行います。

```
PUT http://<host>:<port>/<context>/<database ID>/task/<taskId>
```

一部のタスクフィールドを更新するには、［タスクの更新］URL に対して次の HTTP PATCH 要求を行います。

```
PATCH http://<host>:<port>/<context>/<database ID>/task/<taskId>
```

要求と一緒に送信するデータのメディアタイプを指定するには、Content-Type ヘッダーを追加します。

```
PUT http://<host>:<port>/<context>/<database ID>/task/<taskId>
Content-Type: application/<json/xml>
```

要求本文

［タスクの読み取り］API は、タスクの詳細を取得するために使用します。タスクを更新するときにはタスク属性を指定します。要求で更新するデータを送信するには、JSON 形式または XML 形式を使用します。

次の表に、要求本文内のタスクパラメータを示します。

パラメータ	説明
taskType	レコードで実行できる一連のアクションです。タスクタイプの指定には名前属性を使用します。タスクタイプの詳細については、『*Multidomain MDM Data Director の実装ガイド*』を参照してください。
taskId	タスクの ID。
owner	タスクを実行するユーザー。

パラメータ	説明
title	タスクの簡単な説明。
comments	タスクについてのコメント。
添付ファイル	タスクの添付ファイル。
dueDate	所有者がそのタスクをいつ完了する必要があるかを示す日付。
status	ワークフロー内のタスクの状態。以下の 2 つの値のいずれかを使用します。 - Open: タスクがまだ開始されていないか、または進捗中である。 - Closed: タスクが完了しているか、またはキャンセルされている。
priority	タスクの重要度。high、normal、low のうちのいずれかの値を使用します。デフォルトは normal です。
creator	タスクを作成するユーザー。
createDate	タスクが作成された日。
updatedBy	そのタスクを更新するユーザー。
lastUpdateDate	タスクが最後に更新された日。
orsId	Hub コンソールのデータベースツールで登録された ORS の ID。
processId	タスクを含むワークフロープロセスの ID。
taskRecord	タスクに関連付けられたルートレコードまたは相互参照レコード。行 ID、またはソースシステムとソースキーを使用してレコードを指定します。
businessEntity name	taskRecord が属するビジネスエンティティの名前。

次のサンプルコードは、行 ID を使用して taskRecord を指定しています。

```
taskRecord: [{
       businessEntity:{
         name: 'Person',
         key:{
            rowid: '233',
            systemName: '',
            sourceKey: ''
         }
       }
     }]
```

PATCH 要求の場合、要求本文には変更するタスクフィールドが含まれます。タスクタイトル、優先順位、期限、および所有者は変更できます。

PUT 要求の場合、要求本文にはすべてのタスクフィールドが含まれます。PUT 要求には次の要求本文を使用します。

```
{
     taskType: {name:"name of the task"},
     taskId: "ID of the task",
     owner: "user who performs the task",
     title: "title of the task",
     comments: "description of the task",
     attachments: [
          {
```

```
            id: "TEMP_SVR1.1VDVS"
        }
    ],
    dueDate: "date to complete the task",
    status: "status of the task",
    priority: "priority of the task",
    creator: "use who creates the task",
    createDate: "date on which the task is created",
    updatedBy: "user who last updated the task",
    lastUpdateDate: "date on which the task was last updated",
    businessEntity: "name of the business entity",
    orsId: "database ID",
    processId: 'ActiveVOS task type ID',
    taskRecord: [{
      businessEntity:{
        name: 'name of the business entity',
        key:{
           rowid: 'rowId of the record',//Use the rowId or the source system and source key to identify the
record.
           systemName: '',
           sourceKey: ''
        }
      }
    }]
}
```

関連項目：

- 「UTC での日付と時刻の形式」 (ページ 34)

サンプル API 要求

次のサンプル PUT 要求はタスク全体を更新します。

```
PUT http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/task/urn:b4p2:15934
```

```
{
    taskType: {name:"UpdateWithApprovalWorkflow"},
    taskId: "urn:b4p2:15934",
    owner: "John",
    title: "Smoke test task - updated",
    comments: "Smoke testing - updated",
    attachments: [
        {
            id: "TEMP_SVR1.1VDVS"
        }
    ],
    dueDate: "2015-08-15T00:00:00",
    status: "OPEN",
    priority: "HIGH",
    creator: "admin",
    createDate: "2015-06-15T00:00:00",
    updatedBy: "admin",
    lastUpdateDate: "2015-06-15T00:00:00",
    businessEntity: "Person",
    orsId: "localhost-orcl-DS_UI1",
    processId: '3719',
    taskRecord: [{
      businessEntity:{
        name: 'Person',
        key:{
           rowid: '123',
           systemName: '',
           sourceKey: ''
        }
      }
    }]
}
```

次のサンプル PATCH 要求は、一部のタスクフィールドを更新します。

```
PATCH http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/task/urn:b4p2:15934
{
     processId: "3719",
     priority: "HIGH",
     owner: "John"
}
```

サンプル API 応答

この API は、タスクが正常に更新された場合に 200 OK 応答を返します。応答本文は空です。

タスクの完了

［タスクの完了］REST API は、ワークフロー内のすべてのタスクを完了した後、タスクワークフローを閉じます。この API は、タスク関連のレコードをすべて処理した後でワークフローを閉じるために使用します。例えば、マージ候補を選択する場合、マージワークフローを開始するタスクを作成できます。各候補をプレビューし、その候補をマージするかまたは一致でないとしてマークすると、マージタスクは完了します。この API はマージワークフローを閉じるために使用します。

この API は PUT メソッドを使用します。

要求 URL

［タスクの完了］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/task/<taskId>?action=complete
```

［タスクの完了］URL に対して、次の HTTP PUT 要求を行います。

```
PUT http://<host>:<port>/<context>/<database ID>/task/<taskId>?action=complete
```

要求と一緒に送信するデータのメディアタイプを指定するには、Content-Type ヘッダーを追加します。

```
PUT http://<host>:<port>/<context>/<database ID>/task/<taskId>?action=complete
Content-Type: application/<json/xml>
```

要求本文

タスク詳細を要求本文で送信します。［タスクの読み取り］API は、タスクの詳細を取得するために使用します。

次の表に、要求本文内のタスクパラメータを示します。

パラメータ	説明
taskType	レコードで実行できる一連のアクションです。タスクタイプの指定には名前属性を使用します。タスクタイプの詳細については、『*Multidomain MDM Data Director の実装ガイド*』を参照してください。
taskId	タスクの ID。
owner	タスクを実行するユーザー。
title	タスクの簡単な説明。
comments	タスクについてのコメント。

パラメータ	説明
dueDate	所有者がそのタスクをいつ完了する必要があるかを示す日付。
status	ワークフロー内のタスクの状態。以下の 2 つの値のいずれかを使用します。 - Open: タスクがまだ開始されていないか、または進捗中である。 - Closed: タスクが完了しているか、またはキャンセルされている。
priority	タスクの重要度。
creator	タスクを作成するユーザー。
createDate	タスクが作成された日。
updatedBy	そのタスクを更新するユーザー。
lastUpdateDate	タスクが最後に更新された日。
orsId	Hub コンソールのデータベースツールで登録された ORS の ID。
processId	タスクを含むワークフロープロセスの ID。
taskRecord	タスクに関連付けられたルートレコードまたは相互参照レコード。行 ID、またはソースシステムとソースキーを使用してレコードを指定します。
businessEntity name	taskRecord が属するビジネスエンティティの名前。

次のサンプルコードは、行 ID を使用して taskRecord を指定しています。

```
taskRecord: [{
       businessEntity:{
         name: 'Person',
         key:{
            rowid: '233',
            systemName: '',
            sourceKey: ''
         }
       }
     }]
```

関連項目：

- 「UTC での日付と時刻の形式」 (ページ 34)

サンプル API 要求

次のサンプル要求は、マージワークフローを完了します。

```
PUT http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/task/urn:b4p2:20210?action=complete
```

```
{
     "taskType": {"name":"Merge"},
     "taskId": "urn:b4p2:20210",
     "owner": "admin",
     "dueDate": "2015-08-14T17:00:00-07:00",
     "status": "OPEN",
     "priority": "NORMAL",
     "creator": "admin",
     "createDate": "2015-06-15T00:00:00",
     "updatedBy": "admin",
     "lastUpdateDate": "2015-06-15T00:00:00",
     "businessEntity": "Person",
```

```
    "orsId": "localhost-orcl-DS_UI1",
    "processId": "20208",
    "taskRecord": [{
      "businessEntity":{
        "name": "Person",
        "key":{
           "rowid": "233",
           "systemName": "",
           "sourceKey": "" 
        }
      }
    }]
}
```

サンプル API 応答

この API は、タスクワークフローが正常に完了された場合に 200 OK 応答を返します。応答本文は空です。

タスクアクションの実行

［タスクアクションの実行］REST API は、その後の処理のためにタスクをワークフローに戻します。各タスクタイプには、一連のタスクアクションと、タスクのシーケンスを指定するワークフローがあります。タスクアクションを実行すると、タスクはワークフロー内の次の手順に移動します。タスクアクションに後続タスクがない場合、そのタスクアクションを実行するとワークフローは終了します。

この API は、POST メソッドを使用してアクションを実行します（タスクの承認、エスカレーション、キャンセルなど）。

要求 URL

次の URL は、［タスクアクションの実行］URL の形式を示しています。

```
http://<host>:<port>/<context>/<database ID>/task/<taskId>?action=<taskAction>
```

注: ［タスクの一覧表示］API は、タスクの ID を取得するために使用します。

［タスクアクションの実行］URL に対して、次の HTTP POST 要求を行います。

```
POST http://<host>:<port>/<context>/<database ID>/task/<taskId>?action=<taskAction>
```

タスクアクションを実行する前にタスクを編集する場合は、要求データのメディアタイプを指定する Content-Type ヘッダーを追加します。

```
POST http://<host>:<port>/<context>/<database ID>/task/<taskId>?action=<taskAction>
Content-Type: application/<json/xml>
```

要求本文

タスクアクションを実行する前にタスク詳細を変更する場合は、要求本文にタスクデータを指定します。

次の表に、要求本文内のパラメータを示します。

パラメータ	説明
taskType	レコードで実行できる一連のアクションです。タスクタイプの指定には名前属性を使用します。タスクタイプの詳細については、『*Multidomain MDM Data Director の実装ガイド*』を参照してください。
taskId	タスクの ID。

パラメータ	説明
owner	タスクを実行するユーザー。
title	タスクの簡単な説明。
comments	タスクについてのコメント。
添付ファイル	タスクの添付ファイル。
dueDate	所有者がそのタスクをいつ完了する必要があるかを示す日付。
status	ワークフロー内のタスクの状態。以下の 2 つの値のいずれかを使用します。 - Open: タスクがまだ開始されていないか、または進捗中である。 - Closed: タスクが完了しているか、またはキャンセルされている。
priority	タスクの重要度。high、normal、low のうちのいずれかの値を使用します。
creator	タスクを作成するユーザー。
createDate	タスクが作成された日。
updatedBy	そのタスクを更新するユーザー。
lastUpdateDate	タスクが最後に更新された日。
businessEntity	ビジネスエンティティの名前。
orsId	Hub コンソールのデータベースツールで登録された ORS の ID。
processId	タスクを含むワークフロープロセスの ID。
taskRecord	タスクに関連付けられたルートレコードまたは相互参照レコード。行 ID、またはソースシステムとソースキーを使用してレコードを指定します。
businessEntity name	taskRecord が属するビジネスエンティティの名前。

次のサンプルコードは、行 ID を使用して taskRecord を指定しています。

```
taskRecord: [{
       businessEntity:{
         name: 'Person',
         key:{
            rowid: '233',
            systemName: '',
            sourceKey: ''
         }
       }
     }]
```

関連項目：

- 「UTC での日付と時刻の形式」 (ページ 34)

サンプル API 要求

次のサンプル要求は、タスクをキャンセルしてワークフローを終了させます。

```
POST http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/task/urn:b4p2:15934?taskAction=Cancel
```

```
{
     taskType: {
      name:"UpdateWithApprovalWorkflow",
      taskAction: [{name: "Cancel"}]
     },
     taskId: "urn:b4p2:15934",
     owner: "manager",
     title: "Smoke test task 222",
     comments: "Smoke testing",
     attachments: [
          {
               id: "TEMP_SVR1.1VDVS"
          }
     ],
     dueDate: "2015-06-15T00:00:00",
     status: "OPEN",
     priority: "NORMAL",
     creator: "admin",
     createDate: "2015-06-15T00:00:00",
     updatedBy: "admin",
     lastUpdateDate: "2015-06-15T00:00:00",
     businessEntity: "Person",
     orsId: "localhost-orcl-DS_UI1",
     processId: '3685',
     taskRecord: [{
       businessEntity:{
         name: 'Person',
         key:{
            rowid: '123',
            systemName: '',
            sourceKey: ''
         }
       }
     }]
}
```

サンプル API 応答

この API は、タスクアクションが正常に実行された場合に 200 OK 応答を返します。応答本文は空です。

割り当て可能なユーザーの一覧表示

［割り当て可能なユーザーの一覧表示］REST API は、特定のタスクタイプに属するタスクの割り当て対象として指定できるユーザーのリストを返します。この API は、タスクのターゲットユーザーを取得するために使用します。

この API は GET メソッドを使用します。

要求 URL

［割り当て可能なユーザーの一覧表示］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/user?taskType=<task type>&businessEntity=<business entity name>
```

［割り当て可能なユーザーの一覧表示］URL に対して、次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/<context>/<database ID>/user?taskType=<task type>&businessEntity=<business entity name>
```

クエリパラメータ

次の表に URL 内の必須パラメータを示します。

パラメータ	説明
taskType	レコードで実行できる一連のアクションです。タスクタイプには、承認を伴う更新、オプションで承認を伴う更新、マージ、マージ解除、承認を伴わない確認、最終確認、拒否されたレコードの更新などがあります。
businessEntity	ビジネスエンティティの名前。

サンプル API 要求

次のサンプル要求は、割り当て可能なユーザーのリストを取得します。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/user.json?taskType=ReviewNoApprove&businessEntity=Person
```

サンプル API 応答

次のサンプル応答では、タスクタイプ ReviewNoApprove の割り当て可能なユーザーのリストが示されています。

```
{
    "users": {
        "user": [
            {
                "userName": "admin"
            }
        ]
    },
    "roles": {}
}
```

タスクの一括クレーム

タスクの一括クレーム REST API は、複数のタスクを引き受けます。この API を使用して、自分が潜在的な所有者であるタスクを引き受けます。

この API は POST メソッドを使用します。

注: 最大で 100 タスクを要求に指定できます。

要求 URL

［タスクの一括クレーム］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>
```

［タスクの一括クレーム］URL に対して、次の HTTP POST 要求を行います。

```
POST http://<host>:<port>/<context>/<database ID>
```

要求本文

クレームするタスクを要求本文で指定します。

次の表に、要求本文内のパラメータを示します。

パラメータ	説明
taskId	タスクの ID。
opType	タスクで実行する操作。タスクをクレームするには、Claim を指定します。

サンプル API 要求

次のサンプル要求は、複数のタスクを要求します。

```
POST http://localhost:8080/cmx/task/operations/mdmdb3-MDM_SAMPLE
{
    "tasks":
    [
        {
            "taskId": "urn:b4p2:2315"
        },
        {
            "taskId": "urn:b4p2:2312"
        },
        {
            "taskId": "urn:b4p2:2309"
        },
        {
            "taskId": "urn:b4p2:2306"
        },
        {
            "taskId": "urn:b4p2:2303"
        }
    ],
    "opType": "Claim"
}
```

サンプル API 応答

次のサンプル応答は、正常に要求されたタスクのリストを示しています。

```
{
    "tasks":
    [
        {
            "taskId": "urn:b4p2:2315",
            "status": "Completed"
        },
        {
            "taskId": "urn:b4p2:2312",
            "status": "Completed"
        },
        {
            "taskId": "urn:b4p2:2309",
            "status": "Completed"
        },
        {
            "taskId": "urn:b4p2:2306",
            "status": "Completed"
        },
        {
            "taskId": "urn:b4p2:2303",
            "status": "Completed"
        }
```

```
    ]
}
```

タスクの一括解放

タスクの一括解放 REST API は、複数のタスクを未割り当てタスクのプールに解放します。

この API は POST メソッドを使用します。

注: 最大で 100 タスクを要求に指定できます。

要求 URL

［タスクの一括解放］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>
```

［タスクの一括解放］URL に対して、次の HTTP POST 要求を行います。

```
POST http://<host>:<port>/<context>/<database ID>
```

要求本文

解放してプールに戻すタスクを要求本文で指定します。

次の表に、要求本文内のパラメータを示します。

パラメータ	説明
taskId	タスクの ID。
comments	オプション。タスクについてのコメント。
opType	タスクで実行する操作。Release を指定して、タスクを解放し使用可能なタスクのプールに戻します。

サンプル API 要求

次のサンプル要求は、複数のタスクを解放し、各タスクに同じコメントを追加します。

```
POST http://localhost:8080/cmx/task/operations/mdmdb3-MDM_SAMPLE
{
    "tasks":
    [
        {
            "taskId": "urn:b4p2:2318"
        },
        {
            "taskId": "urn:b4p2:2209"
        }
    ],
    "comments": "John is on vacation. Releasing these tasks back to the pool of available tasks.",
    "opType": "Release"
}
```

次のサンプル要求は、複数のタスクを解放し、各タスクに異なるコメントを追加します。

```
POST http://localhost:8080/cmx/task/operations/mdmdb3-MDM_SAMPLE
{
    "tasks":
    [
        {
            "taskId": "urn:b4p2:2318",
```

```
            "comments": "Jane is on vacation."
        },
        {
            "taskId": "urn:b4p2:2209"
            "comments": "Joe is on vacation."
        }
    ],
    "opType": "Release"
}
```

サンプル API 応答

次のサンプル応答は、正常に解放されたタスクのリストを示しています。

```
{
    "tasks":
    [
        {
            "taskId": "urn:b4p2:2318",
            "status": "Completed"
        },
        {
            "taskId": "urn:b4p2:2209",
            "status": "Completed"
        }
    ]
}
```

タスクの一括割り当て

タスクの一括割り当て REST API により、タスクをユーザーに割り当てまたは再割り当てできます。この API を使用して、タスクをタスクの潜在的な所有者に割り当てまたは再割り当てします。

タスクの潜在的な所有者を判断するには、タスクの潜在的な所有者のリスト AP を使用します。詳細については、「複数のタスクの潜在的な所有者のリスト」 (ページ 109)を参照してください。

この API は POST メソッドを使用します。

注: 最大で 100 タスクを要求に指定できます。

要求 URL

[タスクの一括割り当て] URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>
```

[タスクの一括割り当て] URL に対して、次の HTTP POST 要求を行います。

```
POST http://<host>:<port>/<context>/<database ID>
```

要求本文

タスクに割り当てるタスクとユーザーを要求本文で指定します。

次の表に、要求本文内のパラメータを示します。

パラメータ	説明
taskId	タスクの ID。
comments	オプション。タスクについてのコメント。

パラメータ	説明
assignTo	タスクを担当するユーザー。 **注**: タスクは、そのタスクの潜在的な所有者であるユーザーにのみ割り当てることができます。タスクの潜在的な所有者を判断するには、タスクの潜在的な所有者のリスト AP を使用します。詳細については、「複数のタスクの潜在的な所有者のリスト」 (ページ 109)を参照してください。
opType	タスクで実行する操作。タスクを割り当てるには、Assign を指定します。

サンプル API 要求

次のサンプル要求は、ユーザーを複数のタスクに割り当て、各タスクに同じコメントを追加します。

```
POST http://localhost:8080/cmx/task/operations/mdmdb3-MDM_SAMPLE
{
    "tasks":
    [
        {
            "taskId": "urn:b4p2:5351"
        },
        {
            "taskId": "urn:b4p2:5352"
        }
    ],
    "assignTo": "srmngr1",
    "comments": "Please take a look at these tasks instead.",
    "opType": "Assign"
}
```

次のサンプル要求は、タスクを異なるユーザーに割り当て、各タスクに異なるコメントを追加します。

```
POST http://localhost:8080/cmx/task/operations/mdmdb3-MDM_SAMPLE
{
    "tasks":
    [
        {
            "taskId": "urn:b4p2:5351",
            "assignTo": "srmngr1",
            "comments": "Please help John take a look at this task."
        },
        {
            "taskId": "urn:b4p2:5352",
            "assignTo": "srmngr2"
            "comments": "Please help Jane take a look at this task."
        }
    ],
    "opType": "Assign"
}
```

サンプル API 応答

次のサンプル応答は、正常に割り当てられたタスクのリストを示しています。

```
{
    "tasks":
    [
        {
            "taskId": "urn:b4p2:5351",
            "status": "Completed"
        },
        {
            "taskId": "urn:b4p2:5352",
            "status": "Completed"
        }
```

```
    ]
}
```

タスクの一括編集

タスクの一括編集 REST API により、複数のタスクの詳細を編集できます。

この API は POST メソッドを使用していくつかのタスクフィールドを更新します。

注: 最大で 100 タスクを要求に指定できます。

要求 URL

［タスクの一括編集］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>
```

［タスクの一括編集］URL に対して、次の HTTP POST 要求を行います。

```
POST http://<host>:<port>/<context>/<database ID>
```

要求本文

編集するタスクを指定し、要求本文でタスクデータを指定します。

次の表に、要求本文内のパラメータを示します。

パラメータ	説明
taskId	タスクの ID。
dueDate	オプション。所有者がそのタスクをいつ完了する必要があるかを示す日付。
comments	オプション。タスクについてのコメント。
title	オプション。タスクの簡単な説明。
priority	オプション。タスクの重要度。high、normal、low のうちのいずれかの値を使用します。
assignTo	オプション。タスクを担当するユーザー。
fileId	オプション。タスクに添付するファイルの ID。 タスクにファイルを添付するには、ファイルを一時ストレージにアップロードする必要があります。詳細については、付録 B, 「REST API を使用したファイルのアップロード」 (ページ 329)を参照してください。
opType	タスクで実行する操作。タスクを編集するには、Edit を指定します。

サンプル API 要求

次のサンプル要求は、同じタスクの詳細を使用して複数のタスクを編集します。

```
POST http://localhost:8080/cmx/task/operations/mdmdb3-MDM_SAMPLE
{
    "tasks":
    [
        {
            "taskId": "urn:b4p2:4383"
        },
        {
```

```
            "taskId": "urn:b4p2:4384"
        }
    ],
    "comments": "Please work on this task ASAP. There's more information in the attachment.",
    "priority": "High",
    "title": "Test Demo Review Person",
    "dueDate": "2019-05-01 00:00:00",
    "fileId": "TEMP_SVR1.1IOPK",
    "opType": "Edit"
}
```

次のサンプル要求は、異なるタスクの詳細を使用して複数のタスクを編集します。

```
POST http://localhost:8080/cmx/task/operations/mdmdb3-MDM_SAMPLE
{
    "tasks":
    [
        {
            "taskId": "urn:b4p2:4383",
            "comments": "Added a screenshot to help clarify.",
            "priority": "High",
            "title": "Test Demo Review Person",
            "dueDate": "2019-05-01 00:00:00",
            "fileId": "TEMP_SVR1.1IOPK"
        },
        {
            "taskId": "urn:b4p2:4384",
            "priority": "High"
        }
    ],
    "opType": "Edit"
}
```

サンプル API 応答

次のサンプル応答は、正常に編集されたタスクのリストを示しています。

```
{
    "tasks":
    [
        {
            "taskId": "urn:b4p2:4383",
            "status": "Completed"
        },
        {
            "taskId": "urn:b4p2:4384",
            "status": "Completed"
        }
    ]
}
```

一括タスクアクション

一括タスクアクション REST API は、複数のタスクでタスクアクションを実行します。タスクアクションを実行すると、タスクはワークフロー内の次の手順に移動します。タスクアクションに後続タスクがない場合、そのタスクアクションを実行するとワークフローは終了します。

この API は、POST メソッドを使用してアクションを実行します（タスクの承認、エスカレーション、拒否など）。

タスクで実行できるタスクアクションを判断するには、タスクアクション取得 API を使用します。詳細については、「タスクアクションの取得」 (ページ 107)を参照してください。

注: 最大で 100 タスクを要求に指定できます。

要求 URL

［一括タスクアクション］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>
```

［一括タスクアクション］URL に対して、次の HTTP POST 要求を行います。

```
POST http://<host>:<port>/<context>/<database ID>
```

要求本文

タスクアクションを実行するタスクと、実行するアクションを指定します。

次の表に、要求本文内のパラメータを示します。

パラメータ	説明
taskId	タスクの ID。
actionName	実行するタスクアクション。 タスクアクション取得 API を使用して、タスクで実行できるタスクアクションを決定します。
opType	タスクで実行する操作。Action を指定して、タスクアクションを実行します。

サンプル API 要求

次のサンプル要求は、タスクを承認します。

```
POST http://localhost:8080/cmx/task/operations/mdmdb3-MDM_SAMPLE
{
    "tasks":
    [
        {
            "taskId": "urn:b4p2:5351"
        },
        {
            "taskId": "urn:b4p2:5352"
        }
    ],
    "actionName": "Approve",
    "opType": "Action"
}
```

次のサンプル要求は、1 つのタスクを承認し、1 つのタスクをエスカレーションします。

```
POST http://localhost:8080/cmx/task/operations/mdmdb3-MDM_SAMPLE
{
    "tasks":
    [
        {
            "actionName": "Approve",
            "taskId": "urn:b4p2:5353"
        },
        {
            "actionName": "Escalate",
            "taskId": "urn:b4p2:5354"
        }
    ],
    "opType": "Action"
}
```

サンプル API 応答

次のサンプル応答は、正常に編集されたタスクのリストを示しています。

```
{
    "tasks":
    [
        {
            "taskId": "urn:b4p2:5351",
            "status": "Completed"
        },
        {
            "taskId": "urn:b4p2:5352",
            "status": "Completed"
        }
    ]
}
```

タスクアクションの取得

タスクアクションの取得 REST API は、タスクで実行できるタスクアクションのリストを取得します。

この API は POST メソッドを使用してタスクアクションのリストを取得します。

注: 最大で 100 タスクを要求に指定できます。

要求 URL

［タスクアクションの取得］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>
```

［タスクアクションの取得］URL に対して、次の HTTP POST 要求を行います。

```
POST http://<host>:<port>/<context>/<database ID>
```

要求本文

実行できるタスクアクションを知りたいタスクを指定します。

次の表に、要求本文内のパラメータを示します。

パラメータ	説明
taskId	タスクの ID。
opType	タスクで実行する操作。GetTaskActions を指定して、タスクで実行できるタスクアクションのリストを取得します。

サンプル API 要求

次のサンプル要求は、タスクで実行できるタスクアクションを取得します。

```
POST http://localhost:8080/cmx/task/operations/mdmdb3-MDM_SAMPLE
{
    "tasks":
    [
        {
            "taskId": "urn:b4p2:5351"
        },
        {
            "taskId": "urn:b4p2:5352"
        }
```

```
    ],
    "opType": "GetTaskActions"
}
```

サンプル API 応答

次のサンプル応答は、タスクと各タスクで実行できるタスクアクションのリストを示しています。

```
{
    "tasks":
    [
        {
            "taskId": "urn:b4p2:5351",
            "status": "Completed",
            "taskActions":
            [
                {
                    "name": "Approve",
                    "displayName": "Approve",
                    "description": null,
                    "manualReassign": false,
                    "closeTaskView": true,
                    "cancelTask": false,
                    "nextTaskType": "AVOSBeNotification"
                },
                {
                    "name": "Reject",
                    "displayName": "Reject",
                    "description": null,
                    "manualReassign": false,
                    "closeTaskView": true,
                    "cancelTask": false,
                    "nextTaskType": "AVOSBeUpdate"
                },
                {
                    "name": "Disclaim",
                    "displayName": "Disclaim",
                    "description": null,
                    "manualReassign": true,
                    "closeTaskView": true,
                    "cancelTask": false,
                    "nextTaskType": "AVOSBeFinalReview"
                }
            ]
        },
        {
            "taskId": "urn:b4p2:5352",
            "status": "Completed",
            "taskActions":
            [
                {
                    "name": "Approve",
                    "displayName": "Approve",
                    "description": null,
                    "manualReassign": false,
                    "closeTaskView": true,
                    "cancelTask": false,
                    "nextTaskType": "AVOSBeNotification"
                },
                {
                    "name": "Reject",
                    "displayName": "Reject",
                    "description": null,
                    "manualReassign": false,
                    "closeTaskView": true,
                    "cancelTask": false,
                    "nextTaskType": "AVOSBeUpdate"
                },
                {
                    "name": "Disclaim",
```

```
                "displayName": "Disclaim",
                "description": null,
                "manualReassign": true,
                "closeTaskView": true,
                "cancelTask": false,
                "nextTaskType": "AVOSBeFinalReview"
            }
        ]
    }
  ]
}
```

複数のタスクの潜在的な所有者のリスト

複数のタスクの潜在的な所有者のリスト REST API は、複数のタスクの潜在的な所有者を取得します。

この API は GET メソッドを使用します。

注: 最大で 100 タスクを要求に指定できます。

要求 URL

［タスクの潜在的なタスク所有者の一覧表示］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>
```

［タスクの潜在的なタスク所有者の一覧表示］URL に対して、次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/<context>/<database ID>
```

要求本文

潜在的な所有者を取得するタスクを指定します。

次の表に、要求本文内のパラメータを示します。

パラメータ	説明
taskId	タスクの ID。
opType	タスクで実行する操作。PotentialOwners を指定して、タスクの潜在的な所有者のリストを取得します。

サンプル API 要求

次のサンプル応答は、複数のタスクの潜在的な所有者を取得します。

```
GET http://localhost:8080/cmx/task/operations/mdmdb3-MDM_SAMPLE
{
    "tasks":
    [
        {
            "taskId": "urn:b4p2:4383"
        },
        {
            "taskId": "urn:b4p2:4382"
        }
    ],
    "opType": "PotentialOwners"
}
```

サンプル API 応答

次のサンプル応答は、各タスクの潜在的な所有者を示しています。

```
{
    "tasks":
    [
        {
            "taskId": "urn:b4p2:4383",
            "status": "Completed",
            "users":
            [

                [
                    "srmngr2",
                    "srmngr 2"
                ],

                [
                    "srmngr1",
                    "senior manager"
                ]
            ]
        },
        {
            "taskId": "urn:b4p2:4382",
            "status": "Completed",
            "users":
            [

                [
                    "srmngr2",
                    "srmngr 2"
                ],

                [
                    "srmngr1",
                    "senior manager"
                ]
            ]
        }
    ]
}
```

タスクの潜在的な所有者のリスト

タスクの潜在的な所有者のリスト REST API は、タスクの潜在的な所有者を取得します。

この API は GET メソッドを使用します。

要求 URL

［タスクの潜在的なタスク所有者の一覧表示］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/list/owners/<database ID>/<taskId>
```

［タスクの潜在的なタスク所有者の一覧表示］URL に対して、次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/<context>/list/owners/<database ID>
```

要求本文

潜在的な所有者を取得するタスクを指定します。

次の表に、要求本文内のパラメータを示します。

パラメータ	説明
taskId	タスクの ID。
opType	タスクで実行する操作。PotentialOwners を指定して、タスクの潜在的な所有者のリストを取得します。

サンプル API 要求

次のサンプル応答は、タスクの潜在的な所有者を取得します。

```
GET http://localhost:8080/cmx/task/operations/list/owners/mdmdb3-MDM_SAMPLE
{
    "tasks":
    [
        {
            "taskId": "urn:b4p2:4383"
        },
        {
            "taskId": "urn:b4p2:4382"
        }
    ],
    "opType": "PotentialOwners"
}
```

サンプル API 応答

次のサンプル応答は、タスクの潜在的な所有者を示しています。

```
{
    "tasks":
    [
        {
            "taskId": "urn:b4p2:4383",
            "status": "Completed",
            "users":
            [

                [
                    "srmngr2",
                    "srmngr 2"
                ],

                [
                    "srmngr1",
                    "senior manager"
                ]
            ]
        },
        {
            "taskId": "urn:b4p2:4382",
            "status": "Completed",
            "users":
            [

                [
                    "srmngr2",
                    "srmngr 2"
                ],
```

```
                [
                    "srmngr1",
                    "senior manager"
                ]
            ]
        }
    ]
}
```

ファイルのメタデータの一覧表示

［ファイルのメタデータの一覧表示］REST API は、ストレージ内のファイルメタデータのリストを返します。

BPM または TEMP ストレージを指定した［ファイルのメタデータの一覧表示］REST API を使用します。

この API は GET メソッドを使用します。

要求 URL

［ファイルのメタデータの一覧表示］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/<storage>
```

［ファイルのメタデータの一覧表示］URL に対して、次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/<context>/<database ID>/<storage>
```

サンプル API 要求

次のサンプル要求は、TEMP ストレージ内のファイルのメタデータのリストを取得します:

```
GET http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/TEMP
```

サンプル API 応答

次のサンプル応答は、ファイルのメタデータのリストを示しています:

```
{
    files: [
        {
            “fileId”: “TEMP_SVR1.1VDVS”,
            “fileName”: “file1.txt”,
            “fileType”: “text”,
            “fileContentType”: “text/plain”,
        },
        {
            “fileId”: “TEMP_SVR1.2ESDS”,
            “fileName”: “image1.png”,
            “fileType”: “image”,
            “fileContentType”: “image/png”,
        },
        ...
    ]
}
```

ファイルのメタデータの作成

［ファイルのメタデータの作成］REST API はファイルのメタデータを作成し、そのファイルのファイル ID を返します。

ファイル ID を使用して、ファイルのアップロード、添付、更新、ダウンロード、および削除を行うことができます。

DB または TEMP ストレージを指定した［ファイルのメタデータの作成］REST API を使用します。

この API は POST メソッドを使用します。

要求 URL

［ファイルのメタデータの作成］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/<storage>
```

［ファイルのメタデータの作成］URL に対して、次の HTTP POST 要求を行います。

```
POST http://<host>:<port>/<context>/<database ID>/<storage>
```

要求本文

ファイルのメタデータを指定します。

次の表では、要求本文のファイルのメタデータのパラメータについて説明します。

パラメータ	説明
fileName	ファイルの名前。例えば、file.txt。
fileType	ファイルタイプのカテゴリ。例えば、text や image。
fileContentType	ファイルのコンテンツタイプ。コンテンツタイプは、/で区切られたタイプとサブタイプで構成されます。例えば、image/png。

注: ［ファイルのメタデータの作成］REST API 要求に必要なパラメータは、ストレージに固有のものです。

サンプル API 要求

次のサンプル要求は、TEMP ストレージ内にファイルのメタデータを作成します:

```
POST http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/TEMP
```

```
{
    “fileName”: “file1.txt”,
    “fileType”: “text”,
    “fileContentType”: “text/plain”
}
```

サンプル API 応答

次の例は、TEMP ストレージ内にファイルのメタデータが正常に作成された場合の応答を示しています。この API はファイルのファイル ID を返します。

```
TEMP_SVR1.1VDVS
```

注: ファイル ID の形式は、<storage type>_<uniqueID>です。

ファイルのメタデータの取得

［ファイルのメタデータの取得］REST API は、ファイル ID に関連付けられているファイルのメタデータを返します。

BPM、BUNDLE、DB、または TEMP ストレージを指定した［ファイルのメタデータの取得］REST API を使用します。

この API は GET メソッドを使用します。

要求 URL

［ファイルのメタデータの取得］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/<storage>/<fileId>
```

［ファイルのメタデータの取得］URL に対して、次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/<context>/<database ID>/<storage>/<fileId>
```

サンプル API 要求

次のサンプル要求は、TEMP ストレージ内でファイル ID が TEMP_SVR1.1VDVS のファイルのメタデータを返します:

```
GET http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/TEMP/TEMP_SVR1.1VDVS
```

次のサンプル要求は、BUNDLE ストレージ内でファイル ID が besMetadata のリソースバンドルファイルのメタデータを返します:

```
GET http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/BUNDLE/besMetadata
```

サンプル API 応答

次のサンプル応答は、TEMP ストレージ内のファイル ID が TEMP_SVR1.1VDVS のファイルのメタデータを示しています:

```
{
    "fileName”: "file1.txt",
    "fileType”: "text",
    "fileContentType”: "text/plain"
}
```

次のサンプル応答は、BUNDLE ストレージ内のリソースバンドルファイル besMetadata のメタデータを示しています:

```
{
    "fileName": "besMetadata.zip",
    "fileType": "BES Metadata Bundle",
    "fileContentType": "application/zip",
    "digest": "a08c5d97da7e6a780ed7c427ff14a8d2d396438cd65b654ad67424e226f64a41"
}
```

ファイルのメタデータの更新

［ファイルのメタデータの更新］REST API は、ファイル ID に関連付けられているファイルのメタデータを更新します。

DB または TEMP ストレージを指定した［ファイルのメタデータの更新］REST API を使用します。

この API は PUT メソッドを使用します。

要求 URL

［ファイルのメタデータの更新］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/<storage>/<fileId>
```

［ファイルのメタデータの更新］URL に対して、次の HTTP PUT 要求を行います。

```
PUT
http://<host>:<port>/<context>/<database ID>/<storage>/<fileId>
```

サンプル API 要求

次のサンプル要求は、TEMP ストレージ内でファイル ID が TEMP_SVR1.1VDVS のファイルのメタデータを更新します:

```
PUT http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/TEMP/TEMP_SVR1.1VDVS

{
    “fileName”: “file2.txt”,
    “fileType”: “text”,
    “fileContentType”: “text/plain”
}
```

次のサンプル要求は、DB ストレージ内でファイル ID が DB_SVR1.0JU1 のファイルのメタデータを更新します:

```
PUT http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/DB/DB_SVR1.0JU1
{
    "fileName": "Document_2.pdf",
    "fileType": "pdf",
    "fileContentType": "application/pdf"
}
```

サンプル API 応答

この API は、ファイルのメタデータが正常に更新された場合に 200 OK 応答コードを返します。応答本文は空です。

ファイルコンテンツのアップロード

［ファイルコンテンツのアップロード］REST API は、ファイル ID に関連付けられているファイルのコンテンツをアップロードします。

BUNDLE、DB、または TEMP ストレージを指定した［ファイルコンテンツのアップロード］REST API を使用します。

この API は PUT メソッドを使用します。

要求 URL

［ファイルコンテンツのアップロード］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/<storage>/<fileId>/content
```

［ファイルコンテンツのアップロード］URL に対して、次の HTTP PUT 要求を行います。

```
PUT http://<host>:<port>/<context>/<database ID>/<storage>/<fileId>/content
```

サンプル API 要求

次のサンプル要求は、ファイル ID が TEMP_SVR1.1VDVS のファイルのコンテンツを TEMP ストレージにアップロードします:

```
PUT http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/TEMP/TEMP_SVR1.1VDVS/content

Test attachment content: file 1
```

次のサンプル要求は、ファイル ID が DB_SVR1.0JU1 のファイルのコンテンツを DB ストレージにアップロードします:

```
PUT http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/DB/DB_SVR1.0JU1/content
Content-Type: application/octet-stream
<file object (upload using REST client)>
```

次のサンプル要求は、ファイル ID が besMetadata のリソースバンドルファイルのコンテンツを BUNDLE ストレージにアップロードします:

```
PUT http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/BUNDLE/besMetadata/content
Content-Type: application/octet-stream
Body: binary stream – zip file with besMetadata bundle
```

サンプル API 応答

この API は、ファイルのコンテンツが正常にアップロードされた場合に 200 OK 応答コードを返します。応答本文は空です。

ファイルコンテンツの取得

［ファイルコンテンツの取得］REST API は、ファイル ID に関連付けられているファイルのコンテンツを返します。

BPM、BUNDLE、DB、または TEMP ストレージを指定した［ファイルコンテンツの取得］REST API を使用します。

この API は GET メソッドを使用します。

要求 URL

［ファイルコンテンツの取得］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/<storage>/<fileId>/content
```

［ファイルコンテンツの取得］URL に対して、次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/<context>/<database ID>/<storage>/<fileId>/content
```

サンプル API 要求

次のサンプル要求は、BPM ストレージ内でファイル ID が urn:b4p2:22203::file1.txt のファイルのコンテンツを返します:

```
GET http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/BPM/urn:b4p2:22203::file1.txt/content
```

注: BPM ストレージ内でタスク添付ファイルのファイル ID を取得するには、［タスクの読み取り］REST API を使用します。

次のサンプル要求は、DB ストレージ内でファイル ID が DB_SVR1.OJU1 のファイルのコンテンツを返します:

```
GET http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/DB/DB_SVR1.OJU1/content
```

注: レコードに添付するファイルのファイル ID を取得するには、［レコードの読み取り］REST API を使用します。

次のサンプル要求は、BUNDLE ストレージ内でファイル ID が besMetadata のリソースバンドルファイルのコンテンツを返します:

```
GET http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/BUNDLE/besMetadata/content
```

サンプル API 応答

次のサンプル応答は、BPM ストレージ内の TXT ファイルのコンテンツを示しています:

```
Test attachment content: file 1
```

次のサンプル応答は、BUNDLE ストレージ内のリソースバンドルファイルのコンテンツを示しています:

```
Content-Disposition → attachment; filename=besMetadata.zip
Content-Type → application/octet-stream
Transfer-Encoding → chunked
```

ファイルの削除

［ファイルの削除］REST API は、ファイルのメタデータおよびコンテンツを含む、ファイル ID に関連付けられているファイルを削除します。

BUNDLE、DB、または TEMP ストレージを指定した［ファイルの削除］REST API を使用します。

この API は DELETE メソッドを使用します。

要求 URL

［ファイルの削除］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/<storage>/<fileId>
```

［ファイルの削除］URL に対して、次の HTTP DELETE 要求を行います。

```
DELETE http://<host>:<port>/<context>/<database ID>/<storage>/<fileId>
```

サンプル API 要求

次のサンプル要求は、TEMP ストレージ内でファイル ID TEMP_SVR1.1VDVS に関連付けられているファイル（ファイルのメタデータおよびコンテンツを含む）を削除します:

```
DELETE http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/TEMP/TEMP_SVR1.1VDVS
```

次のサンプル要求は、DB ストレージ内でファイル ID DB_SVR1.OJU1 に関連付けられているファイル（ファイルのメタデータおよびコンテンツを含む）を削除します:

```
DELETE http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/DB/DB_SVR1.OJU1
```

次のサンプル要求は、BUNDLE ストレージ内でファイル ID が besMetadata のリソースバンドルファイル（ファイルのメタデータおよびコンテンツを含む）を削除します:

```
DELETE http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/BUNDLE/besMetadata
```

サンプル API 応答

この API は、ファイルが正常に削除された場合に 200 OK 応答コードを返します。応答本文は空です。

置換されたレコードのプレビュー

置換されたレコードのプレビュー REST API は、編集されて新しい値で置換されたビジネスエンティティレコードを返します。

この API は POST メソッドを使用します。

要求 URL

［上書きレコードのプレビュー］URL の形式は次のとおりです。

```
http://<host>:<port>/find-replace/<orsId>/<entity>/<preview{?firstRecord,order,recordsToReturn,returnTotal}>
```

［上書きレコードのプレビュー］URL に対して、次の HTTP POST 要求を行います。

```
POST http://<host>:<port>/find-replace/<orsId>/<entity>/<preview{?
firstRecord,order,recordsToReturn,returnTotal}>
```

パラメータ

編集されて新しい値で置換されたビジネスエンティティレコードを返すためのパラメータを指定します。

次の表に、要求本文内のパラメータを示します。

パラメータ	タイプ	説明
entity	パス	返すビジネスエンティティレコードの名前。
orsId	パス	置換されたビジネスエンティティレコードを返すデータベースのオペレーショナルリファレンスストア ID。
firstRecord	クエリ	オプション。レコードリストの行。
order	クエリ	オプション。フィールド名のカンマ区切りリスト。オプションでプレフィックス+または-を指定できます。プレフィックス+は結果を昇順でソートし、プレフィックス-は結果を降順でソートすることを指定します。デフォルトは+です。複数のパラメータを指定した場合、結果セットは、指定した最初のパラメータでまずソートされてから、次のパラメータでソートされます。
recordsToReturn	クエリ	オプション。返す行の数を指定します。
returnTotal	クエリ	オプション。true に設定すると、結果内のレコードの数が返されます。デフォルトは false です。

サンプル API 要求

次のサンプル要求は、編集および新しい値で上書きされたビジネスエンティティレコードを返します。

```
POST /cmx/find-replace/localhost-orcl-MDM_SAMPLE/Person/preview?returnTotal=true HTTP/1.1
Content-Type: application/json
Host: localhost:8080
Content-Length: 234

{
  "scope" : {
    "type" : "query",
    "filter" : "lastName='jones'"
  },
  "findReplace" : {
    "field" : "Emails.electAddrTypeCd",
    "find" : null,
    "replacement" : {
      "electronicType" : "EMAIL"
    }
  }
}
```

サンプル API 応答

次のサンプル応答は、上書きされたビジネスエンティティレコードを示しています。

```
HTTP/1.1 200 OK
Server: JBoss-EAP/7
Date: Mon, 13 Jan 2020 18:07:02 GMT
X-Powered-By: Undertow/1
Content-Type: application/json;charset=UTF-8
Content-Length: 4891

{
```

```
"link" : [ ],
"firstRecord" : 1,
"recordCount" : 6,
"pageSize" : 10,
"foundRecords" : 2,
"item" : [ {
  "origin" : {
    "key" : {
      "rowid" : "401924        "
    },
    "name" : "Person",
    "label" : "jones, alice",
    "object" : {
      "key" : {
        "rowid" : "426           "
      },
      "name" : "Emails",
      "label" : "Emails"
    }
  },
  "found" : false,
  "record" : {
    "rowidObject" : "426           ",
    "creator" : "admin",
    "createDate" : "2020-01-13T12:40:17.811-05:00",
    "updatedBy" : "admin",
    "lastUpdateDate" : "2020-01-13T12:40:18.583-05:00",
    "consolidationInd" : 4,
    "lastRowidSystem" : "SYSO          ",
    "hubStateInd" : 1,
    "label" : "Emails",
    "electronicAddress" : "alice@mail.com",
    "electAddrTypeCd" : {
      "electronicType" : "EMAIL"
    }
  }
}, {
  "origin" : {
    "key" : {
      "rowid" : "401925        "
    },
    "name" : "Person",
    "label" : "jones, bob",
    "object" : {
      "key" : {
        "rowid" : "427           "
      },
      "name" : "Emails",
      "label" : "Emails"
    }
  },
  "found" : false,
  "record" : {
    "rowidObject" : "427           ",
    "creator" : "admin",
    "createDate" : "2020-01-13T12:40:18.006-05:00",
    "updatedBy" : "admin",
    "lastUpdateDate" : "2020-01-13T12:40:18.655-05:00",
    "consolidationInd" : 4,
    "lastRowidSystem" : "SYSO          ",
    "hubStateInd" : 1,
    "label" : "Emails",
    "electronicAddress" : "bob@mail.com",
    "electAddrTypeCd" : {
      "electronicType" : "EMAIL"
    }
  }
}, {
  "origin" : {
    "key" : {
      "rowid" : "401926        "
```

```
      },
      "name" : "Person",
      "label" : "jones, charlie",
      "object" : {
        "key" : {
          "rowid" : "428           "
        },
        "name" : "Emails",
        "label" : "Emails"
      }
    },
    "found" : false,
    "record" : {
      "rowidObject" : "428           ",
      "creator" : "admin",
      "createDate" : "2020-01-13T12:40:18.216-05:00",
      "updatedBy" : "admin",
      "lastUpdateDate" : "2020-01-13T12:40:18.216-05:00",
      "consolidationInd" : 4,
      "lastRowidSystem" : "SYSO          ",
      "hubStateInd" : 1,
      "label" : "Emails",
      "electronicAddress" : "charlie@mail.com",
      "electAddrTypeCd" : {
        "electronicType" : "EMAIL"
      }
    }
  }, {
    "origin" : {
      "key" : {
        "rowid" : "401935        "
      },
      "name" : "Person",
      "label" : "jones, alice",
      "object" : {
        "key" : {
          "rowid" : "437           "
        },
        "name" : "Emails",
        "label" : "Emails"
      }
    },
    "found" : true,
    "record" : {
      "rowidObject" : "437           ",
      "creator" : "admin",
      "createDate" : "2020-01-13T13:07:02.187-05:00",
      "updatedBy" : "admin",
      "lastUpdateDate" : "2020-01-13T13:07:02.187-05:00",
      "consolidationInd" : 4,
      "lastRowidSystem" : "SYSO          ",
      "hubStateInd" : 1,
      "label" : "Emails",
      "electronicAddress" : "alice@mail.com"
    }
  }, {
    "origin" : {
      "key" : {
        "rowid" : "401936        "
      },
      "name" : "Person",
      "label" : "jones, bob",
      "object" : {
        "key" : {
          "rowid" : "438           "
        },
        "name" : "Emails",
        "label" : "Emails"
      }
    },
    "found" : true,
```

```
    "record" : {
      "rowidObject" : "438            ",
      "creator" : "admin",
      "createDate" : "2020-01-13T13:07:02.389-05:00",
      "updatedBy" : "admin",
      "lastUpdateDate" : "2020-01-13T13:07:02.390-05:00",
      "consolidationInd" : 4,
      "lastRowidSystem" : "SYSO           ",
      "hubStateInd" : 1,
      "label" : "Emails",
      "electronicAddress" : "bob@mail.com"
    }
  },{
    "origin" : {
      "key" : {
        "rowid" : "401937        "
      },
      "name" : "Person",
      "label" : "jones, charlie",
      "object" : {
        "key" : {
          "rowid" : "439           "
        },
        "name" : "Emails",
        "label" : "Emails"
      }
    },
    "found" : false,
    "record" : {
      "rowidObject" : "439            ",
      "creator" : "admin",
      "createDate" : "2020-01-13T13:07:02.587-05:00",
      "updatedBy" : "admin",
      "lastUpdateDate" : "2020-01-13T13:07:02.587-05:00",
      "consolidationInd" : 4,
      "lastRowidSystem" : "SYSO           ",
      "hubStateInd" : 1,
      "label" : "Emails",
      "electronicAddress" : "charlie@mail.com",
      "electAddrTypeCd" : {
        "electronicType" : "EMAIL"
      }
    }
  } ]
}
```

検索と置換の更新

上書きレコードの更新 REST API は、編集して新しい値で置換するビジネスエンティティのフィルタリング済みレコードを返します。

この API は POST メソッドを使用します。

要求 URL

［上書きレコードの更新］URL の形式は次のとおりです。

```
http://<host>:<port>/find-replace/<orsId>/<entity>/{?systemName}
```

［上書きレコードの更新］URL に対して、次の HTTP POST 要求を行います。

```
POST http://<host>:<port>/find-replace/<orsId>/<entity>/{?systemName}
```

パラメータ

ビジネスエンティティのフィルタされたレコードを更新するためのパラメータを指定します。

次の表に、これらのパラメータについて説明します。

パラメータ	タイプ	説明
entity	パス	返すビジネスエンティティレコードの名前。
orsId	パス	置換されたビジネスエンティティレコードを返すデータベースのオペレーショナルリファレンスストア ID。
systemName	クエリ	オプション。ソースシステムの名前。
If-Unmodified-Since	ヘッダー	オプション。以前に置換されたレコードのスキップを可能にします。

サンプル API 要求

次のサンプル要求は、編集して新しい値で置換するビジネスエンティティのフィルタリング済みレコードを更新します。

```
POST /cmx/find-replace/localhost-orcl-MDM_SAMPLE/Person?systemName=Admin HTTP/1.1
Content-Type: application/json
Host: localhost:8080
Content-Length: 234

{
  "scope" : {
    "type" : "query",
    "filter" : "lastName='jones'"
  },
  "findReplace" : {
    "field" : "Emails.electAddrTypeCd",
    "find" : null,
    "replacement" : {
      "electronicType" : "EMAIL"
    }
  }
}
```

サンプル API 応答

次のサンプル応答は、ビジネスエンティティの更新されたフィルタリング済みレコードを示しています。

```
HTTP/1.1 200 OK
Server: JBoss-EAP/7
X-Powered-By: Undertow/1
Content-Type: application/json;charset=UTF-8
Content-Length: 1339

{
  "successful" : 2,
  "failed" : 0,
  "skipped" : 0,
  "item" : [ {
    "object" : {
      "Person" : {
        "key" : {
          "rowid" : "401935        "
        },
        "rowidObject" : "401935        ",
        "Emails" : {
```

```
          "link" : [ ],
          "item" : [ {
            "key" : {
              "rowid" : "437",
              "sourceKey" : "480000041000"
            },
            "rowidObject" : "437",
            "electAddrTypeCd" : {
              "key" : {
                "rowid" : "1             "
              },
              "rowidObject" : "1             "
            }
          } ]
        }
      }
    },
    "parameters" : { }
  }, {
    "object" : {
      "Person" : {
        "key" : {
          "rowid" : "401936        "
        },
        "rowidObject" : "401936        ",
        "Emails" : {
          "link" : [ ],
          "item" : [ {
            "key" : {
              "rowid" : "438",
              "sourceKey" : "480000043000"
            },
            "rowidObject" : "438",
            "electAddrTypeCd" : {
              "key" : {
                "rowid" : "1             "
              },
              "rowidObject" : "1             "
            }
          } ]
        }
      }
    },
    "parameters" : { }
  } ]
}
```


新規ファイルのインポート

新規ファイルのインポート REST API は、新規ファイルのインポートジョブを開始し、ファイルからビジネスエンティティをインポートします。

この API は POST メソッドを使用します。

要求 URL

［新規ファイルのインポート］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<orsId>/<job>
```

［新規ファイルのインポート］URL に対して、次の HTTP POST 要求を行います。

```
POST http://<host>:<port>/<context>/<orsId>/<job>
```

パラメータ

要求本文の新しいファイルインポートのための次のパラメータを指定します。

次の表に、これらのパラメータについて説明します。

パラメータ	タイプ	説明
orsID	パス	新しいファイルをインポートする先のオペレーショナルリファレンスストアデータベース ID。
fileID	クエリ	オプション。インポートされるファイルのファイル ID。既存のレコードを更新している場合にこのオプションを使用します。
mappingID	クエリ	オプション。インポートされるファイルのマッピング ID。以前インポートしたファイルを更新している場合にこのオプションを使用します。
systemName	クエリ	オプション。ファイルをインポートするシステムユーザーの名前。

サンプル API 要求

次のサンプル要求は、新規ファイルインポートジョブを開始し、ビジネスエンティティをインポートします。

```
POST /cmx/beimport/localhost-orcl-MDM_SAMPLE/job HTTP/1.1
Content-Type: application/json
Host: localhost:8080
Content-Length: 94

{
  "fileId" : "TEMP_SVR1.1G7UW",
  "mappingId" : "SVR1.1G7UX",
  "systemName" : "Admin"
}
```

サンプル API 応答

次のサンプル応答は、新規ファイルが正常にインポートされたことを示しています。

```
HTTP/1.1 201 Created
Server: JBoss-EAP/7
X-Powered-By: Undertow/1
Location: http://localhost:8080/cmx/jobcontrol/localhost-orcl-MDM_SAMPLE/group/SVR1.1G7UY
```

一致ファイルのインポート

一致ファイルのインポート REST API は、ファイルをインポートする前に、新しいビジネスエンティティを既存のビジネスエンティティと照合します。インポート操作が完了した後、一致したビジネスエンティティは新しいデータで更新されます。

重複ビジネスエンティティ、一意のビジネスエンティティまたは重複および一意のビジネスエンティティをインポートできます。

この API は POST メソッドを使用します。

要求 URL

［一致したファイルのインポート］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<orsId>/<aftermatch>
```

［一致したファイルのインポート］URL に対して、次の HTTP POST 要求を行います。

```
POST http://<host>:<port>/<context>/<orsId>/<aftermatch>
```

パラメータ

ファイルをインポートする前に、新しいビジネスエンティティを既存のビジネスと一致させるためのパラメータを指定します。

次の表に、これらのパラメータについて説明します。

パラメータ	タイプ	説明
orsId	パス	既存のレコードの一致後、ファイルをインポートするオペレーショナルリファレンスストアデータベース ID。
filter	クエリ	オプション。次のフィルタ値を使用します: withMatches: レコードが既存のレコードと一致する場合は true を入力します。 withoutMatches: レコードが既存のレコードと一致しない場合は false を入力します。
jobGroupControlId	クエリ	オプション。一致ジョブグループ ID。
systemName	クエリ	オプション。ソースシステム名。

サンプル API 要求

次のサンプル要求は、ファイルをインポートする前に、新しいビジネスエンティティを既存のビジネスエンティティと照合します。

```
POST /cmx/beimport/{ors}/aftermatch
{
    jobGroupControlId: "...",
    systemName: "SFA",
    filter: {
      withMatches: true,
      withoutMatches: false
    }
}
```

サンプル API 応答

API は、新しいビジネスエンティティを既存のビジネスエンティティと正常に照合し、ファイルをインポートした後、200 OK 応答コードを返します。応答本文は空です。

ファイルプロパティの取得

ファイルプロパティの取得 REST API は、インポートされたファイルのファイルプロパティを返します。例えば、この API は区切り文字、テキスト修飾子、小数位の区切り文字のファイルプロパティを返します（ファイルをインポートしたときにこれらのプロパティを選択した場合）。

この API は GET メソッドを使用します。

注: この API は、5 行以上存在する場合にのみ、区切り文字を含む既存のファイルプロパティを返します。

要求 URL

［ファイルプロパティの取得］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<orsId>/<fileId>/<properties{query}>
```

［ファイルプロパティの取得］URL に対して次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/<context>/<orsId>/<fileId>/<properties{query}>
```

パラメータ

インポートされたファイルのファイルプロパティを取得するためのパラメータを指定します。

次の表に、これらのパラメータについて説明します。

パラメータ	タイプ	説明
orsId	パス	オペレーショナルリファレンスストアデータベース ID。
fileId	パス	インポートされるファイルのファイル ID。
suggest	クエリ	オプション。true に設定すると、要求はインポートされたファイルに基づきファイルのプロパティを返します。それ以外の場合、要求はインポートされたファイルのデータに基づき推奨するファイルのプロパティを返します。

サンプル API 要求

次のサンプル要求は、インポートされたファイルのファイルプロパティを検出して返します。

```
GET /cmx/file/parser/localhost-orcl-MDM_SAMPLE/TEMP_SVR1.1G7UW/properties?suggest=true HTTP/1.1
Content-Type: application/json
Host: localhost:8080
```

サンプル API 応答

次のサンプル応答は、インポートされたファイルのファイルプロパティを返した後のヘッダーと本文を示しています。

```
HTTP/1.1 200 OK
Server: JBoss-EAP/7
X-Powered-By: Undertow/1
Content-Type: application/json;charset=UTF-8
Content-Length: 370

{
  "confidence" : "HIGH",
  "type" : "CSV",
  "properties" : {
    "headerRow" : 1,
    "dataStartsAtRow" : 2,
    "regionalSettings" : {
      "locale" : "en-CA",
      "datePattern" : "MM-dd-yyyy",
      "decimalSeparator" : "."
      "thousandsSeparator" : ","
    },
    "delimiter" : ",",
    "encoding" : "UTF-8",
```

```
    "textQualifier" : "\""
  }
}
```

ファイルプロパティの保存

ファイルプロパティの保存 REST API は、インポートされたファイルプロパティを解析して保存します。

この API は PUT メソッドを使用します。

要求 URL

［ファイルプロパティの保存］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<orsId>/<fileId>/<properties>
```

［ファイルプロパティの保存］URL に対して次の HTTP PUT 要求を行います。

```
PUT http://<host>:<port>/<context>/<orsId>/<fileId>/<properties>
```

パラメータ

インポートされたファイルのファイルプロパティを保存するためのパラメータを指定します。

次の表に、要求本文内のパラメータを示します。

パラメータ	タイプ	説明
orsId	パス	オペレーショナルリファレンスストアデータベース ID。
fileId	パス	インポートされるファイルのファイル ID。
headerRow	クエリ	オプション。必要なカラムの行ヘッダー番号。
dataStartAtRow	クエリ	オプション。データのインポート元のソースファイルの行。
locale	クエリ	オプション。ローカリゼーション言語。
dataPattern	クエリ	オプション。インポートファイルの日付フィールドの日付形式。
decimalSeparator	クエリ	オプション。小数位の区切り文字カンマ（,）またはピリオド（.）を入力します。
thousandsSeparator	クエリ	オプション。桁区切り。カンマ（,）またはピリオド（.）を入力します。
delimiter	クエリ	オプション。インポートファイル内でデータ値の区切りに使用されている文字。事前に定義された区切り文字を入力するかカスタムの区切り文字を定義します。
encoding	クエリ	オプション。Unicode エンコード標準。
textQualifier	クエリ	オプション。文字列を囲むためにファイルで使用される記号。

サンプル API 要求

次のサンプル要求は、ファイルプロパティを解析して保存します。

```
PUT /cmx/file/parser/localhost-orcl-MDM_SAMPLE/TEMP_SVR1.1G7UW/properties HTTP/1.1
Content-Type: application/json
Host: localhost:8080
Content-Length: 344

{
  "type" : "CSV",
  "properties" : {
    "headerRow" : 1,
    "dataStartsAtRow" : 2,
    "regionalSettings" : {
      "locale" : "en-CA",
      "datePattern" : "MM-dd-yyyy",
      "decimalSeparator" : "."
      "thousandsSeparator" : ","
    },
    "delimiter" : ",",
    "encoding" : "UTF-8",
    "textQualifier" : "\""
  }
}
```

サンプル API 応答

次のサンプル応答は、ファイルプロパティが保存されていることを示しています。

```
HTTP/1.1 200 OK
Server: JBoss-EAP/7
X-Powered-By: Undertow/1
```

ファイルプロパティの戻り

［ファイルプロパティの戻り］REST API は、存続済みファイルのプロパティを返します。

この API は GET メソッドを使用します。

要求 URL

［ファイルプロパティの戻り］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<orsId>/<fileId>/<properties>
```

［ファイルプロパティの戻り］URL に対して次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/<context>/<orsId>/<fileId>/<properties>
```

パラメータ

永続ファイルのプロパティを返すためのパラメータを指定します。

次の表に、これらのパラメータについて説明します。

パラメータ	タイプ	説明
orsId	パス	ファイルのプロパティの取得元のオペレーショナルリファレンスストア ID。
fileId	パス	インポートされるファイルのファイル ID。
suggest	クエリ	オプション。true に設定すると、要求は永続ファイルのプロパティを返します。それ以外の場合、要求はすべてのプロパティを返します。

サンプル API 要求

次のサンプル要求は、存続済みファイルのプロパティを返します。

```
GET /cmx/file/parser/localhost-orcl-MDM_SAMPLE/TEMP_SVR1.1G7UW/properties?suggest=true HTTP/1.1
Content-Type: application/json
Host: localhost:8080
```

サンプル API 応答

次のサンプル応答は、存続済みファイルのプロパティを示しています。

```
HTTP/1.1 200 OK
Server: JBoss-EAP/7
X-Powered-By: Undertow/1
Content-Type: application/json;charset=UTF-8
Content-Length: 370

{
  "confidence" : "HIGH",
  "type" : "CSV",
  "properties" : {
    "headerRow" : 1,
    "dataStartsAtRow" : 2,
    "regionalSettings" : {
      "locale" : "en-CA",
      "datePattern" : "MM-dd-yyyy",
      "decimalSeparator" : ".",
      "thousandsSeparator" : ","
    },
    "delimiter" : ",",
    "encoding" : "UTF-8",
    "textQualifier" : "\""
  }
}
```

解析済みファイルのプレビュー

解析済みファイルのプレビュー REST API は、ファイルのプロパティを解析し、要求本文に指定されたプロパティに基づくフラットレコードを返します。この API は、CSV および Excel ファイルを解析します。

この API は POST メソッドを使用します。

要求 URL

[解析済みファイルのプレビュー API] URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<orsId>/<fileId>/preview{?pageSize}
```

[解析済みファイルのプレビュー API] URL に対して、次の HTTP POST 要求を行います。

```
POST http://<host>:<port>/<context>/<orsId>/<fileId>/preview{?pageSize}
```

パラメータ

ファイルのプロパティを解析し、要求本文に指定されたプロパティに基づくフラットレコードを返すためのパラメータを指定します。

次の表に、要求本文内のパラメータを示します。

パラメータ	タイプ	説明
orsId	パス	データベースに保存されたファイルのオペレーショナルリファレンスストアデータベース ID。
fileId	パス	インポートされるファイルのファイル ID。
pagesize	クエリ	オプション。返すレコード行の数。

サンプル API 要求

次のサンプル要求は、ファイルのプロパティを解析し、要求本文で提供されるプロパティに基づいてフラットレコードを返します。

```
POST /cmx/file/parser/localhost-orcl-MDM_SAMPLE/TEMP_SVR1.1G7UW/preview HTTP/1.1
Content-Type: application/json
Host: localhost:8080
Content-Length: 345

{
  "type" : "CSV",
  "properties" : {
    "headerRow" : 1,
    "dataStartsAtRow" : 2,
    "regionalSettings" : {
      "locale" : "en-CA",
      "datePattern" : "d MMM, yyyy",
      "decimalSeparator" : "."
      "thousandsSeparator" : ","
    },
    "delimiter" : ",",
    "encoding" : "UTF-8",
    "textQualifier" : "\""
  }
}
```

サンプル API 応答

次のサンプル応答は、要求本文で提供されるプロパティに基づいて取得されたフラットレコードを示しています。

```
HTTP/1.1 200 OK
Server: JBoss-EAP/7
X-Powered-By: Undertow/1
Content-Type: application/json;charset=UTF-8
Content-Length: 381
```

```
{
  "header" : [ "first_name", "last_name", "job_title", "gender", "birth_date" ],
  "data" : [ [ "alice", "smith", "developer", "F", "01-01-1950" ], [ "bob", "smith", "tester", "M",
"02-02-1960" ], [ "charlie", "smith", "manager", "X", "03-03-1970" ], [ "dave", "smith", "accountant", "M",
"ten years ago" ], [ "eve", "smith", "", "F", "04-04-1990" ] ],
  "numberOfRows" : 5
}
```


解析済みファイルの取得

解析済みファイルの取得 REST API は、ファイルプロパティを解析し、そのプロパティに基づくフラットレコードを返します。この API は、CSV および Excel ファイルを解析します。

この API は GET メソッドを使用します。

要求 URL

[解析済みファイルプロパティの取得] URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<orsId>/<fileId>/preview{?pageSize}
```

[解析済みファイルプロパティの取得] URL に次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/<context>/<orsId>/<fileId>/preview{?pageSize}
```

パラメータ

ファイルのプロパティを解析し、ファイルのプロパティに基づくフラットレコードを返すためのパラメータを指定します。

次の表に、これらのパラメータについて説明します。

パラメータ	タイプ	説明
orsId	パス	オペレーショナルリファレンスストアデータベース ID。
fileId	パス	インポートされるファイルのファイル ID。
pagesize	クエリ	オプション。返すレコード行の数。

サンプル API 要求

次のサンプル要求は、ファイルを解析し、ファイルに関連付けられたプロパティに基づいてフラットレコードを返します。

```
GET /cmx/file/parser/localhost-orcl-MDM_SAMPLE/TEMP_SVR1.1G7UW/preview HTTP/1.1
Content-Type: application/json
Host: localhost:8080
```

サンプル API 応答

次のサンプル応答は、ファイルに関連付けられたプロパティに基づいて解析されたフラットレコードを示しています。

```
HTTP/1.1 200 OK
Server: JBoss-EAP/7
X-Powered-By: Undertow/1
Content-Type: application/json;charset=UTF-8
Content-Length: 381
```

```
{
  "header" : [ "first_name", "last_name", "job_title", "gender", "birth_date" ],
  "data" : [ [ "alice", "smith", "developer", "F", "01-01-1950" ], [ "bob", "smith", "tester", "M",
"02-02-1960" ], [ "charlie", "smith", "manager", "X", "03-03-1970" ], [ "dave", "smith", "accountant", "M",
"ten years ago" ], [ "eve", "smith", "", "F", "04-04-1990" ] ],
  "numberOfRows" : 5
}
```

ファイルのインポートの解析エラー

ファイルのインポートの解析エラー REST API は、ファイルのインポート操作からの解析エラーを CSV ファイル形式で返します。

この API は GET メソッドを使用します。

要求 URL

［ファイル解析エラーのインポート］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<orsId>/<parse-errors>/{jobGroupControlId}{?entity}
```

［ファイル解析エラーのインポート］URL に対して次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/<context>/<orsId>/<parse-errors>/{jobGroupControlId}{?entity}
```

パラメータ

ファイルのインポート操作からの解析エラーを CSV ファイルで返すためのパラメータを指定します。

次の表に、要求本文内のパラメータを示します。

パラメータ	タイプ	説明
orsId	パス	ファイルのインポート操作からの解析エラーを返すための オペレーショナルリファレンスストアデータベース ID。
jobGroupControlId	クエリ	オプション。バッチジョブグループ ID。
entity	クエリ	オプション。ビジネスエンティティの名前。

サンプル API 要求

次のサンプル要求は、ファイルのインポート操作から解析エラーを返します。

```
GET /cmx/beimport/localhost-orcl-MDM_SAMPLE/parse-errors/SVR1.1G7UY HTTP/1.1
Host: localhost:8080
```

サンプル API 応答

次のサンプル応答は、ファイルのインポート操作からの解析エラーを示しています。

```
HTTP/1.1 200 OK
Server: JBoss-EAP/7
X-Powered-By: Undertow/1
Content-Type: text/csv
Content-Length: 200

first_name,last_name,job_title,gender,birth_date,ROW,COLUMN,ERROR,ENTITY
dave,smith,accountant,M,ten years ago,3,birth_date,Failed to convert value [ten years ago] for field
[birthdate].,PersonView
```

ファイルのインポートのロードエラー

ファイルのインポートのロードエラー REST API は、ファイルのインポート操作からのエラーを CSV ファイル形式で返します。

この API は GET メソッドを使用します。

要求 URL

［ファイルのロードエラーのインポート］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<orsId>/<load-errors>/{jobGroupControlId}{?entity}
```

［ファイルのロードエラーのインポート］URL に対して次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/<context>/<orsId>/<load-errors>/{jobGroupControlId}{?entity}
```

パラメータ

ファイルのインポート操作からのロードエラーを CSV ファイルで返すためのパラメータを指定します。

次の表に、要求本文内のパラメータを示します。

パラメータ	タイプ	説明
orsId	パス	ファイルのインポート操作からのロードエラーを返すためのオペレーショナルリファレンスストアデータベース ID。
jobGroupControlId	クエリ	オプション。バッチジョブグループ ID。
entity	クエリ	オプション。ビジネスエンティティの名前。

サンプル API 要求

次のサンプル要求は、ファイルのインポート操作からロードのエラーを返します。

```
GET /cmx/beimport/localhost-orcl-MDM_SAMPLE/load-errors/SVR1.1G7UY HTTP/1.1
Host: localhost:8080
```

サンプル API 応答

次のサンプル応答は、ファイルのインポート操作からのロードのエラーを示しています。

```
HTTP/1.1 200 OK
Server: JBoss-EAP/7
X-Powered-By: Undertow/1
Content-Type: text/csv
Content-Length: 257

first_name,last_name,job_title,gender,birth_date,ROW,COLUMN,ERROR,ENTITY
charlie,smith,manager,X,03-03-1970,2,,SIP-50112: Could not run WriteCO business entity service.  A record for
[genderCd] could not be found in business entity [LUGender].,PersonView
```

昇格のプレビュー

保留中の変更を昇格させると、［昇格のプレビュー］REST API は結果として生成されるレコードのプレビューを返します。

この API は GET メソッドを使用します。保留中の変更をレコードに適用するとレコードがどのようになるかを確認できます。API 応答には、値が新しくなったレコードと変更の要約が古い値とともに含まれます。この

API は、ユーザーが削除するデータについての情報は返しません。保留中の変更の相互作用 ID を URL に指定します。

要求 URL

［昇格のプレビュー］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID><business entity>/<rowId>?
action=previewPromote&interactionID=<interaction ID>
```

［昇格のプレビュー］URL に対して、次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/<context>/<database ID><business entity>/<rowId>?
action=previewPromote&interactionID=<interaction ID>
```

クエリパラメータ

保留中の変更の相互作用 ID は、URL の必須パラメータです。

次の表にクエリパラメータを示します。

パラメータ	説明
contentMetadata	マージのプレビューのメタデータ。カンマ区切りのリストを指定します。 以下の値を使用できます。 - BVT。マージのプレビューで使用する最も信頼できる値を含むレコードの行 ID を指定します。相互参照レコードおよび元のレコード ID についての情報を返します。 - MERGE。マージするレコードの行 ID を指定します。子孫レコードがマージされた方法についての情報を返します。
interactionId	保留中の変更の相互作用 ID。
effectiveDate	オプション。変更のプレビュー対象である日付。このパラメータはタイムラインが有効になったベースオブジェクトに使用します。

関連項目：

- 「UTC での日付と時刻の形式」 (ページ 34)

サンプル API 要求

次のサンプル要求は、Person ビジネスエンティティ内のルートレコードのプレビューを作成します。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/233?
action=previewPromote&interactionId=72300000001000
```

サンプル API 応答

次のサンプル応答では、新しい値と、古い値の変更要約とともに、レコードのプレビューが返されています。

```
{
    "rowidObject": "233           ",
    "creator": "admin",
    "createDate": "2008-08-12T02:15:02-07:00",
    "updatedBy": "admin",
    "lastUpdateDate": "2015-07-14T03:42:38.778-07:00",
    "consolidationInd": "1",
    "lastRowidSystem": "SYS0          ",
    "dirtyIndicator": "0",
    "interactionId": "72300000001000",
    "hubStateInd": "1",
```

```
    "label": "LLOYD,BOB",
    "partyType": "Person",
    "lastName": "LLOYD",
    "firstName": "BOB",
    "displayName": "BOB LLOYD",
    "preferredPhone": {
        "rowidObject": "164           ",
        "$original": {
            "rowidObject": "164           "
        }
    },
    "$original": {
        "label": "DUNN,LLOYD",
        "lastName": "DUNN",
        "firstName": "LLOYD",
        "displayName": "LLOYD DUNN"
    }
}
```

昇格

［昇格］REST API は、レコードの保留中の変更をすべて、変更要求の相互作用 ID に基づいて昇格させます。

この API は POST メソッドを使用します。相互作用 ID をクエリパラメータとして指定します。

要求 URL

［昇格］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<rowId of the root record>?
action=promote&interactionId=<interaction ID>
```

［昇格］URL に対して、次の HTTP POST 要求を行います。

```
POST http://<host>:<port>/<context>/<database ID>/<business entity>/<rowId of the root record>?
action=promote&interactionId=<interaction ID>
```

クエリパラメータ

保留中の変更の相互作用 ID は必須パラメータです。この API は、保留中の変更に関連しているレコードをすべて見つけるために、相互作用 ID を使用します。

サンプル API 要求

次のサンプル要求は、保留中の変更をすべて、変更要求の相互参照 ID に基づいて昇格させます。

```
POST http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/1038246?
action=promote&interactionId=69120000294000
```

サンプル API 応答

次のサンプル応答には、保留中の変更が昇格された後のレコードの行 ID が含まれています。

```
{
            Person: {
                        rowidObject: "1038246"
                }
}
```

保留中の削除

［保留中の削除］REST API は、変更要求の相互作用 ID に基づいて、レコードに対する保留中の変更をすべて削除します。

この API は DELETE メソッドを使用し、レコードの行 ID を返します。

要求 URL

［保留中の削除］URL のフォーマットは次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<rowid>?
action=deletePending&interactionId=<interaction ID>
```

［保留中の削除］URL に対して、次の DELETE 要求を行います。

```
DELETE http://<host>:<port>/<context>/<database ID>/<business entity>/<rowid>?
action=deletePending&interactionId=<interaction ID>
```

クエリパラメータ

削除する保留中の変更の相互作用 ID を指定します。

サンプル API 要求

次のサンプル要求は、変更要求の相互作用 ID に基づいて保留中の変更をすべて削除します。

```
DELETE http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/233?
action=deletePending&interactionId=72300000001000
```

サンプル API 応答

次のサンプル応答には、保留中の変更が削除された後のレコードの行 ID が含まれています。

```
{
    Person: {
        rowidObject: "233"
    }
}
```

マージのプレビュー

2 つ以上のルートレコードをマージすると、［マージのプレビュー］REST API は統合されたルートレコードのプレビューを返します。

この API は、マージされたレコードのプレビューを返すために、POST メソッドを使用し、ルートレコードとフィールドレベルのオーバーライドのリストを受け入れます。ターゲットレコードの行 ID は必須パラメータです。

要求 URL

［マージのプレビュー］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<rowid of the record>?action=previewMerge
```

［マージのプレビュー］URL 形式は、返す子レベルの数を指定します。

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<rowid of the record>?
action=previewMerge&depth=2
```

［マージのプレビュー］URL に対して、次の HTTP POST 要求を行います。

```
POST http://<host>:<port>/<context>/<database ID>/<business entity>/<rowid of the record>?action=previewMerge
```

注: 要求本文で、keys プロパティを追加し、ターゲットレコードとマージするルートレコードを指定します。

子レコードの一致をオーバーライドするには、contentMetadata パラメータを追加します。

```
POST http://<host>:<port>/<context>/<database ID>/<business entity>/<rowid of the record>?
action=previewMerge&contentMetadata=MERGE/<json/xml>
```

注: 要求本文で、overrides プロパティを追加して、マージのオーバーライドを指定します。

要求と一緒に送信するデータのメディアタイプを指定するには、Content-Type ヘッダーを追加します。

```
POST http://<host>:<port>/<context>/<database ID>/<business entity>/<rowid of the record>?action=previewMerge
Content-Type: application/<json/xml>
```

クエリパラメータ

ターゲットレコードの行 ID は必須パラメータです。

以下のクエリパラメータを使用できます。

パラメータ	説明
contentMetadata	マージのプレビューのメタデータ。カンマ区切りのリストを指定します。 以下の値を使用できます。 - BVT。優先される相互参照レコードおよび元のレコード ID についての情報を返します。 - MERGE。子孫レコードがマージされた方法についての情報を返します。
depth	返す子レベルの数。
effectiveDate	プレビューの生成対象となる日付。

関連項目：

- 「UTC での日付と時刻の形式」 (ページ 34)

要求本文

始める前に、[一致するレコードの読み取り] API を使用し、どの一致するレコードを元のルートレコードとマージできるかを判断できます。[マージのプレビュー] API のレコードのリストを要求本文で送信します。

ルートレコードでフィールドの値をオーバーライドできます。例えば、一致したルートレコードに名（ファーストネーム）の正しいスペルが含まれていない場合、要求本文で正しい名を指定できます。また、一致したレコードを削除したり、他の一致レコードを指定したりすることもできます。

要求本文で、次のプロパティを使用します。

プロパティ/要素	タイプ	説明
keys	配列	必須。マージする、一致したルートレコードの順序付きリスト。レコードは行 ID によって、またはソースシステムとソースキーの組み合わせによって識別できます。
overrides	オブジェクト	ルートレコードおよび一致子レコードで、フィールドの値をオーバーライドします。
MERGE	オブジェクト	マージする子レコードのフィールド値をオーバーライドします。overrides オブジェクト内で子レコードのタイプを追加してから、MERGE オブジェクトを追加します。

次の JSON コードサンプルは、ターゲットルートレコードとマージするルートレコードを識別します。

```
{
  keys: [
    {
       rowid: "P2"
    }
  ]
}
```

次のコードは、Party ルートレコードのフィールドをオーバーライドする方法、および Telephone 子レコードのマージ候補をオーバーライドする方法を示します。

```
{
  keys: [
    {
      rowid: "P2"
    }
  ]
  overrides: {
    Party: {
      rowidObject: "P1",
      firstName: "Serge",   //override the value for the first name
      Telephone: {   // override which Telephone child records to merge
        item:[
          {
            rowidObject: "T1",
            MERGE: {
              item: [   // to remove the original merge candidates, specify null
                null,
                null
              ],
              $original: {
                item: [
                  {key:{rowid: "T2"}},
                  {key:{rowid: "T3"}}
                ]
              }
            }
          },
          {
            rowidObject: "T4",
            MERGE: {
              item: [   // to add or change merge candidates, specify matched records
                {key:{rowid: "T2"}}
              ],
            $original: {
              item: [
                null
              ]
            }
```

```
          }
        ]
      }
    }
  }
}
```

サンプル API 要求

次のサンプル要求は、統合されたレコードのプレビューを返します。

```
POST http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/2478245?action=previewMerge
```

```
{
  keys: [
    {
        rowid: "2478246"
    },
   {
        rowid: "2478230"
    }
  ],
  overrides: {
    Person: {
       firstName: "Charlie"
    }
  }
}
```

サンプル API 応答

次のサンプル応答では、統合されたレコードのプレビューが示されています。

```
{
    "Person": {
        "rowidObject": "2478245       ",
        "partyType": "Person",
        "lastName": "Smith",
        "firstName": "Charlie",
        "displayName": "ALICE SMITH"
    }
}
```

保留中のマージの更新

保留中のマージタスクの一部であるレコードに、レコード値のオーバーライドなどの変更を保存するには、[保留中のマージの更新] REST API を使用します。この API は、レコードの相互作用 ID に基づいて変更を保存します。

この API は POST メソッドを使用します。

要求 URL

要求 URL のパスコンポーネントには、ターゲットレコードの行 ID が含まれる必要があります。

[保留中のマージの更新] URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/<business entity>
/<rowid of the root record>?
action=updatePendingMerge&interactionId=<interaction ID>
```

[保留中のマージの更新] URL に対して、次の HTTP POST 要求を行います。

```
POST http://<host>:<port>/<context>/<database ID>/<business entity>
/<rowid of the root record>?
```

```
action=updatePendingMerge&interactionId=<interaction ID>
```

要求本文で、キーを追加し、ターゲットルートレコードとマージするルートレコードを指定します。一致をオーバーライドする子レコードを指定することもできます。

クエリパラメータ

次の表に、URL に使用できるクエリパラメータを示します。

パラメータ	説明
action	必須。フィールドレベルのオーバーライドなどの変更を、保留中のマージタスクに保存します。updatePendingMerge に設定し、interactionId パラメータを設定してこのパラメータを使用します。 例えば、次のクエリを使用すると、マージアクションが保留中の Person ビジネスエンティティレコードに変更が保存されます。 `Person?action=updatePendingMerge&interactionId`
interactionId	必須。保留中のマージタスクの相互作用 ID。

要求本文

[保留中のマージの更新] API を使用する前に、[一致するレコードの読み取り] REST API を使用して、ターゲットルートレコードとマージできる一致レコードを判断します。[保留中のマージの更新] API の要求本文で、レコードのリストを送信します。

ルートレコードでフィールドの値をオーバーライドできます。例えば、一致したルートレコードに名（ファーストネーム）の正しいスペルが含まれていない場合、要求本文で正しい名を指定できます。また、一致したレコードを削除したり、他の一致レコードを指定したりすることもできます。

要求本文で、次のプロパティを使用します。

プロパティ/要素	タイプ	説明
keys	配列	必須。マージする、一致したルートレコードの順序付きリスト。レコードは行 ID によって、またはソースシステムとソースキーの組み合わせによって識別できます。
overrides	オブジェクト	ルートレコードおよび一致子レコードで、フィールドの値をオーバーライドします。
MERGE	オブジェクト	マージする子レコードのフィールド値をオーバーライドします。overrides オブジェクト内で子レコードのタイプを追加してから、MERGE オブジェクトを追加します。

次の JSON コードサンプルは、ターゲットルートレコードとマージする 2 つのルートレコードを識別します。

```
{
  keys: [
    {rowid: "2478246"},
    {rowid: "2478230"}
  ]
}
```

次のサンプル要求の本文は、Party ルートレコードのフィールドをオーバーライドする方法、および Telephone 子レコードの一致レコードをオーバーライドする方法を示します。

```
{
  keys: [
    {
      rowid: "2478246"
    }
  ]
  overrides: {
    Party: {
      rowidObject: "2478230",
      firstName: "Charlie",   //Override the value for the first name
      Telephone: {   // Specifies the Telephone child records to merge
        item:[
          {
            rowidObject: "2511",
            MERGE: {
              item: [   // To remove the original merge candidates, specify null
                null,
                null
              ],
              $original: {
                item: [
                  {key:{rowid: "2822"}},
                  {key:{rowid: "2733"}}
                ]
              }
            }
          },
          {
            rowidObject: "2644",
            MERGE: {
              item: [   // To add or change merge candidates, specify matched records
                {key:{rowid: "2822"}}
              ],
            $original: {
              item: [
                null
              ]
            }
          }
        ]
      }
    }
  }
}
```

サンプル API 要求

次のサンプル要求は、値が Charlie のターゲットレコードの、名前フィールドをオーバーライドします。

```
POST http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/2478245?
action=updatePendingMerge&interactionId=3982462873645
```

```
{
  keys: [
    {
        rowid: "2478246"
    },
   {
        rowid: "2478230"
    }
  ],
  overrides: {
    Person: {
       firstName: "Charlie"
    }
  }
}
```

サンプル API 応答

次のサンプル応答は、マージアクションが保留中のターゲットレコードの行 ID を示します。

```
{
    "Person": {
        "key": {
            "rowid": "2478245"
        },
        "rowidObject": "2478245"
    }
}
```

保留中のマージ

［保留中のマージ］REST API は、変更要求の相互作用 ID に基づいて、レコードに対して行ったすべての保留中のマージタスクを更新します。［保留中のマージ］では、すべてのマージタスクの承認をワークフロープロセスが許可するまで、マージ操作を延期できます。

この API は POST メソッドを使用し、レコードの行 ID を返します。

要求 URL

［保留中のマージ］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<rowid>?
action=PendingMerge&interactionId=<interaction ID>
```

［保留中のマージ］URL に対して、次の HTTP POST 要求を行います。

```
POST http://<host>:<port>/<context>/<database ID>/<business entity>/<rowid of the record>?action=Merge
```

クエリパラメータ

保留中のマージの相互作用 ID は必須パラメータです。

サンプル API 要求

次のサンプル要求は、相互作用 ID に関連付けられているすべての保留中のマージタスクを更新します。

```
POST /Person/123?action=pendingMerge&interactionId=123
```

サンプル API 応答

次のサンプル応答には、影響を受けるルートベースオブジェクトの行 ID が含まれています。

```
{
    keys: [{rowid: "456"}, {rowid: "789"}],
    overrides: {...}
}
```

PromoteMerge

［マージの昇格］REST API は、変更要求の相互作用 ID に関連付けられているすべての保留中のマージタスクを実行します。

この API は POST メソッドを使用し、優先レコードの行 ID を返します。

要求 URL

［マージの昇格］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<rowid>?
action=PromoteMerge&interactionId=<interaction ID>
```

［マージの昇格］URL に対して、次の HTTP POST 要求を行います。

```
POST http://<host>:<port>/<context>/<database ID>/<business entity>/<rowid of the record>?action=Merge
```

クエリパラメータ

保留中のマージタスクの相互作用 ID は必須パラメータです。この API は、相互作用 ID を使用して、すべての保留中のマージタスクを検索し、マージを実行します。

サンプル API 要求

次のサンプル要求は、相互作用 ID に関連付けられているすべての保留中のマージタスクを昇格させします。

```
POST /Person/123?action=promoteMerge&interactionId=123
```

サンプル API 応答

次のサンプル応答には、保留中のマージタスクを昇格した後のレコードの行 ID が含まれています。

```
POST /Person/123?action=promoteMerge&interactionId=123
```

ファイルトランスフォーメーション

ファイルトランスフォーメーション API は、ビジネスエンティティをターゲットフィールドにマッピングする前に、インポートされたファイルからビジネスエンティティのリストを取得します。

この API は POST メソッドを使用します。

要求 URL

［ファイルトランスフォーメーション API］の形式は次のとおりです。

```
http://<host>:<port>/cmx/flat2be/{orsId}/preview{?entity,fileId,pageSize}
```

［ファイルトランスフォーメーション API］URL に対して、次の HTTP POST 要求を行います。

```
POST /cmx/flat2be/{orsId}/preview{?entity,fileId,pageSize}
```

パラメータ

ビジネスエンティティをターゲットフィールドにマッピングする前に、インポートされたファイルからビジネスエンティティのリストをプレビューするためのプロパティを指定します。

次の表に、これらのパラメータについて説明します。

パラメータ	説明
fileId	プレビューを生成するのに使用するフィールド ID。
orsId	マッピングをプレビューする元のオペレーショナルリファレンスストア ID。

パラメータ	説明
entity	オプション。プレビューするビジネスエンティティまたはリレーションの数。
pageSize	オプション。返すページ数。

サンプル API 要求

次のサンプル要求は、ビジネスエンティティをマッピングする前にインポートしたファイルからマッピング済みビジネスエンティティのリストを取得します。

```
POST /cmx/flat2be/localhost-orcl-MDM_SAMPLE/preview?entity=Person&fileId=TEMP_SVR1.1G7UW HTTP/1.1
Content-Type: application/json
Host: localhost:8080
Content-Length: 431

{
  "regionalSettings" : {
    "locale" : "en-CA",
    "datePattern" : "d MMM, yyyy",
    "decimalSeparator" : "."
    "thousandsSeparator" : ","
  },
  "objects" : [ {
    "name" : "Person",
    "fields" : [ {
      "name" : "firstName",
      "fileColumn" : "first_name",
      "skipNulls" : false
    }, {
      "name" : "lastName",
      "fileColumn" : "last_name",
      "skipNulls" : false
    } ]
  } ]
}
```

サンプル API 応答

次のサンプル応答は、ビジネスエンティティをターゲットフィールドにマッピングする前のインポートしたファイルからのマッピング済みビジネスエンティティのリストを示しています。

```
HTTP/1.1 200 OK
Server: JBoss-EAP/7
X-Powered-By: Undertow/1
Content-Type: application/json
Content-Length: 825

{
  "link" : [ ],
  "item" : [ {
    "object" : {
      "Person" : {
        "label" : "smith, alice",
        "lastName" : "smith",
        "firstName" : "alice"
      }
    }
  }, {
    "object" : {
      "Person" : {
        "label" : "smith, bob",
        "lastName" : "smith",
        "firstName" : "bob"
      }
```

```
    }
  },{
    "object" : {
      "Person" : {
        "label" : "smith, charlie",
        "lastName" : "smith",
        "firstName" : "charlie"
      }
    }
  },{
    "object" : {
      "Person" : {
        "label" : "smith, dave",
        "lastName" : "smith",
        "firstName" : "dave"
      }
    }
  },{
    "object" : {
      "Person" : {
        "label" : "smith, eve",
        "lastName" : "smith",
        "firstName" : "eve"
      }
    }
  } ]
}
```

マッピングの提案

マッピングの提案 REST API は、インポートされたファイル内のビジネスエンティティまたはリレーションのマッピングを提案します。

この API は GET メソッドを使用します。

要求 URL

［マッピングの提案 API］形式は次のとおりです。

```
http://<host>:<port>/<context>/<orsId>/<suggest{?entity,fileId,purpose}>
```

［マッピングの提案 API］URL に対して、次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/<context>/<orsId>/<suggest{?entity,fileId,purpose}>
```

要求本文

インポートされたファイルのビジネスエンティティまたはリレーションのマッピングを推奨するためのパラメータを指定します。

次の表に、要求本文内のパラメータを示します。

パラメータ	タイプ	説明
orsId	パス	マッピングをプレビューする元のオペレーショナルリファレンスストアデータベースの ID。
fileId	クエリ	オプション。インポートされるファイルのファイル ID。

パラメータ	タイプ	説明
purpose	クエリ	オプション。マッピングの目的および使用方法。
entity	クエリ	オプション。プレビューするビジネスエンティティまたはリレーションの数。エンティティパラメータを指定すると、API は指定したエンティティのマッピングのみを検出し、返します。

サンプル API 要求

次のサンプル要求は、インポートされたファイル内のビジネスエンティティまたはリレーションの推奨マッピングを返します。

```
GET /cmx/flat2be/localhost-orcl-MDM_SAMPLE/suggest?fileId=TEMP_SVR1.1G7UW HTTP/1.1
Host: localhost:8080
```

サンプル API 応答

次のサンプル応答は、インポートされたファイル内のビジネスエンティティまたはリレーションの推奨マッピングを示しています。

```
HTTP/1.1 200 OK
Server: JBoss-EAP/7
X-Powered-By: Undertow/1
Content-Type: application/json;charset=UTF-8
Content-Length: 1315

{
  "name" : "my_mapping",
  "systemName" : "Admin",
  "mapping" : {
    "confidence" : "HIGH",
    "regionalSettings" : {
      "locale" : "en-CA",
      "datePattern" : "MM-dd-yyyy",
      "decimalSeparator" : "."
      "thousandsSeparator" : ","
    },
    "objects" : [ {
      "confidence" : "HIGH",
      "name" : "PersonView",
      "fields" : [ {
        "confidence" : "HIGH",
        "name" : "firstName",
        "fileColumn" : "first_name",
        "skipNulls" : false
      }, {
        "confidence" : "HIGH",
        "name" : "jobTitle",
        "fileColumn" : "job_title",
        "skipNulls" : false
      }, {
        "confidence" : "HIGH",
        "name" : "lastName",
        "fileColumn" : "last_name",
        "skipNulls" : false
      }, {
        "confidence" : "HIGH",
        "name" : "birthdate",
        "fileColumn" : "birth_date",
        "skipNulls" : false
      }, {
        "confidence" : "HIGH",
        "name" : "genderCd",
```

```
        "fileColumn" : "gender",
        "skipNulls" : false
      } ],
      "children" : [ ]
    } ],
    "purpose" : "IMPORT"
  },
  "creator" : "admin",
  "createDate" : "2020-01-13T12:39:37.186-05:00",
  "updatedBy" : "admin",
  "lastUpdateDate" : "2020-01-13T12:39:37.186-05:00",
  "system" : false
}
```

レコードのマージ

［レコードのマージ］REST API は、2 つ以上のルートレコードをマージして単一の統合されたレコードを作成します。統合されたレコードの行 ID は、他のレコードのマージ先となるレコードの行 ID です。

この API は POST メソッドを使用します。要求本文内のマージされたレコードには、フィールドレベルのオーバーライドを指定できます。

要求 URL

レコードの［レコードのマージ］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<rowid of the record>?action=merge
```

レコードの［レコードのマージ］URL に対して、次の HTTP POST 要求を行います。

```
POST http://<host>:<port>/<context>/<database ID>/<business entity>/<rowid of the record>?action=merge
```

要求と一緒に送信するデータのメディアタイプを指定するには、Content-Type ヘッダーを追加します。

```
POST http://<host>:<port>/<context>/<database ID>/<business entity>/<rowid of the record>?action=merge
Content-Type: application/<json/xml>
```

クエリパラメータ

次の表に、URL に使用できるパラメータを示します。

パラメータ	説明
taskComment	API によってトリガされたワークフロータスクにコメントを追加します。
taskAttachments	タスク添付が有効になっている場合は、API によってトリガされたワークフロータスクにファイルを添付します。

要求本文

始める前に、［マージのプレビュー］API を使用して、選択したルートレコードのマージ結果をプレビューします。プレビューに問題がない場合、要求本文で［レコードのマージ］API に同じプロパティを使用します。

ルートレコードでフィールドの値をオーバーライドできます。例えば、一致したルートレコードに名（ファーストネーム）の正しいスペルが含まれていない場合、要求本文で正しい名を指定できます。また、一致したレコードを削除したり、他の一致レコードを指定したりすることもできます。

要求本文で、次のプロパティを使用します。

プロパティ/要素	タイプ	説明
keys	配列	必須。マージする、一致したルートレコードの順序付きリスト。レコードは行 ID によって、またはソースシステムとソースキーの組み合わせによって識別できます。
overrides	オブジェクト	ルートレコードおよび一致子レコードで、フィールドの値をオーバーライドします。
MERGE	オブジェクト	マージする子レコードのフィールド値をオーバーライドします。overrides オブジェクト内で子レコードのタイプを追加してから、MERGE オブジェクトを追加します。

次の JSON コードサンプルは、ターゲットルートレコードとマージする 2 つのルートレコードを識別します。

```
{
  keys: [
    {rowid: "2478246"},
    {rowid: "2478269"}
  ]
}
```

［レコードのマージ］API で overrides および MERGE プロパティを使用する方法の例については、［マージのプレビュー］API の要求本文を参照してください。

サンプル API 要求

次のサンプル要求は、レコードをマージし、統合されたレコードを作成します。この要求は、API によってトリガされたワークフロータスクにコメントと添付ファイルを追加します。

```
POST http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/2478245?action=merge&taskComment=Read my
comment&taskAttachments=TEMP_SVR1.29OT8,TEMP_SVR1.29OT9
Content-Type: application/<json/xml>
```

```
{
  keys: [
    {
        rowid: "2478246"
    }
  ],
  overrides: {
    Person: {
       firstName: "Charlie"
    }
  }
}
```

サンプル API 応答

次のサンプル応答では、統合されたレコードが示されています。

```
{
    "Person": {
        "link": [
            {
                "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/2478245",
                "rel": "self"
            }
        ],
        "key": {
            "rowid": "2478245"
        },
```

```
        "rowidObject": "2478245"
    }
}
```

レコードのマージ解除

［レコードのマージ解除］REST API は、ルートレコードのマージを解除し、レコードがマージされる前に存在した個々のルートレコードの状態にします。

この API は POST メソッドを使用します。

要求 URL

［レコードのマージ解除］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<rowId>?action=unmerge
```

［レコードのマージ解除］URL に対して、次の HTTP POST 要求を行います。

```
POST http://<host>:<port>/<context>/<database ID>/<business entity>/<rowId>?action=unmerge
```

要求と一緒に送信するデータのメディアタイプを指定するには、Content-Type ヘッダーを追加します。

```
POST http://<host>:<port>/<context>/<database ID>/<business entity>/<rowId>?action=unmerge
Content-Type: application/<json/xml>
```

クエリパラメータ

次の表に、URL に使用できるパラメータを示します。

パラメータ	説明
taskComment	API によってトリガされたワークフロータスクにコメントを追加します。
taskAttachments	タスク添付が有効になっている場合は、API によってトリガされたワークフロータスクにファイルを添付します。

要求本文

統合されたレコードからマージ解除するレコードのリストを要求本文で送信します。相互参照行 ID、またはソースシステムとソースキーを使用してレコードを指定します。

［レコードの読み取り］API は、マージ解除するレコードの XREF 行 ID を取得するために使用します。次のサンプル要求では、レコードの XREF メタデータを取得します。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/2638243?contentMetadata=XREF
```

サンプル API 要求

次のサンプル要求は、統合されたレコードからレコードのマージを解除します。この要求は、API によってトリガされたワークフロータスクにコメントと添付ファイルを追加します。

```
POST http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/2478248?action=unmerge&taskComment=Read my
comment&taskAttachments=TEMP_SVR1.29OT8,TEMP_SVR1.29OT9

{
        rowid: "4880369"
}
```

サンプル API 応答

次のサンプル応答では、統合されたレコードからマージを解除するレコードが示されています。

```
{
    "Person": {
        "link": [
            {
                "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/2478249",
                "rel": "self"
            }
        ],
        "key": {
            "rowid": "2478249       "
        },
        "rowidObject": "2478249       "
    }
}
```

マッピングの作成

マッピングの作成 REST API は、新しいファイルをデータベースにインポートしたときに新しいマッピングを作成します。

この API は POST メソッドを使用します。

要求 URL

［マッピングの作成］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<orsId>/<mapping>
```

［マッピングの作成］URL に対して、次の HTTP POST 要求を行います。

```
POST //<host>:<port>/<context>/<orsId>/<mapping>
```

パラメータ

要求本文に新しいマッピングを作成するためのパラメータを指定します。JSON または XML 形式を使用して要求にマッピングを作成します。

次の表に、要求本文内のパラメータを示します。

パラメータ	タイプ	説明
orsID	パス	新しいマッピングを作成するためのオペレーショナルリファレンスストアデータベース ID。
mapping	クエリ	オプション。マッピングのプロパティを入力します。
confidence	クエリ	オプション。マッピングの信頼性レベル。High、Medium、Low のうちのいずれかの値を使用します。
creator	クエリ	オプション。ユーザー名。
createDate	クエリ	オプション。マッピングを作成する日。
UpdatedBy	クエリ	オプション。既存のマッピングを更新している場合、自分のユーザー名を入力します。
lastUpdateDate	クエリ	オプション。ユーザーがマッピングを更新する日。

パラメータ	タイプ	説明
modified	クエリ	オプション。次の値を使用します: 既存のマッピングを変更している場合は true、新しいマッピングを作成している場合は false。
name	クエリ	オプション。マッピング名。
score	クエリ	オプション。マッピングの一致率。
system	クエリ	オプション。使用する値は次のとおりです。 True: システムがマッピングを作成する場合。 False: ユーザーが手動でマッピングを作成する場合。
systemName	クエリ	オプション。マッピングを作成するシステムの名前。
objects	クエリ	オプション。ビジネスエンティティまたはリレーションベースオブジェクト用のフィールドを入力します。

サンプル API 要求

次のサンプル要求は、新規マッピングを作成します。

```
POST /cmx/flat2be/localhost-orcl-MDM_SAMPLE/mapping HTTP/1.1
Content-Type: application/json
Host: localhost:8080
Content-Length: 737

{
  "mapping" : {
    "regionalSettings" : {
      "locale" : "en-CA",
      "datePattern" : "dd-MM-yyyy",
      "decimalSeparator" : "."
      "thousandsSeparator" : ","
    },
    "objects" : [ {
      "name" : "PersonView",
      "fields" : [ {
        "name" : "firstName",
        "fileColumn" : "first_name",
        "skipNulls" : false
      }, {
        "name" : "jobTitle",
        "fileColumn" : "job_title",
        "skipNulls" : false
      }, {
        "name" : "lastName",
        "fileColumn" : "last_name",
        "skipNulls" : false
      }, {
        "name" : "birthdate",
        "fileColumn" : "birth_date",
        "skipNulls" : false
      } ]
    } ],
    "purpose" : "IMPORT"
  }
}
```

サンプル API 応答

次のサンプル応答は、データベースに作成された新しいマッピングを示しています。

```
HTTP/1.1 201 Created
Server: JBoss-EAP/7
X-Powered-By: Undertow/1
Location: http://localhost:8080/cmx/flat2be/localhost-orcl-MDM_SAMPLE/mapping/SVR1.1G7UX
Content-Type: application/json;charset=UTF-8
Content-Length: 27

{
  "id" : "SVR1.1G7UX"
}
```

マッピングのプレビュー

マッピングのプレビュー REST API は、インポートされたファイルからマップされたビジネスエンティティのリストを取得します。

この API は GET メソッドを使用します。

要求 URL

［マッピングのプレビュー API］の形式は次のとおりです。

```
http://<host>:<port>/<context>/<orsId>/<preview{?entity,fileId,mapping,pageSize}>
```

［マッピングのプレビュー API］URL に対して、次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/<context>/<orsId>/<preview{?entity,fileId,mapping,pageSize}>
```

パラメータ

インポートされたファイルからのマップされたビジネスエンティティのリストをプレビューするためのパラメータを指定します。

次の表に、要求本文内のパラメータを示します。

パラメータ	タイプ	説明
orsId	パス	マッピングをプレビューする元のオペレーショナルリファレンスストア ID。
fileId	クエリ	フィールド ID。指定すると、API は指定したファイル ID からマップされたビジネスエンティティを取得します。
mappingId	クエリ	オプション。インポートされるファイルのマッピング ID を入力します。
entity	クエリ	オプション。プレビューするビジネスエンティティまたはリレーションの数を入力します。
pageSize	クエリ	オプション。返すページまたはレコードの数を入力します。

サンプル API 要求

次のサンプル要求は、インポートされたファイルからマッピング済みビジネスエンティティのプレビューを表示します。

```
GET /cmx/flat2be/localhost-orcl-MDM_SAMPLE/preview?mapping=SVR1.1G7UX&entity=PersonView&fileId=TEMP_SVR1.1G7UW
HTTP/1.1
Host: localhost:8080
```

サンプル API 応答

次のサンプル応答は、インポートされたファイルからのマッピング済みビジネスエンティティのプレビューを示しています。

```
HTTP/1.1 200 OK
Server: JBoss-EAP/7
X-Powered-By: Undertow/1
Content-Type: application/json
Content-Length: 1751

{
  "link" : [ ],
  "item" : [ {
    "object" : {
      "PersonView" : {
        "firstName" : "alice",
        "lastName" : "smith",
        "birthdate" : "1950-01-01T00:00:00-05:00",
        "genderCd" : {
          "genderCode" : "F",
          "genderDisp" : "F"
        },
        "jobTitle" : "developer"
      }
    }
  },{
    "object" : {
      "PersonView" : {
        "firstName" : "bob",
        "lastName" : "smith",
        "birthdate" : "1960-02-02T00:00:00-05:00",
        "genderCd" : {
          "genderCode" : "M",
          "genderDisp" : "M"
        },
        "jobTitle" : "tester"
      }
    }
  },{
    "object" : {
      "PersonView" : {
        "firstName" : "charlie",
        "lastName" : "smith",
        "birthdate" : "1970-03-03T00:00:00-05:00",
        "genderCd" : {
          "genderCode" : "X",
          "genderDisp" : "X"
        },
        "jobTitle" : "manager"
      }
    }
  },{
    "object" : {
      "PersonView" : {
        "firstName" : "dave",
        "lastName" : "smith",
        "genderCd" : {
          "genderCode" : "M",
          "genderDisp" : "M"
        },
```

```
        "jobTitle" : "accountant"
      }
    },
    "errors" : [ {
      "code" : "SIP-55206",
      "message" : "Failed to convert value [ten years ago] for field [birthdate].",
      "row" : 3,
      "column" : "birth_date",
      "field" : "birthdate"
    } ]
  },{
    "object" : {
      "PersonView" : {
        "firstName" : "eve",
        "lastName" : "smith",
        "birthdate" : "1990-04-04T00:00:00-04:00",
        "genderCd" : {
          "genderCode" : "F",
          "genderDisp" : "F"
        },
        "jobTitle" : ""
      }
    }
  } ]
}
```

マッピングの検出

マッピングの検出 REST API は、既存のユーザーまたはシステム定義マッピングをマッピング ID で検出します。

この API は GET メソッドを使用します。

要求 URL

［マッピングの検索］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<orsId>/<mapping{?
createdBy,fileId,firstRecord,purpose,recordsToReturn,returnTotal}>
```

［マッピングの検索］URL に対して、次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/<context>/<orsId>/<mapping {?
createdBy,fileId,firstRecord,purpose,recordsToReturn,returnTotal}>
```

パラメータ

既存のユーザーまたはシステム定義マッピングをマッピング ID で検出するためのパラメータを指定します。

次の表に、要求本文内のパラメータを示します。

パラメータ	パス	説明
orsId	パス	マッピングを検出するためのオペレーショナルリファレンスストアデータベース ID。
fileId	クエリ	フィールド ID。指定すると、API は指定したファイル ID からマッピングを取得します。
createdBy	クエリ	マッピングを作成したユーザーを入力します。

パラメータ	パス	説明
firstRecord	クエリ	オプション。レコードの最初の行のみを返す場合に入力します。
purpose	クエリ	オプション。使用する値は次のとおりです。 Import: 新しいマッピングを作成している場合。 Match: 既存のマッピングを照合している場合。
recordsToReturn	クエリ	オプション。返す行の数を入力します。
returnTotal	クエリ	オプション。レコードの合計数を入力します。

サンプル API 要求

次のサンプル要求は、既存のユーザーまたはシステム定義のマッピングをマッピング ID で検索します。

```
GET http://<host>:<port>/cmx/flat2be/{orsId}/mapping{?
createdBy,fileId,firstRecord,purpose,recordsToReturn,returnTotal}
```

サンプル API 応答

次のサンプル応答は、既存のユーザーまたはシステム定義のマッピングをマッピング ID で示しています。

```
{
    mappings: [
        {
            id: 123, name: "....", ...
        },
        ...
    ]
}
```

読み取りマッピング

読み取りマッピング REST API は、マッピング ID による既存のビジネスエンティティまたはリレーションのマッピングを返します。

この API は GET メソッドを使用します。

要求 URL

［読み取りマッピング］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<orsId>/<mapping>/<id><fileId>
```

［読み取りマッピング］URL に対して、次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/<context>/<orsId>/<mapping>/<id><fileId>
```

パラメータ

インポートされたファイル ID からのマップされたビジネスエンティティまたはリレーションを返すためのパラメータを指定します。

次の表に、要求本文内のパラメータを示します。

パラメータ	タイプ	説明
mappingId	パス	ユーザーが指定したマッピングの名前。
orsId	パス	マッピングを取得する元のオペレーショナルリファレンスストアデータベース ID。
fileId	クエリ	オプション。既存のマッピングと関連付けられているファイル ID を指定します。

サンプル API 要求

次のサンプル要求は、マッピング済みビジネスエンティティまたはリレーションをマッピング ID で返します。

```
GET /cmx/flat2be/localhost-orcl-MDM_SAMPLE/mapping/SVR1.1G7UX HTTP/1.1
Content-Type: application/json
Host: localhost:8080
```

サンプル API 応答

次のサンプル応答は、既存のビジネスエンティティまたはリレーションをマッピング ID で返します。

```
HTTP/1.1 200 OK
Server: JBoss-EAP/7
Content-Disposition: inline;filename=f.txt
X-Powered-By: Undertow/1
Content-Type: application/json;charset=UTF-8
Content-Length: 993

{
  "id" : "SVR1.1G7UX    ",
  "systemName" : "Admin",
  "mapping" : {
    "regionalSettings" : {
      "locale" : "en-CA",
      "datePattern" : "dd-MM-yyyy",
      "decimalSeparator" : "."
      "thousandsSeparator" : ","
    },
    "objects" : [ {
      "name" : "PersonView",
      "fields" : [ {
        "name" : "firstName",
        "fileColumn" : "first_name",
        "skipNulls" : false
      }, {
        "name" : "jobTitle",
        "fileColumn" : "job_title",
        "skipNulls" : false
      }, {
        "name" : "lastName",
        "fileColumn" : "last_name",
        "skipNulls" : false
      }, {
        "name" : "birthdate",
        "fileColumn" : "birth_date",
        "skipNulls" : false
      } ],
```

```
      "children" : [ ]
    } ],
    "purpose" : "IMPORT"
  },
  "creator" : "admin",
  "createDate" : "2020-01-13T13:06:26.153-05:00",
  "updatedBy" : "admin",
  "lastUpdateDate" : "2020-01-13T13:06:26.153-05:00",
  "system" : true
}
```

マッピングの更新

マッピングの更新 REST API は、既存のマッピングを新しいデータで更新します。

この API は PUT メソッドを使用します。

要求 URL

［マッピングの更新］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<orsId>/<mapping>/<mappingId>
```

［マッピングの更新］URL に対して、次の HTTP PUT 要求を行います。

```
PUT http://<host>:<port>/<context>/<orsId>/<mapping>/<mappingId>
```

パラメータ

要求本文のユーザー定義マッピングを更新するためのパラメータを指定します。JSON または XML 形式を使用して要求にマッピングを作成します。

次の表に、要求本文内のパラメータを示します。

パラメータ	タイプ	説明
mappingID	パス	ユーザーデータのマッピング ID。
orsId	パス	オペレーショナルリファレンスストアデータベース ID。
mapping	クエリ	オプション。マッピングのプロパティを入力します。
confidence	クエリ	オプション。マッピングの信頼性レベル。High、Medium、Low のうちのいずれかの値を使用します。
creator	クエリ	オプション。ユーザー名。
createDate	クエリ	オプション。マッピングを作成する日。
UpdatedBy	クエリ	オプション。既存のマッピングを更新している場合、自分のユーザー名を入力します。
lastUpdateDate	クエリ	オプション。ユーザーがマッピングを更新する日。
modified	クエリ	オプション。次の値を使用します: 既存のマッピングを変更している場合は true、新しいマッピングを作成している場合は false。
name	クエリ	オプション。マッピング名。

パラメータ	タイプ	説明
score	クエリ	オプション。マッピングの一致率。
system	クエリ	オプション。使用する値は次のとおりです。 True: システムがマッピングを作成する場合。 False: ユーザーが手動でマッピングを作成する場合。
systemName	クエリ	オプション。マッピングを作成するシステムの名前を表示します。
objects	クエリ	オプション。ビジネスエンティティまたはリレーションベースオブジェクト用のフィールドを入力します。
purpose	クエリ	オプション。使用する値は次のとおりです。 Import: 新しいマッピングを作成している場合。 Match: 既存のマッピングを照合している場合。

サンプル API 要求

次のサンプル要求は、既存のマッピングを新規データで更新します。

```
PUT /cmx/flat2be/localhost-orcl-MDM_SAMPLE/mapping/SVR1.1G7UX HTTP/1.1
Content-Type: application/json
Host: localhost:8080
Content-Length: 889

{
  "system" : false,
  "name" : "my_mapping",
  "mapping" : {
    "regionalSettings" : {
      "locale" : "en-CA",
      "datePattern" : "dd-MM-yyyy",
      "decimalSeparator" : "."
      "thousandsSeparator" : ","
    },
    "objects" : [ {
      "name" : "PersonView",
      "fields" : [ {
        "name" : "firstName",
        "fileColumn" : "first_name",
        "skipNulls" : false
      }, {
        "name" : "jobTitle",
        "fileColumn" : "job_title",
        "skipNulls" : false
      }, {
        "name" : "lastName",
        "fileColumn" : "last_name",
        "skipNulls" : false
      }, {
        "name" : "birthdate",
        "fileColumn" : "birth_date",
        "skipNulls" : false
      }, {
        "name" : "genderCd",
        "fileColumn" : "gender",
        "skipNulls" : false
      } ]
    } ],
    "purpose" : "IMPORT"
  }
}
```

サンプル API 応答

次は、既存のマッピングを新しいデータで正常に更新した後のサンプル応答です。

```
HTTP/1.1 204 No Content
Server: JBoss-EAP/7
X-Powered-By: Undertow/1
```

ファイルトランスフォーメーション

ファイルトランスフォーメーション API は、ビジネスエンティティをターゲットフィールドにマッピングする前に、インポートされたファイルからビジネスエンティティのリストを取得します。

この API は POST メソッドを使用します。

要求 URL

［ファイルトランスフォーメーション API］の形式は次のとおりです。

```
http://<host>:<port>/cmx/flat2be/{orsId}/preview{?entity,fileId,pageSize}
```

［ファイルトランスフォーメーション API］URL に対して、次の HTTP POST 要求を行います。

```
POST /cmx/flat2be/{orsId}/preview{?entity,fileId,pageSize}
```

パラメータ

ビジネスエンティティをターゲットフィールドにマッピングする前に、インポートされたファイルからビジネスエンティティのリストをプレビューするためのプロパティを指定します。

次の表に、これらのパラメータについて説明します。

パラメータ	説明
fileId	プレビューを生成するのに使用するフィールド ID。
orsId	マッピングをプレビューする元のオペレーショナルリファレンスストア ID。
entity	オプション。プレビューするビジネスエンティティまたはリレーションの数。
pageSize	オプション。返すページ数。

サンプル API 要求

次のサンプル要求は、ビジネスエンティティをマッピングする前にインポートしたファイルからマッピング済みビジネスエンティティのリストを取得します。

```
POST /cmx/flat2be/localhost-orcl-MDM_SAMPLE/preview?entity=Person&fileId=TEMP_SVR1.1G7UW HTTP/1.1
Content-Type: application/json
Host: localhost:8080
Content-Length: 431

{
  "regionalSettings" : {
    "locale" : "en-CA",
    "datePattern" : "d MMM, yyyy",
    "decimalSeparator" : "."
    "thousandsSeparator" : ","
  },
  "objects" : [ {
    "name" : "Person",
    "fields" : [ {
```

```
      "name" : "firstName",
      "fileColumn" : "first_name",
      "skipNulls" : false
    }, {
      "name" : "lastName",
      "fileColumn" : "last_name",
      "skipNulls" : false
    } ]
  } ]
}
```

サンプル API 応答

次のサンプル応答は、ビジネスエンティティをターゲットフィールドにマッピングする前のインポートしたファイルからのマッピング済みビジネスエンティティのリストを示しています。

```
HTTP/1.1 200 OK
Server: JBoss-EAP/7
X-Powered-By: Undertow/1
Content-Type: application/json
Content-Length: 825

{
  "link" : [ ],
  "item" : [ {
    "object" : {
      "Person" : {
        "label" : "smith, alice",
        "lastName" : "smith",
        "firstName" : "alice"
      }
    }
  }, {
    "object" : {
      "Person" : {
        "label" : "smith, bob",
        "lastName" : "smith",
        "firstName" : "bob"
      }
    }
  }, {
    "object" : {
      "Person" : {
        "label" : "smith, charlie",
        "lastName" : "smith",
        "firstName" : "charlie"
      }
    }
  }, {
    "object" : {
      "Person" : {
        "label" : "smith, dave",
        "lastName" : "smith",
        "firstName" : "dave"
      }
    }
  }, {
    "object" : {
      "Person" : {
        "label" : "smith, eve",
        "lastName" : "smith",
        "firstName" : "eve"
      }
    }
  } ]
}
```

リレーションの読み取り

［リレーションの読み取り］REST API は、パーティタイプ、行 ID、2 つのレコードの表示名など、リレーションレコードの詳細を返します。

この API は GET メソッドを使用します。

要求 URL

［リレーションの読み取り］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/<relationship>/<row ID of the relationship record>
```

この URL に対して次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/<context>/<database ID>/<relationship>/<row ID of the relationship record>
```

クエリパラメータ

次の表にクエリパラメータを示します。

パラメータ	説明
suppressLinks	オプション。API 応答に親と子のリンクを含めるかどうかを示します。応答に親と子のリンクをいっさい含めない場合は、このパラメータを true に設定します。API 応答にリンクを表示するには、このパラメータを false に設定します。デフォルトは false です。
depth	オプション。返す子レベルの数。

サンプル API 要求

次のサンプル要求は、リレーションタイプ ProductGroupProductGroupIsParentOfProductProducts である、行 ID 85 のリレーションレコードの詳細を返します。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/ProductGroupProductGroupIsParentOfProductProducts/85
```

次のサンプル要求は、depth が 2 の詳細を返します。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/ProductGroupProductGroupIsParentOfProductProducts/85?depth=2
```

サンプル API 応答

次の例は、リレーションタイプ ProductGroupProductGroupIsParentOfProductProducts である、行 ID 85 のリレーションレコードの詳細を示しています。

```
{
  "link": [
    {
      "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/
ProductGroupProductGroupIsParentOfProductProducts/85.json",
      "rel": "self"
    },
    {
      "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/
ProductGroupProductGroupIsParentOfProductProducts/85.json?depth=2",
      "rel": "children"
    }
  ],
  "rowidObject": "85            ",
  "label": "ProductGroup Product Group is parent of Product Products",
```

```
  "rowidRelType": "9              ",
  "rowidHierarchy": "3              ",
  "from": {
    "link": [
      {
        "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/
ProductGroupProductGroupIsParentOfProductProducts/85.json",
        "rel": "parent"
      },
      {
        "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/
ProductGroupProductGroupIsParentOfProductProducts/85/from/86.json",
        "rel": "self"
      },
      {
        "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/
ProductGroupProductGroupIsParentOfProductProducts/85/from/86.json?depth=2",
        "rel": "children"
      }
    ],
    "rowidObject": "86             ",
    "label": "ProductGroup",
    "productType": "Product Group",
    "productNumber": "Presenter2",
    "productName": "Presenter",
    "productDesc": "Presenter Family",
    "productTypeCd": {
      "link": [
        {
          "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/
ProductGroupProductGroupIsParentOfProductProducts/85/from/86.json",
          "rel": "parent"
        },
        {
          "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/
ProductGroupProductGroupIsParentOfProductProducts/85/from/86/productTypeCd.json?depth=2",
          "rel": "children"
        },
        {
          "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/
ProductGroupProductGroupIsParentOfProductProducts/85/from/86/productTypeCd.json",
          "rel": "self"
        }
      ],
      "productType": "Product Group"
    }
  },
  "to": {
    "link": [
      {
        "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/
ProductGroupProductGroupIsParentOfProductProducts/85/to/66.json",
        "rel": "self"
      },
      {
        "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/
ProductGroupProductGroupIsParentOfProductProducts/85/to/66.json?depth=2",
        "rel": "children"
      },
      {
        "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/
ProductGroupProductGroupIsParentOfProductProducts/85.json",
        "rel": "parent"
      }
    ],
    "rowidObject": "66             ",
    "label": "Products",
    "productType": "Product",
    "productNumber": "931307-0403",
    "productName": "2.4 GHz Cordless Presenter",
    "productDesc": "A cordless presenter to streamline your delivery.",
```

```
    "productTypeCd": {
      "link": [
        {
          "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/
ProductGroupProductGroupIsParentOfProductProducts/85/to/66/productTypeCd.json",
          "rel": "self"
        },
        {
          "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/
ProductGroupProductGroupIsParentOfProductProducts/85/to/66.json",
          "rel": "parent"
        },
        {
          "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/
ProductGroupProductGroupIsParentOfProductProducts/85/to/66/productTypeCd.json?depth=2",
          "rel": "children"
        }
      ],
      "productType": "Presenter"
    }
  }
}
```

リレーションの作成

［リレーションの作成］REST API では、指定したレコード間のリレーションを作成します。レコード間のリレーションを作成するには、レコードが属するビジネスエンティティの間にリレーションが存在する必要があります。例えば、Informatica と John Smith との間のリレーションを指定する場合、Organization ビジネスエンティティと Person ビジネスエンティティの間にリレーションが存在する必要があります。リレーションデータは要求本文で送信する必要があります。

この API は、PUT および POST メソッドを使用します。

要求 URL

［リレーションの作成］REST URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/<relationship>?systemName=<name of the source system>
```

注: ソースシステムの名前は URL の必須パラメータです。

URL に対して、次の HTTP POST または PUT 要求を行います。

```
POST http://<host>:<port>/<context>/<database ID>/<relationship>?systemName=<name of the source system>
```

要求と一緒に送信するデータのメディアタイプを指定するには、Content-Type ヘッダーを追加します。

```
Content-Type: application/<json/xml>
```

URL パラメータ

ソースシステムの名前は要求 URL の必須パラメータです。

要求本文

リレーションレコードのデータを REST 要求本文で送信します。データを送信するには、JSON 形式または XML 形式を使用します。必須パラメータ値を要求本文で指定します。

サンプル API 要求

次のサンプル要求は、行 ID 101 の Organization ビジネスエンティティと行 ID 1101 の Person ビジネスエンティティの間の OrganizationEmploysPerson リレーションを作成します。

```
POST http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/OrganizationEmploysPerson?systemName=SFA
Content-Type: application/json

{
  "from": {
    "rowidObject": "101"
    },

  "to": {
    "rowidObject": "1101          "
    },

    "relName": "Documentation",
    "relDesc": "Writer"
}
```

OrganizationEmploysPerson リレーションは、Organization ビジネスエンティティから Person ビジネスエンティティへのリレーションを定義します。from 要素はリレーションが始まるレコードを指定し、to 要素はリレーションが終わるレコードを指定します。

サンプル API 応答

次のサンプル応答は、行 ID 101 の Organization ビジネスエンティティと行 ID 1101 の Person ビジネスエンティティの間のリレーションを作成した後の、応答ヘッダーと応答本文を示しています:

```
{
        "OrganizationEmployesPerson": {
        "key": {
        "rowid": "414721"
        "sourceKey": "SVR1.1E7UW"
 }-

"rowidObject": "414721"

    "from": {
                    "key": {
                                "rowid": "101 "
                                                }-
        "rowidObject": "101 "
        }-

 "to": {
                "key": {
                            "rowid": "1101 "
        }-

"rowidObject": "1101 "
                }-

    }-

}
```

リレーションの更新

［リレーションの更新］REST API は、2 つのレコード間のリレーションを更新します。この API は、リレーションに定義された追加の属性を更新します。

この API は、POST および PUT メソッドを使用します。

要求 URL

［リレーションの更新］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/<relationship>/<row ID>?systemName=<name of the source system>
```

注: ソースシステムの名前は URL の必須パラメータです。

URL に対して、次の HTTP POST または PUT 呼び出しを行います。

```
http://<host>:<port>/<context>/<database ID>/<relationship>/<row ID>?systemName=<name of the source system>
```

要求と一緒に送信するデータのメディアタイプを指定するには、Content-Type ヘッダーを追加します。

要求本文

リレーションレコードの更新を要求本文で送信します。データを送信するには、JSON 形式または XML 形式を使用します。必須パラメータ値を要求本文で指定します。

サンプル API 要求

行 ID 414721 のリレーションは、行 ID 101 の Organization エンティティと行 ID 1101 の Person エンティティの間の OrganizationEmploysPerson リレーションです。

次のサンプル要求は、行 ID 414721 のリレーションレコードを更新します:

```
POST http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/OrganizationEmploysPerson/414721?systemName=SFA
Content-Type: application/json
{
  "from": {
    "rowidObject": "101"
    },

  "to": {
    "rowidObject": "1101          "
    },

    "relName": "Development",
    "relDesc": "Software Engineer",

                "$original":
                {"relName": "Documentation",
                    "relDesc": "Writer"}


}
```

サンプル API 応答

行 ID 414721 のリレーションを更新後、次のサンプル応答を受信します。

```
{
        "OrganizationEmployesPerson": {
                "key": {
                                "rowid": "414721"
                                "sourceKey": "SVR1.1E7UW"
                                        }-
            "rowidObject": "414721"
        "from": {
            "key": {
                            "rowid": "101"
                                    }-
                "rowidObject": "101"
                }-
        "to": {
                "key": {
```

```
                        "rowid": "1101 "
                            }-
        "rowidObject": "1101 "
                        }-
            }-
}
```

リレーションの削除

［リレーションの削除］REST API は、2 つのレコード間のリレーションを削除します。

この API は DELETE メソッドを使用します。

要求 URL

［リレーションの削除］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/<relationship>/<rowID of the relationship record>?
systemName=<name of the source system>
```

注: ソースシステムの名前は URL の必須パラメータです。

［削除］URL に対して、次の HTTP DELETE 要求を行います。

```
DELETE http://<host>:<port>/<context>/<database ID>/<relationship>/<rowID of the relationship record>?
systemName=<name of the source system>
```

クエリパラメータ

ソースシステムの名前は必須 URL パラメータです。ソースシステムを指定するには、systemName パラメータを使用します。

サンプル API 要求

次のサンプル要求は、行 ID 414721 のリレーションレコードを削除します。

```
DELETE http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/OrganizationEmploysPerson/414721?systemName=SFA
```

サンプル API 応答

次のサンプル応答は、行 ID 414721 のリレーションレコードを正常に削除した後の応答を示しています:

```
{
"OrganizationEmployesPerson": {
        "key": {
                "rowid": "414721"
               }-
"rowidObject": "414721"
            }-
}
```

関連するレコードの取得

［関連するレコードの取得］REST API は、設定されているリレーションに基づいて、指定されたルートレコードに関連するレコードのリストを返します。この API は、リレーションの詳細も返します。

この API は GET メソッドを使用します。

要求 URL

［関連するレコードの取得］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<rowId>?action=related
```

［関連するレコードの取得］URL に対して、次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/<context>/<database ID>/<business entity>/<rowId>?action=related
```

クエリパラメータ

クエリパラメータを要求 URL に付加できます。

次の表に、使用できるクエリパラメータをリストします。

パラメータ	説明
recordsToReturn	読み取る many の子のレコード数。
searchToken	結果セットの後続ページを取得する検索トークン。
returnTotal	結果セット内のレコード数を返します。結果セット内のレコード数を取得するには、true に設定します。デフォルトは false です。

フィルタパラメータ

URL にパラメータを付加して、関連するレコードをフィルタリングできます。

次の表に、使用できるフィルタパラメータを示します。

パラメータ	説明
recordStates	取得するレコードの状態のカンマ区切りリスト。サポートされるレコードの状態は、ACTIVE、PENDING、および DELETED です。デフォルトは ACTIVE です。 例えば、`/Party/123?action=related&recordStates=ACTIVE,PENDING` というクエリは、アクティブまたは保留中のレコードを返します。
entityLabel	エンティティのラベル。
relationshipLabel	リレーションのラベル。
entityType	エンティティタイプのカンマ区切りリスト。例えば、`entityType=Person,Organization` リストは Person および Organization エンティティタイプに関連するレコードを返します。
relationshipType	リレーションタイプのカンマ区切りリスト。例えば、`relationshipType=Employee,Employer` リストは Employee および Employer リレーションタイプに関連するレコードを返します。

注: 複数のフィルタ条件を指定すると、結果には AND 条件を満たすすべてのレコードが含まれます。

応答本文

応答本文には、関連するレコードのリスト、関連するレコードおよびリレーションの詳細、および検索トークンが含まれます。検索トークンは、後続の結果ページの取得に使用します。

サンプル API 要求

次のサンプル要求は、行 ID 101 の Organization ビジネスエンティティに構成された関連レコードとリレーションを取得します。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Organization/101?action=related
```

サンプル API 応答

次の例は、行 ID 101 の Organization ビジネスエンティティの関連レコードとリレーションを示しています。


```
{
  "link": [],
  "firstRecord": 1,
  "pageSize": 10,
  "searchToken": "SVR1.1H7YB",
  "relatedEntity": [
    {
      "businessEntity": {
        "SecurePerson": {
          "link": [
            {
              "href": "http://10.21.43.42:8080/cmx/cs/localhost-orcl-ds_ui1/SecurePerson/1101",
              "rel": "self"
            }
          ],
          "rowidObject": "1101           ",
          "creator": "admin",
          "createDate": "2008-11-11T21:22:20-08:00",
          "updatedBy": "admin",
          "lastUpdateDate": "2012-03-29T19:03:19-07:00",
          "consolidationInd": "1",
          "lastRowidSystem": "SYS0           ",
          "dirtyIndicator": "0",
          "interactionId": "20003000",
          "hubStateInd": "1",
          "partyType": "Person",
          "lastName": "Obama",
          "firstName": "Barack"
        }
      },
      "entityLabel": "Obama,Barack",
      "relationshipLabel": "Organization employes SecurePerson",
      "relationship": {
        "rowidObject": "414721         ",
        "creator": "admin",
        "createDate": "2016-10-17T01:58:12.436-07:00",
        "updatedBy": "admin",
        "lastUpdateDate": "2016-10-19T01:40:28.830-07:00",
        "consolidationInd": "4",
        "lastRowidSystem": "SFA            ",
        "interactionId": "1476866426786",
        "hubStateInd": "1",
        "rowidRelType": "101            ",
        "rowidHierarchy": "1              ",
        "relName": "Documentation",
        "relDesc": "Writer"
      },
      "entityType": "SecurePerson",
      "relationshipType": "OrganizationEmployesSecurePerson"
    },
    {
      "businessEntity": {
        "SecurePerson": {
          "link": [
            {
              "href": "http://10.21.43.42:8080/cmx/cs/localhost-orcl-ds_ui1/SecurePerson/114",
              "rel": "self"
            }
          ],
```

```
      "rowidObject": "114           ",
      "creator": "admin",
      "createDate": "2008-08-11T23:00:55-07:00",
      "updatedBy": "Admin",
      "lastUpdateDate": "2008-08-12T02:59:17-07:00",
      "consolidationInd": "1",
      "lastRowidSystem": "Legacy         ",
      "dirtyIndicator": "0",
      "hubStateInd": "1",
      "partyType": "Person",
      "lastName": "HERNANDEZ",
      "displayName": "ALEJANDRO  HERNANDEZ                                    ",
      "firstName": "ALEJANDRO"
    }
  },
  "entityLabel": "HERNANDEZ,ALEJANDRO",
  "relationshipLabel": "Organization employes SecurePerson",
  "relationship": {
    "rowidObject": "434721        ",
    "creator": "admin",
    "createDate": "2016-10-19T01:49:03.415-07:00",
    "updatedBy": "admin",
    "lastUpdateDate": "2016-10-19T01:49:03.415-07:00",
    "consolidationInd": "4",
    "lastRowidSystem": "SFA            ",
    "hubStateInd": "1",
    "rowidRelType": "101           ",
    "rowidHierarchy": "1             ",
    "relName": "Documentation",
    "relDesc": "Writer"
  },
  "entityType": "SecurePerson",
  "relationshipType": "OrganizationEmployesSecurePerson"
},
{
  "businessEntity": {
    "Person": {
      "link": [
        {
          "href": "http://10.21.43.42:8080/cmx/cs/localhost-orcl-ds_ui1/Person/1101",
          "rel": "self"
        }
      ],
      "rowidObject": "1101          ",
      "creator": "admin",
      "createDate": "2008-11-11T21:22:20-08:00",
      "updatedBy": "admin",
      "lastUpdateDate": "2012-03-29T19:03:19-07:00",
      "consolidationInd": "1",
      "lastRowidSystem": "SYS0          ",
      "dirtyIndicator": "0",
      "interactionId": "20003000",
      "hubStateInd": "1",
      "partyType": "Person",
      "lastName": "Obama",
      "firstName": "Barack"
    }
  },
  "entityLabel": "Obama,Barack",
  "relationshipLabel": "Organization employes Person",
  "relationship": {
    "rowidObject": "414721        ",
    "creator": "admin",
    "createDate": "2016-10-17T01:58:12.436-07:00",
    "updatedBy": "admin",
    "lastUpdateDate": "2016-10-19T01:40:28.830-07:00",
    "consolidationInd": "4",
    "lastRowidSystem": "SFA           ",
    "interactionId": "1476866426786",
    "hubStateInd": "1",
    "rowidRelType": "101           ",
```

```
          "rowidHierarchy": "1             ",
          "relName": "Documentation",
          "relDesc": "Writer"
        },
        "entityType": "Person",
        "relationshipType": "OrganizationEmployesPerson"
      },
      {
        "businessEntity": {
          "Person": {
            "link": [
              {
                "href": "http://10.21.43.42:8080/cmx/cs/localhost-orcl-ds_ui1/Person/114",
                "rel": "self"
              }
            ],
            "rowidObject": "114           ",
            "creator": "admin",
            "createDate": "2008-08-11T23:00:55-07:00",
            "updatedBy": "Admin",
            "lastUpdateDate": "2008-08-12T02:59:17-07:00",
            "consolidationInd": "1",
            "lastRowidSystem": "Legacy        ",
            "dirtyIndicator": "0",
            "hubStateInd": "1",
            "partyType": "Person",
            "lastName": "HERNANDEZ",
            "displayName": "ALEJANDRO  HERNANDEZ                                ",
            "statusCd": "A ",
            "firstName": "ALEJANDRO"
          }
        },
        "entityLabel": "HERNANDEZ,ALEJANDRO",
        "relationshipLabel": "Organization employes Person",
        "relationship": {
          "rowidObject": "434721        ",
          "creator": "admin",
          "createDate": "2016-10-19T01:49:03.415-07:00",
          "updatedBy": "admin",
          "lastUpdateDate": "2016-10-19T01:49:03.415-07:00",
          "consolidationInd": "4",
          "lastRowidSystem": "SFA           ",
          "hubStateInd": "1",
          "rowidRelType": "101           ",
          "rowidHierarchy": "1             ",
          "relName": "Documentation",
          "relDesc": "Writer"
        },
        "entityType": "Person",
        "relationshipType": "OrganizationEmployesPerson"
      }
    ]
}
```


関連ビジネスエンティティのエクスポート

概念の簡単な説明をここに入力します（必須）。

これが概念のスタートです。

要求 URL

概念の簡単な説明をここに入力します（必須）。

これが概念のスタートです。

サンプル API 要求

概念の簡単な説明をここに入力します（必須）。

これが概念のスタートです。

サンプル API 応答

概念の簡単な説明をここに入力します（必須）。

これが概念のスタートです。

リスト階層

階層のリスト REST API はすべての階層を返します。この API を使用して、任意のレベルの特定のビジネスエンティティを含む階層を返すこともできます。

この API は GET メソッドを使用します。

要求 URL

［階層の一覧表示］URL は、次の形式にすることができます。

すべての階層を返す URL

次の URL を使用して、すべての階層を一覧表示します。

`http://<host>:<port>/<context>/<database ID>/metadata/hierarchy`

指定されたビジネスエンティティを含むすべての階層を返す URL

次の URL を使用して、任意のレベルで指定されたビジネスエンティティを含むすべての階層を一覧表示します。

`http://<host>:<port>/<context>/<database ID>/metadata/hierarchy?entityName=<entity name>`

サンプル API 要求

次のサンプル要求は、すべての階層を一覧表示します。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/metadata/hierarchy
```

次のサンプル要求は、Product ビジネスエンティティを含むすべての階層を一覧表示します。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/metadata/hierarchy?entityName=Product
```

サンプル API 応答

次のサンプル応答は、Product ビジネスエンティティを含む階層を示しています。

```
{
  "link":[],
  "item":[
    {
      "operations":{
        "read":{
          "allowed":true
        },
        "update":{
          "allowed":true
        }
      },
      "root":"ProductGroup",
      "name":"Product"
    }
```

```
  ]
}
```

階層メタデータの取得

階層メタデータの取得 REST API は、階層のメタデータを返します。

この API は GET メソッドを使用します。

要求 URL

［階層メタデータの取得］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/metadata/hierarchy/<hierarchy name>
```

［階層メタデータの取得］URL に対して次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/<context>/<database ID>/metadata/hierarchy/<hierarchy name>
```

サンプル API 要求

次のサンプル要求は、階層のメタデータを返します。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/metadata/hierarchy/ODI
```

サンプル API 応答

次のサンプル応答は、階層のメタデータを示しています。

```
{
  "relationship":[
    {
      "field":[
       {
         "allowedValues":[
           "2"
         ],
         "name":"rowidRelType",
         "label":"Rowid Rel Type",
         "dataType":"String",
         "length":14,
         "readOnly":false,
         "required":false,
         "system":false,
         "trust":false,
         "applyNullValues":false,
         "displayFormat":"DATETIME_LONG_FORMAT",
         "filterable":true,
         "sortable":true,
         "lookup":{
           "link":[
             {
               "href":"http://localhost:8080/cmx/lookup/localhost-orcl-MDM_SAMPLE/id-label/IsOdiParentOf/
rowidRelType",
               "rel":"lookup"
             },
             {
               "href":"http://localhost:8080/cmx/lookup/localhost-orcl-MDM_SAMPLE/object/IsOdiParentOf/
rowidRelType",
               "rel":"list"
             }
           ],
           "object":"RboRelType",
           "key":"rowidObject",
           "value":"displayName"
         }
```

```
        },
        {
          "allowedValues":[
            "1"
          ],
          "name":"rowidHierarchy",
          "label":"Rowid Hierarchy",
          "dataType":"String",
          "length":14,
          "readOnly":false,
          "required":false,
          "system":false,
          "trust":false,
          "applyNullValues":false,
          "displayFormat":"DATETIME_LONG_FORMAT",
          "filterable":true,
          "sortable":true,
          "lookup":{
            "link":[
              {
                "href":"http://localhost:8080/cmx/lookup/localhost-orcl-MDM_SAMPLE/id-label/IsOdiParentOf/
rowidHierarchy",
                "rel":"lookup"
              },
              {
                "href":"http://localhost:8080/cmx/lookup/localhost-orcl-MDM_SAMPLE/object/IsOdiParentOf/
rowidHierarchy",
                "rel":"list"
              }
            ],
            "object":"RboHierarchy",
            "key":"rowidObject",
            "value":"displayName"
          }
        },
        {
          "name":"relName",
          "label":"Rel Name",
          "dataType":"String",
          "length":50,
          "trust":false,
          "applyNullValues":false,
          "displayFormat":"DATETIME_LONG_FORMAT",
          "filterable":true,
          "sortable":true
        },
        {
          "name":"relDesc",
          "label":"Rel Desc",
          "dataType":"String",
          "length":200,
          "trust":false,
          "applyNullValues":false,
          "displayFormat":"DATETIME_LONG_FORMAT",
          "filterable":true,
          "sortable":true
        },
        {
          "name":"consolidationInd",
          "label":"Consolidation Ind",
          "dataType":"Integer",
          "length":38,
          "readOnly":true,
          "system":true,
          "trust":false,
          "applyNullValues":false,
          "filterable":true,
          "sortable":true
        },
        {
          "name":"creator",
```

```
    "label":"Creator",
    "dataType":"String",
    "length":50,
    "readOnly":true,
    "system":true,
    "trust":false,
    "applyNullValues":false,
    "filterable":true,
    "sortable":true
  },
  {
    "name":"interactionId",
    "label":"Interaction Id",
    "dataType":"Integer",
    "length":38,
    "readOnly":true,
    "system":true,
    "trust":false,
    "applyNullValues":false,
    "filterable":true,
    "sortable":true
  },
  {
    "name":"updatedBy",
    "label":"Updated By",
    "dataType":"String",
    "length":50,
    "readOnly":true,
    "system":true,
    "trust":false,
    "applyNullValues":false,
    "filterable":true,
    "sortable":true
  },
  {
    "name":"lastUpdateDate",
    "label":"Last Update Date",
    "dataType":"Date",
    "readOnly":true,
    "system":true,
    "trust":false,
    "applyNullValues":false,
    "filterable":true,
    "sortable":true
  },
  {
    "name":"lastRowidSystem",
    "label":"Last Rowid System",
    "dataType":"String",
    "length":14,
    "readOnly":true,
    "system":true,
    "trust":false,
    "applyNullValues":false,
    "filterable":true,
    "sortable":true
  },
  {
    "name":"dirtyIndicator",
    "label":"Dirty Indicator",
    "dataType":"Integer",
    "length":38,
    "readOnly":true,
    "system":true,
    "trust":false,
    "applyNullValues":false,
    "filterable":true,
    "sortable":true
  },
  {
    "name":"deletedBy",
```

```
    "label":"Deleted By",
    "dataType":"String",
    "length":50,
    "readOnly":true,
    "system":true,
    "trust":false,
    "applyNullValues":false,
    "filterable":true,
    "sortable":true
  },
  {
    "name":"deletedInd",
    "label":"Deleted Indicator",
    "dataType":"Integer",
    "length":38,
    "readOnly":true,
    "system":true,
    "trust":false,
    "applyNullValues":false,
    "filterable":true,
    "sortable":true
  },
  {
    "name":"hubStateInd",
    "label":"Hub State Ind",
    "dataType":"Integer",
    "length":38,
    "readOnly":true,
    "system":true,
    "trust":false,
    "applyNullValues":false,
    "filterable":true,
    "sortable":true
  },
  {
    "name":"deletedDate",
    "label":"Deleted Date",
    "dataType":"Date",
    "readOnly":true,
    "system":true,
    "trust":false,
    "applyNullValues":false,
    "filterable":true,
    "sortable":true
  },
  {
    "name":"rowidObject",
    "label":"Rowid Object",
    "dataType":"String",
    "length":14,
    "readOnly":true,
    "system":true,
    "trust":false,
    "applyNullValues":false,
    "filterable":true,
    "sortable":true
  },
  {
    "name":"cmDirtyInd",
    "label":"Content metadata dirty Ind",
    "dataType":"Integer",
    "length":38,
    "readOnly":true,
    "system":true,
    "trust":false,
    "applyNullValues":false,
    "filterable":true,
    "sortable":true
  },
  {
    "name":"createDate",
```

```
            "label":"Create Date",
            "dataType":"Date",
            "readOnly":true,
            "system":true,
            "trust":false,
            "applyNullValues":false,
            "filterable":true,
            "sortable":true
          }
        ],
        "contentMetadata":[
          {
            "operations":{
              "read":{
                "allowed":true
              },
              "create":{
                "allowed":true
              },
              "update":{
                "allowed":true
              },
              "delete":{
                "allowed":true
              }
            },
            "name":"XREF"
          }
        ],
        "name":"IsOdiParentOf",
        "label":"is ODI parent of",
        "color":"#990066",
        "direction":"ENTITY_1_TO_ENTITY_2",
        "bidirectional":false,
        "from":{
          "dataType":"BusinessEntity",
          "required":true,
          "lookup":{
            "object":"Organization"
          }
        },
        "to":{
          "dataType":"BusinessEntity",
          "required":true,
          "lookup":{
            "object":"Organization"
          }
        },
        "hierarchy":"ODI"
      }
    ],
    "operations":{
      "read":{
        "allowed":true
      },
      "update":{
        "allowed":true
      }
    },
    "root":"Organization",
    "name":"ODI"
}
```


階層パスの取得

階層パスの取得 REST API は、ビジネスエンティティレコードから階層ルートへのパスを返します。

この API は GET メソッドを使用します。

要求 URL

［階層パスの取得］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/hierarchy/<hierarchy name>/entity/<entity name>/<entity ID>/path
```

［階層パスの取得］URL に対して次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/<context>/<database ID>/hierarchy/<hierarchy name>/entity/<entity name>/<entity ID>/
path
```

クエリパラメータ

クエリパラメータを要求 URL に追加して、リレーションをフィルタリングできます。

以下の表に、クエリパラメータを示します。

パラメータ	説明
effectiveDate	リレーションの取得対象となる日付。
interactionId	相互作用の ID。
readLabelsOnly	結果でラベルのみを返すかどうかを示します。値は true または false です。
readSystemFields	結果でシステムフィールドを返すかどうかを示します。値は true または false です。
recordsToReturn	返す行の数を指定します。
rejectInteractionId	変更を却下するための相互作用の ID。
relationshipType	リレーションタイプのカンマ区切りリスト。例えば、relationshipType=Employee,Employer リストは Employee および Employer リレーションタイプに関連するレコードを返します。
returnTotal	結果セット内のレコード数を返します。結果セット内のレコード数を取得するには、true に設定します。
rootId	階層のルートの ID。

サンプル API 要求

次のサンプル要求は、ビジネスエンティティレコードから階層ルートへのパスを一覧表示します。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/hierarchy/ODI/entity/Organization/35/path
```

サンプル API 応答

次のサンプル応答は、ビジネスエンティティレコードから階層ルートへのパスを示しています。

```
{
  "link":[],
  "item":[
    {
      "businessEntity":{
        "Organization":{
          "link":[
            {
              "href":"http://localhost:8080/cmx/request/hm_icons/hierarchymanager/Hospital/Hospital.png?
ors=localhost-orcl-MDM_SAMPLE",
              "rel":"icon"
            },
```

```
            {
              "href":"http://localhost:8080/cmx/cs/localhost-orcl-MDM_SAMPLE/Organization/584",
              "rel":"self"
            }
          ],
          "rowidObject":"584",
          "label":"Time Warner Inc",
          "partyType":"Organization",
          "displayName":"Time Warner Inc",
          "dunsNumber":"799527630"
        }
      },
      "entityLabel":"Time Warner Inc",
      "entityType":"Organization",
      "depth":2,
      "object":{
        "businessEntity":{
          "Organization":{
            "link":[
              {
                "href":"http://localhost:8080/cmx/request/hm_icons/hierarchymanager/Hospital/Hospital.png?
ors=localhost-orcl-MDM_SAMPLE",
                "rel":"icon"
              },
              {
                "href":"http://localhost:8080/cmx/cs/localhost-orcl-MDM_SAMPLE/Organization/55",
                "rel":"self"
              }
            ],
            "rowidObject":"55",
            "label":"Historic TW Inc",
            "partyType":"Organization",
            "displayName":"Historic TW Inc",
            "dunsNumber":"958466278"
          }
        },
        "entityLabel":"Historic TW Inc",
        "relationshipLabel":"is ODI parent of",
        "relationship":{
          "rowidObject":"834",
          "label":"is ODI parent of",
          "rowidRelType":"2",
          "rowidHierarchy":"1",
          "relName":"Parent",
          "relDesc":"Parent",
          "from":{
            "rowidObject":"584"
          },
          "to":{
            "rowidObject":"55"
          }
        },
        "entityType":"Organization",
        "relationshipType":"IsOdiParentOf",
        "object":{
          "businessEntity":{
            "Organization":{
              "link":[
                {
                  "href":"http://localhost:8080/cmx/request/hm_icons/hierarchymanager/Hospital/Hospital.png?
ors=localhost-orcl-MDM_SAMPLE",
                  "rel":"icon"
                },
                {
                  "href":"http://localhost:8080/cmx/cs/localhost-orcl-MDM_SAMPLE/Organization/35",
                  "rel":"self"
                }
              ],
              "rowidObject":"35",
              "effectivePeriod":{
```

```
            },
            "label":"People Magazine",
            "partyType":"Organization",
            "displayName":"People Magazine",
            "dunsNumber":"011584096"
          }
        },
        "entityLabel":"People Magazine",
        "relationshipLabel":"is ODI parent of",
        "relationship":{
          "rowidObject":"889",
          "label":"is ODI parent of",
          "rowidRelType":"2",
          "rowidHierarchy":"1",
          "relName":"Parent",
          "relDesc":"Parent",
          "from":{
            "rowidObject":"55"
          },
          "to":{
            "rowidObject":"35"
          }
        },
        "entityType":"Organization",
        "relationshipType":"IsOdiParentOf"
      }
    }
  }
 ]
}
```


親の取得

親の取得 REST API は、ビジネスエンティティレコードの直接の親リレーションを返します。

この API は GET メソッドを使用します。

要求 URL

［親の取得］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/hierarchy/<hierarchy name>/entity/<entity name>/<entity ID>/parent
```

［親の取得］URL に対して次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/<context>/<database ID>/hierarchy/<hierarchy name>/entity/<entity name>/<entity ID>/
parent
```

クエリパラメータ

クエリパラメータを要求 URL に追加して、リレーションをフィルタリングできます。

以下の表に、クエリパラメータを示します。

パラメータ	説明
effectiveDate	リレーションの取得対象となる日付。
interactionId	相互作用の ID。
readLabelsOnly	結果でラベルのみを返すかどうかを示します。値は true または false です。
readSystemFields	結果でシステムフィールドを返すかどうかを示します。値は true または false です。

パラメータ	説明
recordsToReturn	返す行の数を指定します。
rejectInteractionId	変更を却下するための相互作用の ID。
relationshipType	リレーションタイプのカンマ区切りリスト。例えば、relationshipType=Employee,Employer リストは Employee および Employer リレーションタイプに関連するレコードを返します。
returnTotal	結果セット内のレコード数を返します。結果セット内のレコード数を取得するには、true に設定します。
rootId	階層のルートの ID。

サンプル API 要求

次のサンプル要求は、ビジネスエンティティレコードの直接の親リレーションを一覧表示します。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/hierarchy/ODI/entity/Organization/55/parent
```

サンプル API 応答

次のサンプル応答は、ビジネスエンティティレコードの直接の親リレーションを示しています。

```
{
  "link":[],
  "firstRecord":1,
  "pageSize":10,
  "searchToken":"SVR1.1G7VH",
  "relatedEntity":[
    {
      "businessEntity":{
        "Organization":{
          "link":[
            {
              "href":"http://localhost:8080/cmx/cs/localhost-orcl-MDM_SAMPLE/Organization/584",
              "rel":"self"
            },
            {
              "href":"http://localhost:8080/cmx/request/hm_icons/hierarchymanager/Hospital/Hospital.png?
ors=localhost-orcl-MDM_SAMPLE",
              "rel":"icon"
            }
          ],
          "rowidObject":"584",
          "label":"Time Warner Inc",
          "partyType":"Organization",
          "displayName":"Time Warner Inc",
          "dunsNumber":"799527630"
        }
      },
      "entityLabel":"Time Warner Inc",
      "relationshipLabel":"is ODI parent of",
      "relationship":{
        "rowidObject":"834",
        "label":"is ODI parent of",
        "rowidRelType":"2",
        "rowidHierarchy":"1",
        "relName":"Parent",
        "relDesc":"Parent",
        "from":{
          "rowidObject":"584"
        },
```

```
        "to":{
          "rowidObject":"55"
        }
      },
      "entityType":"Organization",
      "relationshipType":"IsOdiParentOf"
    }
  ]
}
```

子の取得

子の取得 REST API は、ビジネスエンティティレコードの子リレーションを返します。

この API は GET メソッドを使用します。

要求 URL

［子の取得］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/hierarchy/<hierarchy name>/entity/<entity name>/<entity ID>/children
```

［子の取得］URL に対して次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/<context>/<database ID>/hierarchy/<hierarchy name>/entity/<entity name>/<entity ID>/children
```

クエリパラメータ

クエリパラメータを要求 URL に追加して、リレーションをフィルタリングできます。

以下の表に、クエリパラメータを示します。

パラメータ	説明
effectiveDate	リレーションの取得対象となる日付。
interactionId	相互作用の ID。
readLabelsOnly	結果でラベルのみを返すかどうかを示します。値は true または false です。
readSystemFields	結果でシステムフィールドを返すかどうかを示します。値は true または false です。
recordsToReturn	返す行の数を指定します。
rejectInteractionId	変更を却下するための相互作用の ID。
relationshipType	リレーションタイプのカンマ区切りリスト。例えば、relationshipType=Employee,Employer リストは Employee および Employer リレーションタイプに関連するレコードを返します。
returnTotal	結果セット内のレコード数を返します。結果セット内のレコード数を取得するには、true に設定します。
rootId	階層のルートの ID。

サンプル API 要求

次のサンプル要求は、ビジネスエンティティレコードの子リレーションを一覧表示します。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/hierarchy/ODI/entity/Organization/55/children
```

サンプル API 応答

次のサンプル応答は、ビジネスエンティティレコードの子リレーションを示しています。


```
{
  "link":[],
  "firstRecord":1,
  "pageSize":10,
  "searchToken":"SVR1.1G7VF",
  "relatedEntity":[
    {
      "businessEntity":{
        "Organization":{
          "link":[
            {
              "href":"http://localhost:8080/cmx/request/hm_icons/hierarchymanager/Hospital/Hospital.png?
ors=localhost-orcl-MDM_SAMPLE",
              "rel":"icon"
            },
            {
              "href":"http://localhost:8080/cmx/cs/localhost-orcl-MDM_SAMPLE/Organization/593",
              "rel":"self"
            }
          ],
          "rowidObject":"593           ",
          "label":"Turner Broadcasting System Inc",
          "partyType":"Organization",
          "displayName":"Turner Broadcasting System Inc",
          "dunsNumber":"003319068"
        }
      },
      "entityLabel":"Turner Broadcasting System Inc",
      "relationshipLabel":"is ODI parent of",
      "relationship":{
        "rowidObject":"887           ",
        "label":"is ODI parent of",
        "rowidRelType":"2             ",
        "rowidHierarchy":"1             ",
        "relName":"Parent",
        "relDesc":"Parent",
        "from":{
          "rowidObject":"55            "
        },
        "to":{
          "rowidObject":"593           "
        }
      },
      "entityType":"Organization",
      "relationshipType":"IsOdiParentOf"
    },
    {
      "businessEntity":{
        "Organization":{
          "link":[
            {
              "href":"http://localhost:8080/cmx/request/hm_icons/hierarchymanager/Hospital/Hospital.png?
ors=localhost-orcl-MDM_SAMPLE",
              "rel":"icon"
            },
            {
              "href":"http://localhost:8080/cmx/cs/localhost-orcl-MDM_SAMPLE/Organization/414",
              "rel":"self"
            }
          ],
```

```
          "rowidObject":"414           ",
          "label":"Historic TW Inc",
          "partyType":"Organization",
          "displayName":"Historic TW Inc",
          "dunsNumber":"007305290"
        }
      },
      "entityLabel":"Historic TW Inc",
      "relationshipLabel":"is ODI parent of",
      "relationship":{
        "rowidObject":"888           ",
        "label":"is ODI parent of",
        "rowidRelType":"2             ",
        "rowidHierarchy":"1             ",
        "relName":"HQ",
        "relDesc":"HQ",
        "from":{
          "rowidObject":"55            "
        },
        "to":{
          "rowidObject":"414           "
        }
      },
      "entityType":"Organization",
      "relationshipType":"IsOdiParentOf"
    },
    {
      "businessEntity":{
        "Organization":{
          "link":[
            {
              "href":"http://localhost:8080/cmx/cs/localhost-orcl-MDM_SAMPLE/Organization/35",
              "rel":"self"
            },
            {
              "href":"http://localhost:8080/cmx/request/hm_icons/hierarchymanager/Hospital/Hospital.png?
ors=localhost-orcl-MDM_SAMPLE",
              "rel":"icon"
            }
          ],
          "rowidObject":"35            ",
          "label":"People Magazine",
          "partyType":"Organization",
          "displayName":"People Magazine",
          "dunsNumber":"011584096"
        }
      },
      "entityLabel":"People Magazine",
      "relationshipLabel":"is ODI parent of",
      "relationship":{
        "rowidObject":"889           ",
        "label":"is ODI parent of",
        "rowidRelType":"2             ",
        "rowidHierarchy":"1             ",
        "relName":"Parent",
        "relDesc":"Parent",
        "from":{
          "rowidObject":"55            "
        },
        "to":{
          "rowidObject":"35            "
        }
      },
      "entityType":"Organization",
      "relationshipType":"IsOdiParentOf"
    },
    {
      "businessEntity":{
        "Organization":{
          "link":[
            {
```

```
              "href":"http://localhost:8080/cmx/cs/localhost-orcl-MDM_SAMPLE/Organization/391",
              "rel":"self"
            },
            {
              "href":"http://localhost:8080/cmx/request/hm_icons/hierarchymanager/Hospital/Hospital.png?
ors=localhost-orcl-MDM_SAMPLE",
              "rel":"icon"
            }
          ],
          "rowidObject":"391           ",
          "label":"Historic TW Inc",
          "partyType":"Organization",
          "displayName":"Historic TW Inc",
          "dunsNumber":"011816704"
        }
      },
      "entityLabel":"Historic TW Inc",
      "relationshipLabel":"is ODI parent of",
      "relationship":{
        "rowidObject":"890           ",
        "label":"is ODI parent of",
        "rowidRelType":"2            ",
        "rowidHierarchy":"1            ",
        "relName":"HQ",
        "relDesc":"HQ",
        "from":{
          "rowidObject":"55            "
        },
        "to":{
          "rowidObject":"391           "
        }
      },
      "entityType":"Organization",
      "relationshipType":"IsOdiParentOf"
    },
    {
      "businessEntity":{
        "Organization":{
          "link":[
            {
              "href":"http://localhost:8080/cmx/cs/localhost-orcl-MDM_SAMPLE/Organization/448",
              "rel":"self"
            },
            {
              "href":"http://localhost:8080/cmx/request/hm_icons/hierarchymanager/Hospital/Hospital.png?
ors=localhost-orcl-MDM_SAMPLE",
              "rel":"icon"
            }
          ],
          "rowidObject":"448           ",
          "label":"Historic TW Inc",
          "partyType":"Organization",
          "displayName":"Historic TW Inc",
          "dunsNumber":"069110711"
        }
      },
      "entityLabel":"Historic TW Inc",
      "relationshipLabel":"is ODI parent of",
      "relationship":{
        "rowidObject":"891           ",
        "label":"is ODI parent of",
        "rowidRelType":"2            ",
        "rowidHierarchy":"1            ",
        "relName":"HQ",
        "relDesc":"HQ",
        "from":{
          "rowidObject":"55            "
        },
        "to":{
          "rowidObject":"448           "
        }
```

```
      },
      "entityType":"Organization",
      "relationshipType":"IsOdiParentOf"
    },
    {
      "businessEntity":{
        "Organization":{
          "link":[
            {
              "href":"http://localhost:8080/cmx/request/hm_icons/hierarchymanager/Hospital/Hospital.png?
ors=localhost-orcl-MDM_SAMPLE",
              "rel":"icon"
            },
            {
              "href":"http://localhost:8080/cmx/cs/localhost-orcl-MDM_SAMPLE/Organization/545",
              "rel":"self"
            }
          ],
          "rowidObject":"545           ",
          "label":"Historic TW Inc",
          "partyType":"Organization",
          "displayName":"Historic TW Inc",
          "dunsNumber":"072077139"
        }
      },
      "entityLabel":"Historic TW Inc",
      "relationshipLabel":"is ODI parent of",
      "relationship":{
        "rowidObject":"892           ",
        "label":"is ODI parent of",
        "rowidRelType":"2             ",
        "rowidHierarchy":"1              ",
        "relName":"HQ",
        "relDesc":"HQ",
        "from":{
          "rowidObject":"55            "
        },
        "to":{
          "rowidObject":"545           "
        }
      },
      "entityType":"Organization",
      "relationshipType":"IsOdiParentOf"
    },
    {
      "businessEntity":{
        "Organization":{
          "link":[
            {
              "href":"http://localhost:8080/cmx/cs/localhost-orcl-MDM_SAMPLE/Organization/349",
              "rel":"self"
            },
            {
              "href":"http://localhost:8080/cmx/request/hm_icons/hierarchymanager/Hospital/Hospital.png?
ors=localhost-orcl-MDM_SAMPLE",
              "rel":"icon"
            }
          ],
          "rowidObject":"349           ",
          "label":"Historic TW Inc",
          "partyType":"Organization",
          "displayName":"Historic TW Inc",
          "dunsNumber":"074056123"
        }
      },
      "entityLabel":"Historic TW Inc",
      "relationshipLabel":"is ODI parent of",
      "relationship":{
        "rowidObject":"893           ",
        "label":"is ODI parent of",
        "rowidRelType":"2             ",
```

```
        "rowidHierarchy":"1             ",
        "relName":"HQ",
        "relDesc":"HQ",
        "from":{
          "rowidObject":"55            "
        },
        "to":{
          "rowidObject":"349           "
        }
      },
      "entityType":"Organization",
      "relationshipType":"IsOdiParentOf"
    },
    {
      "businessEntity":{
        "Organization":{
          "link":[
            {
              "href":"http://localhost:8080/cmx/request/hm_icons/hierarchymanager/Hospital/Hospital.png?
ors=localhost-orcl-MDM_SAMPLE",
              "rel":"icon"
            },
            {
              "href":"http://localhost:8080/cmx/cs/localhost-orcl-MDM_SAMPLE/Organization/298",
              "rel":"self"
            }
          ],
          "rowidObject":"298           ",
          "label":"WF Cinema Holdings LP",
          "partyType":"Organization",
          "displayName":"WF Cinema Holdings LP",
          "dunsNumber":"075408711"
        }
      },
      "entityLabel":"WF Cinema Holdings LP",
      "relationshipLabel":"is ODI parent of",
      "relationship":{
        "rowidObject":"894           ",
        "label":"is ODI parent of",
        "rowidRelType":"2             ",
        "rowidHierarchy":"1             ",
        "relName":"Parent",
        "relDesc":"Parent",
        "from":{
          "rowidObject":"55            "
        },
        "to":{
          "rowidObject":"298           "
        }
      },
      "entityType":"Organization",
      "relationshipType":"IsOdiParentOf"
    },
    {
      "businessEntity":{
        "Organization":{
          "link":[
            {
              "href":"http://localhost:8080/cmx/request/hm_icons/hierarchymanager/Hospital/Hospital.png?
ors=localhost-orcl-MDM_SAMPLE",
              "rel":"icon"
            },
            {
              "href":"http://localhost:8080/cmx/cs/localhost-orcl-MDM_SAMPLE/Organization/596",
              "rel":"self"
            }
          ],
          "rowidObject":"596           ",
          "label":"Historic TW Inc",
          "partyType":"Organization",
          "displayName":"Historic TW Inc",
```

```
          "dunsNumber":"076749246"
        }
      },
      "entityLabel":"Historic TW Inc",
      "relationshipLabel":"is ODI parent of",
      "relationship":{
        "rowidObject":"895           ",
        "label":"is ODI parent of",
        "rowidRelType":"2             ",
        "rowidHierarchy":"1             ",
        "relName":"HQ",
        "relDesc":"HQ",
        "from":{
          "rowidObject":"55            "
        },
        "to":{
          "rowidObject":"596           "
        }
      },
      "entityType":"Organization",
      "relationshipType":"IsOdiParentOf"
    },
    {
      "businessEntity":{
        "Organization":{
          "link":[
            {
              "href":"http://localhost:8080/cmx/request/hm_icons/hierarchymanager/Hospital/Hospital.png?
ors=localhost-orcl-MDM_SAMPLE",
              "rel":"icon"
            },
            {
              "href":"http://localhost:8080/cmx/cs/localhost-orcl-MDM_SAMPLE/Organization/699",
              "rel":"self"
            }
          ],
          "rowidObject":"699           ",
          "label":"Historic TW Inc",
          "partyType":"Organization",
          "displayName":"Historic TW Inc",
          "dunsNumber":"083115568"
        }
      },
      "entityLabel":"Historic TW Inc",
      "relationshipLabel":"is ODI parent of",
      "relationship":{
        "rowidObject":"896           ",
        "label":"is ODI parent of",
        "rowidRelType":"2             ",
        "rowidHierarchy":"1             ",
        "relName":"HQ",
        "relDesc":"HQ",
        "from":{
          "rowidObject":"55            "
        },
        "to":{
          "rowidObject":"699           "
        }
      },
      "entityType":"Organization",
      "relationshipType":"IsOdiParentOf"
    }
  ]
}
```


階層のエクスポート

階層のエクスポート REST API は、階層を CSV 形式でエクスポートします。

この API は GET メソッドを使用します。

要求 URL

［階層のエクスポート］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/hierarchy/<hierarchy name>/entity/<entity name>/<entity ID>/
exportHierarchy
```

［階層のエクスポート］URL に対して、次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/<context>/<database ID>/hierarchy/<hierarchy name>/entity/<entity name>/<entity ID>/
exportHierarchy
```

クエリパラメータ

クエリパラメータを要求 URL に追加して、リレーションをフィルタリングできます。

以下の表に、クエリパラメータを示します。

パラメータ	説明
effectiveDate	リレーションの取得対象となる日付。
interactionId	相互作用の ID。
readLabelsOnly	結果でラベルのみを返すかどうかを示します。値は true または false です。
readSystemFields	結果でシステムフィールドを返すかどうかを示します。値は true または false です。
recordsToReturn	返す行の数を指定します。
rejectInteractionId	変更を却下するための相互作用の ID。
relationshipType	リレーションタイプのカンマ区切りリスト。例えば、relationshipType=Employee,Employer リストは Employee および Employer リレーションタイプに関連するレコードを返します。
returnTotal	結果セット内のレコード数を返します。結果セット内のレコード数を取得するには、true に設定します。
rootId	階層のルートの ID。

サンプル API 要求

次のサンプル要求は、ビジネスエンティティレコードの階層をエクスポートします。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/hierarchy/ODI/entity/Organization/584/exportHierarchy
```

サンプル API 応答

次のサンプル応答は、CSV 形式でエクスポートされた階層を示しています。

```
Level,Business Entity,ID,Label,Parent ID,Parent Label,Type of Relationship to
Parent,issuingCompany,expirationYear,accountType,accountNumber,securityCode,expirationMonth,cardholderName,prod
uctType,productNumber,productName,inceptionDate,productDesc,groupType,groupName,partyType,lastName,displayName,
middleName,birthdate,firstName,dunsNumber
Root,Organization,584,Time Warner Inc,,,,,,,,,,,,,,,,,,Organization,,Time Warner Inc,,,,799527630
1,Organization,55,Historic TW Inc,584,Time Warner Inc,is ODI parent of,,,,,,,,,,,,,,Organization,,Historic TW
Inc,,,,958466278
1,Organization,206,Courtroom Television Network LLC,584,Time Warner Inc,is ODI parent
of,,,,,,,,,,,,,,,Organization,,Courtroom Television Network LLC,,,,043905707
1,Organization,674,Rebellion Pictures LLC,584,Time Warner Inc,is ODI parent
of,,,,,,,,,,,,,,,Organization,,Rebellion Pictures LLC,,,,557414278
```

```
1,Organization,881,"Time Warner Cable Programming, Inc",584,Time Warner Inc,is ODI parent
of,,,,,,,,,,,,,,,Organization,,"Time Warner Cable Programming, Inc",,,,787474949
...
```

直接の子と親のエクスポート

直接の子と親のエクスポート REST API は、ビジネスエンティティレコードの直接の子と親のリレーションを CSV 形式でエクスポートします。

この API は GET メソッドを使用します。

要求 URL

［直接の子と親のエクスポート］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/hierarchy/<hierarchy name>/entity/<entity name>/<entity ID>/
exportRelated
```

［直接の子と親のエクスポート］URL に対して次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/<context>/<database ID>/hierarchy/<hierarchy name>/entity/<entity name>/<entity ID>/
exportRelated
```

クエリパラメータ

クエリパラメータを要求 URL に追加して、リレーションをフィルタリングできます。

以下の表に、クエリパラメータを示します。

パラメータ	説明
effectiveDate	リレーションの取得対象となる日付。
interactionId	相互作用の ID。
readLabelsOnly	結果でラベルのみを返すかどうかを示します。値は true または false です。
readSystemFields	結果でシステムフィールドを返すかどうかを示します。値は true または false です。
recordsToReturn	返す行の数を指定します。
rejectInteractionId	変更を却下するための相互作用の ID。
relationshipType	リレーションタイプのカンマ区切りリスト。例えば、relationshipType=Employee,Employer リストは Employee および Employer リレーションタイプに関連するレコードを返します。
returnTotal	結果セット内のレコード数を返します。結果セット内のレコード数を取得するには、true に設定します。
rootId	階層のルートの ID。

サンプル API 要求

次のサンプル要求は、ビジネスエンティティレコードの直接の子と親のリレーションをエクスポートします。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/hierarchy/ODI/entity/Organization/55/exportRelated
```

サンプル API 応答

次のサンプル応答は、CSV 形式でビジネスエンティティレコードのエクスポートされた直接の子と親のリレーションを示しています。

```
Level,Business Entity,ID,Label,Parent ID,Parent Label,Type of Relationship to
Parent,issuingCompany,expirationYear,accountType,accountNumber,securityCode,expirationMonth,cardholderName,prod
uctType,productNumber,productName,inceptionDate,productDesc,groupType,groupName,partyType,lastName,displayName,
middleName,birthdate,firstName,dunsNumber
Root,Organization,584,Time Warner Inc,,,,,,,,,,,,,,,,,Organization,,Time Warner Inc,,,,799527630
1,Organization,55,Historic TW Inc,584,Time Warner Inc,is ODI parent of,,,,,,,,,,,,,,Organization,,Historic TW
Inc,,,,958466278
2,Organization,593,Turner Broadcasting System Inc,55,Historic TW Inc,is ODI parent
of,,,,,,,,,,,,,,,Organization,,Turner Broadcasting System Inc,,,,003319068
2,Organization,414,Historic TW Inc,55,Historic TW Inc,is ODI parent of,,,,,,,,,,,,,,,Organization,,Historic TW
Inc,,,,007305290
2,Organization,35,People Magazine,55,Historic TW Inc,is ODI parent of,,,,,,,,,,,,,,,Organization,,People
Magazine,,,,011584096
2,Organization,391,Historic TW Inc,55,Historic TW Inc,is ODI parent of,,,,,,,,,,,,,,,Organization,,Historic TW
Inc,,,,011816704
...
```

階層変更の取得

階層変更の取得 API は、階層への保留中の変更を取得します。

この API は GET メソッドを使用します。

要求 URL

［階層変更の取得］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/hierarchy/{hierarchy name}/changes
```

［階層変更の取得］URL に対して次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/<context>/<database ID>/hierarchy/{hierarchy name}/changes
```

クエリパラメータ

クエリパラメータを要求 URL に追加して、リレーションをフィルタリングできます。

以下の表に、クエリパラメータを示します。

パラメータ	説明
effectiveDate	リレーションの取得対象となる日付。
interactionId	相互作用の ID。
readLabelsOnly	結果でラベルのみを返すかどうかを示します。値は true または false です。
readSystemFields	結果でシステムフィールドを返すかどうかを示します。値は true または false です。
recordsToReturn	返す行の数を指定します。
rejectInteractionId	変更を却下するための相互作用の ID。
relationshipType	リレーションタイプのカンマ区切りリスト。例えば、relationshipType=Employee,Employer リストは Employee および Employer リレーションタイプに関連するレコードを返します。

パラメータ	説明
returnTotal	結果セット内のレコード数を返します。結果セット内のレコード数を取得するには、true に設定します。
rootId	階層のルートの ID。

サンプル API 要求

次の要求は、相互作用 ID、ルート ID、却下相互作用 ID でフィルタリングされた階層に保留中の変更を返します。

```
GET http://localhost:8080/cmx/bulk/localhost-orcl-DS_UI1/hierarchy/Product/changes?
interactionId=480003&rootId=165&rejectInteractionId=480004
```

サンプル API 応答

次のサンプル応答は、保留中の変更を示しています。

```
{
  "interactionId":"480003",
  "rejectInteractionId":"480004",
  "type":"hierarchy",
  "name":"Product",
  "objects":[
    {
      "before":[

      ],
      "after":[
        {
          "businessEntity":{
            "ProductGroup":{
              "link":[
                {
                  "href":"http://localhost:8080/cmx/request/hm_icons/hierarchymanager/Group/Group.png?
ors=localhost-orcl-MDM_SAMPLE",
                  "rel":"icon"
                }
              ],
              "rowidObject":"165            ",
              "creator":"admin",
              "createDate":"2020-01-13T13:07:35.128-05:00",
              "updatedBy":"admin",
              "lastUpdateDate":"2020-01-13T13:07:35.128-05:00",
              "consolidationInd":4,
              "lastRowidSystem":"PRODUCT       ",
              "hubStateInd":1,
              "label":"Vehicles",
              "productType":"Product Group",
              "productName":"Vehicles"
            }
          },
          "entityLabel":"Vehicles",
          "entityType":"ProductGroup",
          "depth":1,
          "object":{
            "businessEntity":{
              "ProductGroup":{
                "link":[
                  {
                    "href":"http://localhost:8080/cmx/request/hm_icons/hierarchymanager/Group/Group.png?
ors=localhost-orcl-MDM_SAMPLE",
                    "rel":"icon"
                  }
```

```
                ],
                "rowidObject":"166           ",
                "label":"Scooter",
                "productType":"Product Group",
                "productName":"Scooter"
              }
            },
            "entityLabel":"Scooter",
            "relationshipLabel":"Product Group is parent of Product Group",
            "relationship":{
              "rowidObject":"124           ",
              "creator":"datasteward",
              "createDate":"2020-01-13T13:07:35.864-05:00",
              "updatedBy":"datasteward",
              "lastUpdateDate":"2020-01-13T13:07:35.865-05:00",
              "consolidationInd":4,
              "lastRowidSystem":"PRODUCT       ",
              "interactionId":"480003",
              "hubStateInd":0,
              "label":"Product Group is parent of Product Group",
              "rowidRelType":"7             ",
              "rowidHierarchy":"3             ",
              "from":{
                "rowidObject":"165           "
              },
              "to":{
                "rowidObject":"166           "
              }
            },
            "entityType":"ProductGroup",
            "relationshipType":"ProductGroupIsParentOfProductGroup"
          }
        },
        {
          "businessEntity":{
            "ProductGroup":{
              "link":[
                {
                  "href":"http://localhost:8080/cmx/request/hm_icons/hierarchymanager/Group/Group.png?
ors=localhost-orcl-MDM_SAMPLE",
                  "rel":"icon"
                }
              ],
              "rowidObject":"165           ",
              "creator":"admin",
              "createDate":"2020-01-13T13:07:35.128-05:00",
              "updatedBy":"admin",
              "lastUpdateDate":"2020-01-13T13:07:35.128-05:00",
              "consolidationInd":4,
              "lastRowidSystem":"PRODUCT       ",
              "hubStateInd":1,
              "label":"Vehicles",
              "productType":"Product Group",
              "productName":"Vehicles"
            }
          },
          "entityLabel":"Vehicles",
          "entityType":"ProductGroup",
          "depth":2,
          "object":{
            "businessEntity":{
              "ProductGroup":{
                "link":[
                  {
                    "href":"http://localhost:8080/cmx/request/hm_icons/hierarchymanager/Group/Group.png?
ors=localhost-orcl-MDM_SAMPLE",
                    "rel":"icon"
                  }
                ],
                "rowidObject":"166           ",
                "creator":"admin",
```

```
            "createDate":"2020-01-13T13:07:35.250-05:00",
            "updatedBy":"admin",
            "lastUpdateDate":"2020-01-13T13:07:35.250-05:00",
            "consolidationInd":4,
            "lastRowidSystem":"PRODUCT       ",
            "hubStateInd":1,
            "label":"Scooter",
            "productType":"Product Group",
            "productName":"Scooter"
          }
        },
        "entityLabel":"Scooter",
        "relationshipLabel":"Product Group is parent of Product Group",
        "relationship":{
          "rowidObject":"124           ",
          "creator":"datasteward",
          "createDate":"2020-01-13T13:07:35.864-05:00",
          "updatedBy":"datasteward",
          "lastUpdateDate":"2020-01-13T13:07:35.865-05:00",
          "consolidationInd":4,
          "lastRowidSystem":"PRODUCT       ",
          "interactionId":"480003",
          "hubStateInd":0,
          "label":"Product Group is parent of Product Group",
          "rowidRelType":"7             ",
          "rowidHierarchy":"3             ",
          "from":{
            "rowidObject":"165           "
          },
          "to":{
            "rowidObject":"166           "
          }
        },
        "entityType":"ProductGroup",
        "relationshipType":"ProductGroupIsParentOfProductGroup",
        "object":{
          "businessEntity":{
            "ProductGroup":{
              "link":[
                {
                  "href":"http://localhost:8080/cmx/request/hm_icons/hierarchymanager/Group/Group.png?
ors=localhost-orcl-MDM_SAMPLE",
                  "rel":"icon"
                }
              ],
              "rowidObject":"167           ",
              "label":"E-Scooter",
              "productType":"Product Group",
              "productName":"E-Scooter"
            }
          },
          "entityLabel":"E-Scooter",
          "relationshipLabel":"Product Group is parent of Product Group",
          "relationship":{
            "rowidObject":"125           ",
            "creator":"datasteward",
            "createDate":"2020-01-13T13:07:36.033-05:00",
            "updatedBy":"datasteward",
            "lastUpdateDate":"2020-01-13T13:07:36.034-05:00",
            "consolidationInd":4,
            "lastRowidSystem":"PRODUCT       ",
            "interactionId":"480003",
            "hubStateInd":0,
            "label":"Product Group is parent of Product Group",
            "rowidRelType":"7             ",
            "rowidHierarchy":"3             ",
            "from":{
              "rowidObject":"166           "
            },
            "to":{
              "rowidObject":"167           "
```

```
                }
              },
              "entityType":"ProductGroup",
              "relationshipType":"ProductGroupIsParentOfProductGroup"
            }
          }
        },
        {
          "businessEntity":{
            "ProductGroup":{
              "link":[
                {
                  "href":"http://localhost:8080/cmx/request/hm_icons/hierarchymanager/Group/Group.png?
ors=localhost-orcl-MDM_SAMPLE",
                  "rel":"icon"
                }
              ],
              "rowidObject":"165            ",
              "creator":"admin",
              "createDate":"2020-01-13T13:07:35.128-05:00",
              "updatedBy":"admin",
              "lastUpdateDate":"2020-01-13T13:07:35.128-05:00",
              "consolidationInd":4,
              "lastRowidSystem":"PRODUCT       ",
              "hubStateInd":1,
              "label":"Vehicles",
              "productType":"Product Group",
              "productName":"Vehicles"
            }
          },
          "entityLabel":"Vehicles",
          "entityType":"ProductGroup",
          "depth":3,
          "object":{
            "businessEntity":{
              "ProductGroup":{
                "link":[
                  {
                    "href":"http://localhost:8080/cmx/request/hm_icons/hierarchymanager/Group/Group.png?
ors=localhost-orcl-MDM_SAMPLE",
                    "rel":"icon"
                  }
                ],
                "rowidObject":"166            ",
                "creator":"admin",
                "createDate":"2020-01-13T13:07:35.250-05:00",
                "updatedBy":"admin",
                "lastUpdateDate":"2020-01-13T13:07:35.250-05:00",
                "consolidationInd":4,
                "lastRowidSystem":"PRODUCT       ",
                "hubStateInd":1,
                "label":"Scooter",
                "productType":"Product Group",
                "productName":"Scooter"
              }
            },
            "entityLabel":"Scooter",
            "relationshipLabel":"Product Group is parent of Product Group",
            "relationship":{
              "rowidObject":"124            ",
              "creator":"datasteward",
              "createDate":"2020-01-13T13:07:35.864-05:00",
              "updatedBy":"datasteward",
              "lastUpdateDate":"2020-01-13T13:07:35.865-05:00",
              "consolidationInd":4,
              "lastRowidSystem":"PRODUCT       ",
              "interactionId":"480003",
              "hubStateInd":0,
              "label":"Product Group is parent of Product Group",
              "rowidRelType":"7              ",
              "rowidHierarchy":"3             ",
```

```
            "from":{
              "rowidObject":"165           "
            },
            "to":{
              "rowidObject":"166           "
            }
          },
          "entityType":"ProductGroup",
          "relationshipType":"ProductGroupIsParentOfProductGroup",
          "object":{
            "businessEntity":{
              "ProductGroup":{
                "link":[
                  {
                    "href":"http://localhost:8080/cmx/request/hm_icons/hierarchymanager/Group/Group.png?
ors=localhost-orcl-MDM_SAMPLE",
                    "rel":"icon"
                  }
                ],
                "rowidObject":"167           ",
                "creator":"admin",
                "createDate":"2020-01-13T13:07:35.364-05:00",
                "updatedBy":"admin",
                "lastUpdateDate":"2020-01-13T13:07:35.364-05:00",
                "consolidationInd":4,
                "lastRowidSystem":"PRODUCT       ",
                "hubStateInd":1,
                "label":"E-Scooter",
                "productType":"Product Group",
                "productName":"E-Scooter"
              }
            },
            "entityLabel":"E-Scooter",
            "relationshipLabel":"Product Group is parent of Product Group",
            "relationship":{
              "rowidObject":"125           ",
              "creator":"datasteward",
              "createDate":"2020-01-13T13:07:36.033-05:00",
              "updatedBy":"datasteward",
              "lastUpdateDate":"2020-01-13T13:07:36.034-05:00",
              "consolidationInd":4,
              "lastRowidSystem":"PRODUCT       ",
              "interactionId":"480003",
              "hubStateInd":0,
              "label":"Product Group is parent of Product Group",
              "rowidRelType":"7             ",
              "rowidHierarchy":"3             ",
              "from":{
                "rowidObject":"166           "
              },
              "to":{
                "rowidObject":"167           "
              }
            },
            "entityType":"ProductGroup",
            "relationshipType":"ProductGroupIsParentOfProductGroup",
            "object":{
              "businessEntity":{
                "Products":{
                  "link":[
                    {
                      "href":"http://localhost:8080/cmx/request/hm_icons/HM_Icons/BulletSquareBlue/
bullet_square_blue.png?ors=localhost-orcl-MDM_SAMPLE",
                      "rel":"icon"
                    }
                  ],
                  "rowidObject":"168           ",
                  "label":"Xiaomi Mi Electric Scooter",
                  "productType":"Product",
                  "productName":"Xiaomi Mi Electric Scooter"
                }
```

```
                },
                "entityLabel":"Xiaomi Mi Electric Scooter",
                "relationshipLabel":"Product Group is parent of Product",
                "relationship":{
                  "rowidObject":"126           ",
                  "creator":"datasteward",
                  "createDate":"2020-01-13T13:07:36.096-05:00",
                  "updatedBy":"datasteward",
                  "lastUpdateDate":"2020-01-13T13:07:36.096-05:00",
                  "consolidationInd":4,
                  "lastRowidSystem":"PRODUCT       ",
                  "interactionId":"480003",
                  "hubStateInd":0,
                  "label":"Product Group is parent of Product",
                  "rowidRelType":"9              ",
                  "rowidHierarchy":"3             ",
                  "from":{
                    "rowidObject":"167           "
                  },
                  "to":{
                    "rowidObject":"168           "
                  }
                },
                "entityType":"Products",
                "relationshipType":"ProductGroupIsParentOfProduct"
              }
            }
          }
        }
      ]
    }
  ]
}
```


一括リレーション変更

一括リレーション変更 API は、ネットワークおよび階層のリレーションを作成、更新および削除します。

この API は POST メソッドを使用します。

要求 URL

［一括リレーション変更］URL の形式は次のとおりです。

`http://<host>:<port>/<context>/<database ID>`

［階層変更の取得］URL に対して次の HTTP POST 要求を行います。

`POST http://<host>:<port>/<context>/<database ID>`

パラメータ

実行する変更のタイプと修正するリレーションを指定します。

次の表に、要求本文内のパラメータを示します。

パラメータ	説明
operation	実行する変更のタイプ。値は、createRelationship、updateRelationship、および deleteRelationship です。
relationship	リレーションの名前。

パラメータ	説明
rowidObject	オブジェクトの行 ID。
startDate	オプション。レコードが有効になる日付を指定します。このパラメータは、タイムラインが有効になったベースオブジェクトに指定します。
endDate	オプション。レコードが無効になる日付を指定します。このパラメータは、タイムラインが有効になったベースオブジェクトに指定します。

サンプル API 要求

次の要求は、新規リレーションを作成します。

```
POST http://localhost:8080/cmx/bulk/localhost-orcl-DS_UI1
```

```
{
  "systemName":"Product",
  "rootRowId":"165",
  "items":[
    {
      "operation":"createRelationship",
      "relationship":"ProductGroupIsParentOfProductGroup",
      "payload":{
        "from":{
          "rowidObject":"165"
        },
        "to":{
          "rowidObject":"166"
        }
      }
    },
    {
      "operation":"createRelationship",
      "relationship":"ProductGroupIsParentOfProductGroup",
      "payload":{
        "from":{
          "rowidObject":"166"
        },
        "to":{
          "rowidObject":"167"
        }
      }
    },
    {
      "operation":"createRelationship",
      "relationship":"ProductGroupIsParentOfProduct",
      "payload":{
        "from":{
          "rowidObject":"167"
        },
        "to":{
          "rowidObject":"168"
        }
      },
      "effectivePeriod":{
        "startDate":"2019-01-01T11:11:53.974-04:00",
        "endDate":"2039-12-31T11:11:53.974-04:00"
      }
    }
  ]
}
```

サンプル API 応答

次のサンプル応答は、変更を昇格および却下するために使用できる、変更に関連付けられた相互作用 ID を示しています。


```
{
  "processId":"14068",
  "interactionId":"480003",
  "rejectInteractionId":"480004",
  "items":[
    {
      "payload":{
        "key":{
          "rowid":"124",
          "sourceKey":"SVR1.1G9AG"
        },
        "rowidObject":"124",
        "from":{
          "key":{
            "rowid":"165"
          },
          "rowidObject":"165"
        },
        "to":{
          "key":{
            "rowid":"166"
          },
          "rowidObject":"166"
        }
      }
    },
    {
      "payload":{
        "key":{
          "rowid":"125",
          "sourceKey":"SVR1.1G9AH"
        },
        "rowidObject":"125",
        "from":{
          "key":{
            "rowid":"166"
          },
          "rowidObject":"166"
        },
        "to":{
          "key":{
            "rowid":"167"
          },
          "rowidObject":"167"
        }
      }
    },
    {
      "payload":{
        "key":{
          "rowid":"126",
          "sourceKey":"SVR1.1G9AI"
        },
        "rowidObject":"126",
        "from":{
          "key":{
            "rowid":"167"
          },
          "rowidObject":"167"
        },
        "to":{
          "key":{
            "rowid":"168"
          },
          "rowidObject":"168"
        }
```

```
      }
    }
  ]
}
```

一括昇格

一括昇格 REST API は、ネットワークまたは階層への変更を昇格します。

この API は POST メソッドを使用します。

要求 URL

［一括昇格］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/promote
```

［一括昇格］URL に対して、次の HTTP GET 要求を行います。

```
POST http://<host>:<port>/<context>/<database ID>/promote
```

サンプル API 要求

次のサンプル要求は、変更を昇格させます。

```
POST http://localhost:8080/cmx/bulk/localhost-orcl-DS_UI1/promote
{
  "interactionId":"67080007",
  "rejectInteractionId":"67080008",
  "objects":[
    {
      "type":"Hierarchy",
      "name":"HPerson"
    }
  ]
}
```

サンプル API 応答

次のサンプル応答は、変更を正常に昇格した後の応答を示しています。

```
{
    "interactionId": "68520005",
    "rejectInteractionId": "68520006",
    "type": "hierarchy",
    "name": "HPerson",
    "objects": [
        {
            "before": [
                {
                    "businessEntity": {
                        "Person": {
                            "link": [
                                {
                                    "href": "http://localhost:8080/cmx/request/hm_icons/hierarchymanager/
Person/Person_Small.png?ors=localhost-mdm103-DS_UI2",
                                    "rel": "icon"
                                }
                            ],
                            "rowidObject": "2568254       ",
                            "creator": "admin",
                            "createDate": "2020-04-01T14:38:06.577-04:00",
                            "updatedBy": "admin",
                            "lastUpdateDate": "2020-04-01T14:38:06.577-04:00",
                            "consolidationInd": 4,
                            "lastRowidSystem": "SFA           ",
```

```
                        "hubStateInd": 1,
                        "effectivePeriod": {},
                        "label": "LN,FN",
                        "partyType": "Person",
                        "lastName": "LN",
                        "displayName": "FN LN",
                        "generationSuffixCd": "X",
                        "firstName": "FN"
                    }
                },
                "entityLabel": "LN,FN",
                "entityType": "Person",
                "depth": 0
            }
        ],
        "after": [
            {
                "businessEntity": {
                    "Person": {
                        "link": [
                            {
                                "href": "http://localhost:8080/cmx/request/hm_icons/hierarchymanager/
Person/Person_Small.png?ors=localhost-mdm103-DS_UI2",
                                "rel": "icon"
                            }
                        ],
                        "rowidObject": "2568253       ",
                        "creator": "admin",
                        "createDate": "2020-04-01T14:38:03.956-04:00",
                        "updatedBy": "admin",
                        "lastUpdateDate": "2020-04-01T14:38:03.956-04:00",
                        "consolidationInd": 4,
                        "lastRowidSystem": "SFA           ",
                        "hubStateInd": 1,
                        "label": "LN,FN",
                        "partyType": "Person",
                        "lastName": "LN",
                        "displayName": "FN LN",
                        "generationSuffixCd": "X",
                        "firstName": "FN"
                    }
                },
                "entityLabel": "Person",
                "entityType": "Person",
                "depth": 1,
                "object": {
                    "businessEntity": {
                        "Automobile": {
                            "rowidObject": "1450063       ",
                            "creator": "admin",
                            "createDate": "2020-04-01T14:38:09.630-04:00",
                            "updatedBy": "admin",
                            "lastUpdateDate": "2020-04-01T14:38:09.630-04:00",
                            "consolidationInd": 4,
                            "lastRowidSystem": "SFA           ",
                            "hubStateInd": 1,
                            "effectivePeriod": {},
                            "label": "Subaru",
                            "model": "MODEL",
                            "drivetrainCode": "4WD",
                            "doorsCode": "2DR",
                            "make": "Subaru"
                        }
                    },
                    "entityLabel": "Subaru",
                    "relationshipLabel": "PersonToAutomobile Relationship",
                    "relationship": {
                        "rowidObject": "1450083       ",
                        "creator": "ds1",
                        "createDate": "2020-04-01T14:40:14.921-04:00",
                        "updatedBy": "admin",
```

```
                        "lastUpdateDate": "2020-04-01T14:40:23.812-04:00",
                        "consolidationInd": 4,
                        "lastRowidSystem": "SYS0          ",
                        "interactionId": "68520005",
                        "hubStateInd": 1,
                        "label": "PersonToAutomobile Relationship",
                        "from": {
                            "rowidObject": "2568253       "
                        },
                        "to": {
                            "rowidObject": "1450063       "
                        }
                    },
                    "entityType": "Automobile",
                    "relationshipType": "PersonToAutomobile2"
                }
            },
            {
                "businessEntity": {
                    "Person": {
                        "link": [
                            {
                                "href": "http://localhost:8080/cmx/request/hm_icons/hierarchymanager/
Person/Person_Small.png?ors=localhost-mdm103-DS_UI2",
                                "rel": "icon"
                            }
                        ],
                        "rowidObject": "2568253       ",
                        "creator": "admin",
                        "createDate": "2020-04-01T14:38:03.956-04:00",
                        "updatedBy": "admin",
                        "lastUpdateDate": "2020-04-01T14:38:03.956-04:00",
                        "consolidationInd": 4,
                        "lastRowidSystem": "SFA           ",
                        "hubStateInd": 1,
                        "label": "LN,FN",
                        "partyType": "Person",
                        "lastName": "LN",
                        "displayName": "FN LN",
                        "generationSuffixCd": "X",
                        "firstName": "FN"
                    }
                },
                "entityLabel": "Person",
                "entityType": "Person",
                "depth": 1,
                "object": {
                    "businessEntity": {
                        "Person": {
                            "link": [
                                {
                                    "href": "http://localhost:8080/cmx/request/hm_icons/hierarchymanager/
Person/Person_Small.png?ors=localhost-mdm103-DS_UI2",
                                    "rel": "icon"
                                }
                            ],
                            "rowidObject": "2568254       ",
                            "creator": "admin",
                            "createDate": "2020-04-01T14:38:06.577-04:00",
                            "updatedBy": "admin",
                            "lastUpdateDate": "2020-04-01T14:38:06.577-04:00",
                            "consolidationInd": 4,
                            "lastRowidSystem": "SFA           ",
                            "hubStateInd": 1,
                            "effectivePeriod": {},
                            "label": "LN,FN",
                            "partyType": "Person",
                            "lastName": "LN",
                            "displayName": "FN LN",
                            "generationSuffixCd": "X",
                            "firstName": "FN"
```

```
                        }
                    },
                    "entityLabel": "LN,FN",
                    "relationshipLabel": "Person associate Person",
                    "relationship": {
                        "rowidObject": "1634729       ",
                        "creator": "ds1",
                        "createDate": "2020-04-01T14:40:14.870-04:00",
                        "updatedBy": "admin",
                        "lastUpdateDate": "2020-04-01T14:40:23.872-04:00",
                        "consolidationInd": 4,
                        "lastRowidSystem": "SYS0          ",
                        "interactionId": "68520005",
                        "hubStateInd": 1,
                        "label": "Person associate Person",
                        "rowidRelType": "103           ",
                        "rowidHierarchy": "1             ",
                        "note": "note DS",
                        "from": {
                            "rowidObject": "2568253       "
                        },
                        "to": {
                            "rowidObject": "2568254       "
                        }
                    },
                    "entityType": "Person",
                    "relationshipType": "PersonAssociatePerson2"
                }
            }
        ],
        "approved": [
            {
                "businessEntity": {
                    "Person": {
                        "link": [
                            {
                                "href": "http://localhost:8080/cmx/request/hm_icons/hierarchymanager/
Person/Person_Small.png?ors=localhost-mdm103-DS_UI2",
                                "rel": "icon"
                            }
                        ],
                        "rowidObject": "2568253       ",
                        "creator": "admin",
                        "createDate": "2020-04-01T14:38:03.956-04:00",
                        "updatedBy": "admin",
                        "lastUpdateDate": "2020-04-01T14:38:03.956-04:00",
                        "consolidationInd": 4,
                        "lastRowidSystem": "SFA           ",
                        "hubStateInd": 1,
                        "label": "LN,FN",
                        "partyType": "Person",
                        "lastName": "LN",
                        "displayName": "FN LN",
                        "generationSuffixCd": "X",
                        "firstName": "FN"
                    }
                },
                "entityLabel": "Person",
                "entityType": "Person",
                "depth": 1,
                "object": {
                    "businessEntity": {
                        "Automobile": {
                            "rowidObject": "1450063       ",
                            "creator": "admin",
                            "createDate": "2020-04-01T14:38:09.630-04:00",
                            "updatedBy": "admin",
                            "lastUpdateDate": "2020-04-01T14:38:09.630-04:00",
                            "consolidationInd": 4,
                            "lastRowidSystem": "SFA           ",
                            "hubStateInd": 1,
```

```
                            "effectivePeriod": {},
                            "label": "Subaru",
                            "model": "MODEL",
                            "drivetrainCode": "4WD",
                            "doorsCode": "2DR",
                            "make": "Subaru"
                        }
                    },
                    "entityLabel": "Subaru",
                    "relationshipLabel": "PersonToAutomobile Relationship",
                    "relationship": {
                        "rowidObject": "1450083       ",
                        "creator": "ds1",
                        "createDate": "2020-04-01T14:40:14.921-04:00",
                        "updatedBy": "ds1",
                        "lastUpdateDate": "2020-04-01T14:40:14.922-04:00",
                        "consolidationInd": 4,
                        "lastRowidSystem": "SYS0          ",
                        "interactionId": "68520005",
                        "hubStateInd": 0,
                        "label": "PersonToAutomobile Relationship",
                        "from": {
                            "rowidObject": "2568253       "
                        },
                        "to": {
                            "rowidObject": "1450063       "
                        }
                    },
                    "entityType": "Automobile",
                    "relationshipType": "PersonToAutomobile2"
                }
            },
            {
                "businessEntity": {
                    "Person": {
                        "link": [
                            {
                                "href": "http://localhost:8080/cmx/request/hm_icons/hierarchymanager/
Person/Person_Small.png?ors=localhost-mdm103-DS_UI2",
                                "rel": "icon"
                            }
                        ],
                        "rowidObject": "2568253       ",
                        "creator": "admin",
                        "createDate": "2020-04-01T14:38:03.956-04:00",
                        "updatedBy": "admin",
                        "lastUpdateDate": "2020-04-01T14:38:03.956-04:00",
                        "consolidationInd": 4,
                        "lastRowidSystem": "SFA           ",
                        "hubStateInd": 1,
                        "label": "LN,FN",
                        "partyType": "Person",
                        "lastName": "LN",
                        "displayName": "FN LN",
                        "generationSuffixCd": "X",
                        "firstName": "FN"
                    }
                },
                "entityLabel": "Person",
                "entityType": "Person",
                "depth": 1,
                "object": {
                    "businessEntity": {
                        "Person": {
                            "link": [
                                {
                                    "href": "http://localhost:8080/cmx/request/hm_icons/hierarchymanager/
Person/Person_Small.png?ors=localhost-mdm103-DS_UI2",
                                    "rel": "icon"
                                }
                            ],
```

```
                    "rowidObject": "2568254       ",
                    "creator": "admin",
                    "createDate": "2020-04-01T14:38:06.577-04:00",
                    "updatedBy": "admin",
                    "lastUpdateDate": "2020-04-01T14:38:06.577-04:00",
                    "consolidationInd": 4,
                    "lastRowidSystem": "SFA           ",
                    "hubStateInd": 1,
                    "effectivePeriod": {},
                    "label": "LN,FN",
                    "partyType": "Person",
                    "lastName": "LN",
                    "displayName": "FN LN",
                    "generationSuffixCd": "X",
                    "firstName": "FN"
                }
            },
            "entityLabel": "LN,FN",
            "relationshipLabel": "Person associate Person",
            "relationship": {
                "rowidObject": "1634729       ",
                "creator": "ds1",
                "createDate": "2020-04-01T14:40:14.870-04:00",
                "updatedBy": "ds1",
                "lastUpdateDate": "2020-04-01T14:40:14.871-04:00",
                "consolidationInd": 4,
                "lastRowidSystem": "SYS0          ",
                "interactionId": "68520005",
                "hubStateInd": 0,
                "label": "Person associate Person",
                "rowidRelType": "103           ",
                "rowidHierarchy": "1             ",
                "note": "note DS",
                "from": {
                    "rowidObject": "2568253       "
                },
                "to": {
                    "rowidObject": "2568254       "
                }
            },
            "entityType": "Person",
            "relationshipType": "PersonAssociatePerson2"
        }
    }
  ]
 }
]
}
```


一括却下

一括却下 REST API は、ネットワークまたは階層への変更を拒否します。

この API は POST メソッドを使用します。

要求 URL

[一括却下] URL の形式は次のとおりです。

`http://<host>:<port>/<context>/<database ID>/reject`

[一括却下] URL に対して、次の HTTP GET 要求を行います。

`POST http://<host>:<port>/<context>/<database ID>/reject`

サンプル API 要求

次のサンプル要求は、変更を却下します。

```
POST http://localhost:8080/cmx/bulk/localhost-orcl-DS_UI1/reject
{
  "interactionId":"67080007",
  "rejectInteractionId":"67080008",
  "objects":[
    {
      "type":"Hierarchy",
      "name":"HPerson"
    }
  ]
}
```

サンプル API 応答

次のサンプルは、変更を正常に却下した後の応答を示しています。

```
{
    "interactionId": "68520003",
    "rejectInteractionId": "68520004",
    "type": "hierarchy",
    "name": "HPerson",
    "objects": [
        {
            "before": [
                {
                    "businessEntity": {
                        "Person": {
                            "link": [
                                {
                                    "href": "http://localhost:8080/cmx/request/hm_icons/hierarchymanager/
Person/Person_Small.png?ors=localhost-mdm103-DS_UI2",
                                    "rel": "icon"
                                }
                            ],
                            "rowidObject": "2568254       ",
                            "creator": "admin",
                            "createDate": "2020-04-01T14:38:06.577-04:00",
                            "updatedBy": "admin",
                            "lastUpdateDate": "2020-04-01T14:38:06.577-04:00",
                            "consolidationInd": 4,
                            "lastRowidSystem": "SFA           ",
                            "hubStateInd": 1,
                            "effectivePeriod": {},
                            "label": "LN,FN",
                            "partyType": "Person",
                            "lastName": "LN",
                            "displayName": "FN LN",
                            "generationSuffixCd": "X",
                            "firstName": "FN"
                        }
                    },
                    "entityLabel": "LN,FN",
                    "entityType": "Person",
                    "depth": 0
                }
            ],
            "after": [
                {
                    "businessEntity": {
                        "Person": {
                            "link": [
                                {
                                    "href": "http://localhost:8080/cmx/request/hm_icons/hierarchymanager/
Person/Person_Small.png?ors=localhost-mdm103-DS_UI2",
                                    "rel": "icon"
                                }
```

```
                        ],
                        "rowidObject": "2568254       ",
                        "creator": "admin",
                        "createDate": "2020-04-01T14:38:06.577-04:00",
                        "updatedBy": "admin",
                        "lastUpdateDate": "2020-04-01T14:38:06.577-04:00",
                        "consolidationInd": 4,
                        "lastRowidSystem": "SFA           ",
                        "hubStateInd": 1,
                        "effectivePeriod": {},
                        "label": "LN,FN",
                        "partyType": "Person",
                        "lastName": "LN",
                        "displayName": "FN LN",
                        "generationSuffixCd": "X",
                        "firstName": "FN"
                    }
                },
                "entityLabel": "LN,FN",
                "entityType": "Person",
                "depth": 0
            }
        ],
        "approved": [
            {
                "businessEntity": {
                    "Person": {
                        "link": [
                            {
                                "href": "/request/hm_icons/hierarchymanager/Person/Person_Small.png?
ors={ors}",
                                "rel": "icon"
                            }
                        ],
                        "rowidObject": "2568253       ",
                        "creator": "admin",
                        "createDate": "2020-04-01T14:38:03.956-04:00",
                        "updatedBy": "admin",
                        "lastUpdateDate": "2020-04-01T14:38:03.956-04:00",
                        "consolidationInd": 4,
                        "lastRowidSystem": "SFA           ",
                        "hubStateInd": 1,
                        "label": "LN,FN",
                        "partyType": "Person",
                        "lastName": "LN",
                        "displayName": "FN LN",
                        "generationSuffixCd": "X",
                        "firstName": "FN"
                    }
                },
                "entityLabel": "Person",
                "entityType": "Person",
                "depth": 1,
                "object": {
                    "businessEntity": {
                        "Automobile": {
                            "rowidObject": "1450063       ",
                            "creator": "admin",
                            "createDate": "2020-04-01T14:38:09.630-04:00",
                            "updatedBy": "admin",
                            "lastUpdateDate": "2020-04-01T14:38:09.630-04:00",
                            "consolidationInd": 4,
                            "lastRowidSystem": "SFA           ",
                            "hubStateInd": 1,
                            "effectivePeriod": {},
                            "label": "Subaru",
                            "model": "MODEL",
                            "drivetrainCode": "4WD",
                            "doorsCode": "2DR",
                            "make": "Subaru"
                        }
```

```
                        },
                        "entityLabel": "Subaru",
                        "relationshipLabel": "PersonToAutomobile Relationship",
                        "relationship": {
                            "rowidObject": "1450082       ",
                            "creator": "ds1",
                            "createDate": "2020-04-01T14:38:13.916-04:00",
                            "updatedBy": "ds1",
                            "lastUpdateDate": "2020-04-01T14:38:13.916-04:00",
                            "consolidationInd": 4,
                            "lastRowidSystem": "SYS0          ",
                            "interactionId": "68520003",
                            "hubStateInd": 0,
                            "label": "PersonToAutomobile Relationship",
                            "from": {
                                "rowidObject": "2568253       "
                            },
                            "to": {
                                "rowidObject": "1450063       "
                            }
                        },
                        "entityType": "Automobile",
                        "relationshipType": "PersonToAutomobile2"
                    }
                },
                {
                    "businessEntity": {
                        "Person": {
                            "link": [
                                {
                                    "href": "/request/hm_icons/hierarchymanager/Person/Person_Small.png?
ors={ors}",
                                    "rel": "icon"
                                }
                            ],
                            "rowidObject": "2568253       ",
                            "creator": "admin",
                            "createDate": "2020-04-01T14:38:03.956-04:00",
                            "updatedBy": "admin",
                            "lastUpdateDate": "2020-04-01T14:38:03.956-04:00",
                            "consolidationInd": 4,
                            "lastRowidSystem": "SFA           ",
                            "hubStateInd": 1,
                            "label": "LN,FN",
                            "partyType": "Person",
                            "lastName": "LN",
                            "displayName": "FN LN",
                            "generationSuffixCd": "X",
                            "firstName": "FN"
                        }
                    },
                    "entityLabel": "Person",
                    "entityType": "Person",
                    "depth": 1,
                    "object": {
                        "businessEntity": {
                            "Person": {
                                "link": [
                                    {
                                        "href": "/request/hm_icons/hierarchymanager/Person/Person_Small.png?
ors={ors}",
                                        "rel": "icon"
                                    }
                                ],
                                "rowidObject": "2568254       ",
                                "creator": "admin",
                                "createDate": "2020-04-01T14:38:06.577-04:00",
                                "updatedBy": "admin",
                                "lastUpdateDate": "2020-04-01T14:38:06.577-04:00",
                                "consolidationInd": 4,
                                "lastRowidSystem": "SFA           ",
```

```
                        "hubStateInd": 1,
                        "effectivePeriod": {},
                        "label": "LN,FN",
                        "partyType": "Person",
                        "lastName": "LN",
                        "displayName": "FN LN",
                        "generationSuffixCd": "X",
                        "firstName": "FN"
                    }
                },
                "entityLabel": "LN,FN",
                "relationshipLabel": "Person associate Person",
                "relationship": {
                    "rowidObject": "1634728       ",
                    "creator": "ds1",
                    "createDate": "2020-04-01T14:38:13.871-04:00",
                    "updatedBy": "ds1",
                    "lastUpdateDate": "2020-04-01T14:38:13.871-04:00",
                    "consolidationInd": 4,
                    "lastRowidSystem": "SYS0          ",
                    "interactionId": "68520003",
                    "hubStateInd": 0,
                    "label": "Person associate Person",
                    "rowidRelType": "103           ",
                    "rowidHierarchy": "1             ",
                    "note": "note DS",
                    "from": {
                        "rowidObject": "2568253       "
                    },
                    "to": {
                        "rowidObject": "2568254       "
                    }
                },
                "entityType": "Person",
                "relationshipType": "PersonAssociatePerson2"
            }
        }
    ]
  }
]
}
```


一致の開始

一致の開始 REST API は、ビジネスエンティティの一致ジョブを開始します。一致の開始 API を呼び出す前に、ビジネスエンティティを含むファイルをインポートする必要があります。

この API は POST メソッドを使用します。

要求 URL

[一致の開始] URL の形式は次のとおりです。

```
http://<host>:<port>/adhocmatch/<orsId>/job
```

[一致の開始] URL に対して、次の HTTP POST 要求を行います。

```
POST http://<host>:<port>/adhocmatch/<orsId>/job
```

パラメータ

ビジネスエンティティー一致ジョブを開始するためのパラメータを指定します。

次の表に、これらのパラメータについて説明します。

パラメータ	タイプ	説明
orsId	パス	一致するビジネスエンティティを取得するためのオペレーショナルリファレンスストアデータベース ID。
fileId	ボディ	オプション。インポートするファイルの ID。
mappingId	ボディ	オプション。インポートするファイルのマッピング ID。
matchRuleSet	ボディ	オプション。ビジネスエンティティー一致ジョブを適用するための一致ルールセットを指定します。

サンプル API 要求

次のサンプル要求は、照合の実行後にインポートされたビジネスエンティティを返します。

```
POST /cmx/jobcontrol/localhost-orcl-MDM_SAMPLE/execute/match HTTP/1.1
Content-Type: application/json
Host: localhost:8080
Content-Length: 64

{
  "tableName" : "C_PRODUCT",
  "matchRuleSetName" : "IDL"
}
```

サンプル API 応答

API は、ビジネスエンティティを照合してファイルを正常にインポートした後、200 OK 応答コードを返します。応答本文は空です。

一致するレコードの読み取り

［一致するレコードの読み取り］REST API は、指定されたルートレコードに一致するレコードを返します。レコードのリストを確認し、どのレコードを元のルートレコードとマージできるかを判断できます。レコードのマージには、［レコードのマージ］API を使用できます。

この API は GET メソッドを使用します。

要求 URL

［一致するレコードの読み取り］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<rowId>?action=matched
```

［一致するレコードの読み取り］URL に対して、次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/<context>/<database ID>/<business entity>/<rowId>?action=matched
```

応答本文

応答本文には、指定されたレコードに一致するレコードの数、一致レコードの詳細、および検索トークンが含まれます。検索トークンは、後続の一致結果ページの取得に使用します。

サンプル API 要求

次のサンプル要求は、ビジネスエンティティから特定のレコードに一致するレコードを検索します。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/1038245?action=matched
```

サンプル API 応答

次のサンプル応答では、指定されたレコードに一致するレコードの詳細が示されています。

```
{
    "firstRecord": 1,
    "recordCount": 1,
    "pageSize": 10,
    "searchToken": "SVR1.AU5HE",
    "matchedEntity": [
        {
            "businessEntity": {
                "Person": {
                    "link": [
                        {
                            "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/1038246",
                            "rel": "self"
                        }
                    ],
                    "rowidObject": "1038246           ",
                    "creator": "admin",
                    "createDate": "2008-08-12T02:15:02-07:00",
                    "updatedBy": "Admin",
                    "lastUpdateDate": "2008-08-12T02:59:17-07:00",
                    "consolidationInd": "1",
                    "lastRowidSystem": "SFA           ",
                    "dirtyIndicator": "0",
                    "hubStateInd": "1",
                    "partyType": "Person",
                    "lastName": "BATES",
                    "firstName": "DAISY",
                    "displayName": "DAISY BATES"
                }
            },
            "matchRule": "PUT"
        }
    ]
}
```

一致するレコードの更新

[一致するレコードの更新] REST API は、マッチテーブルのレコードを作成または更新します。一致テーブルには、ビジネスエンティティに対して一致プロセスを実行した後のビジネスエンティティ内の一致するレコードのペアが含まれます。この API は、指定したレコードとのマージに適したレコードを追加するために使用します。

この API は PUT メソッドを使用します。

要求 URL

[一致するレコードの更新] URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<rowId>?action=matched
```

[一致するレコードの更新] URL に対して、次の HTTP PUT 要求を行います。

```
PUT http://<host>:<port>/<context>/<database ID>/<business entity>/<rowId>?action=matched
```

要求と一緒に送信するデータのメディアタイプを指定するには、Content-Type ヘッダーを追加します。

```
PUT http://<host>:<port>/<context>/<database ID>/<business entity>/<rowId>?action=matched
Content-Type: application/<json/xml>
```

要求本文

指定するレコードに一致するレコードのリストを要求本文で送信します。行 ID、またはソースシステムとソースキーを使用してレコードを指定します。

サンプル API 要求

次のサンプルは、マッチテーブルにレコードを追加します。

```
PUT http://localhost:8080/cmx/cs/localhost-ORCL-DS_UI1/Person/1038245?action=matched
```

```
{
    keys: [
       {
          rowid: "1038246"
       }
   ]
}
```

サンプル API 応答

この API は、マッチテーブルに正常にレコードが作成された場合に 200 OK 応答を返します。応答本文は空です。

一致レコードの削除

[一致レコードの削除] REST API は、照合テーブルのルートレコードに関連付けられている一致レコードを削除します。

この API は DELETE メソッドを使用します。

要求 URL

[一致レコードの削除] URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<rowId>?action=matched
```

[一致レコードの削除] URL に対して、次の HTTP DELETE 要求を行います。

```
DELETE http://<host>:<port>/<context>/<database ID>/<business entity>/<rowId>?action=matched
```

要求と一緒に送信するデータのメディアタイプを指定するには、Content-Type ヘッダーを追加します。

```
DELETE http://<host>:<port>/<context>/<database ID>/<business entity>/<rowId>?action=matched
Content-Type: application/<json/xml>
```

要求本文

マッチテーブルから削除するレコードのリストを要求本文で送信します。

サンプル API 要求

次のサンプル要求は、指定されたルートレコードに一致するレコードをマッチテーブルから削除します。

```
DELETE http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/1038245?action=matched
```

```
{
    keys: [
```

```
      {
         rowid: "1038246"
      }
   ]
}
```

サンプル API 応答

この API は、マッチテーブルからレコードが正常に削除された場合に 200 OK 応答を返します。応答本文は空です。

CSV 形式での一致データの取得

CSV 形式での一致データの取得 REST API は、システムがソースシステムからのレコードとターゲットフィールドを照合した後で、一致および不一致のビジネスエンティティを CSV 形式で取得します。

この API は GET メソッドを使用します。

要求 URL

[一致データの取得] URL の形式は次のとおりです。

```
http://<host>:<port>/adhocmatch/<orsId>/result/{jobGroupControlId}{?withMatches,withoutMatches}
```

[一致データ結果の取得] URL に対して、次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/adhocmatch/<orsId>/result/{jobGroupControlId}{?withMatches,withoutMatches}
```

パラメータ

一致および不一致のビジネスエンティティを CSV 形式で返すためのパラメータを指定します。

次の表に、これらのパラメータについて説明します。

パラメータ	タイプ	説明
jobGroupId	パス	バッチジョブのグループ ID。
orsId	パス	ビジネスエンティティレコードを取得するオペレーショナルリファレンスストアデータベース ID。
withMatches	クエリ	オプション。使用する値は次のとおりです。 True: 既存のデータに一致するレコードを返します。
withoutMatches	クエリ	オプション。使用する値は次のとおりです。 False: 既存のデータに一致しないレコードを返します。

サンプル API 要求

次のサンプル要求は、一致するビジネスエンティティと一致しないビジネスエンティティを CSV 形式で返します。

```
GET /cmx/adhocmatch/{ors}/result/{jobId}?withMatches=true&withoutMatches=true
Accept: application/csv
```

```
{
    item: [
```

```
      {
        businessEntity: {
            Person: {
                firstName: "John"
            }
        },
        matchedEntity: {
            {
                businessEntity: {Person: {rowidObject: 123}},
            }
            matchRule: "MRS|1",
            matchScore: 0.9
        }
      },
      {
        businessEntity: {
            Person: {
                firstName: "Alex"
            }
        }
      }
   ]
}
```

サンプル API 応答

API は、200 OK 応答コードと、一致結果を含む CSV ファイルを返します。

JSON 形式での一致データの取得

JSON 形式での一致データの取得 REST API は、システムがソースシステムからのレコードとターゲットフィールドを照合した後で、一致および不一致のビジネスエンティティを CSV 形式で取得します。

この API は GET メソッドを使用します。

要求 URL

[一致結果] URL の形式は次のとおりです。

```
http://<host>:<port>/adhocmatch/<orsId>/result/{jobGroupControlId}{?withMatches,withoutMatches}
```

[一致結果] URL に対して、次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/adhocmatch/<orsId>/result/{jobGroupControlId}{?withMatches,withoutMatches}
```

パラメータ

一致および不一致のビジネスエンティティを CSV 形式で返すためのパラメータを指定します。

次の表に、これらのパラメータについて説明します。

パラメータ	タイプ	説明
jobGroupId	パス	バッチジョブのグループ ID。
orsId	パス	ビジネスエンティティレコードを取得するオペレーショナルリファレンスストアデータベース ID。

パラメータ	タイプ	説明
withMatches	クエリ	オプション。使用する値は次のとおりです。 True: 既存のデータに一致するレコードを返します。
withoutMatches	クエリ	オプション。使用する値は次のとおりです。 False: 既存のデータに一致しないレコードを返します。

サンプル API 要求

次のサンプル要求は、システムが照合を実行した後に、一致するビジネスエンティティと一致しないビジネスエンティティのリストを返します。

```
GET /cmx/adhocmatch/{ors}/result/{jobId}?withMatches=true&withoutMatches=true
Accept: application/json
```

```
{
    item: [
      {
        businessEntity: {
            Person: {
                firstName: "John"
            }
        },
        matchedEntity: {
            {
                businessEntity: {Person: {rowidObject: 123}},
            }
            matchRule: "MRS|1",
            matchScore: 0.9
        }
      },
      {
        businessEntity: {
            Person: {
                firstName: "Alex"
            }
        }
      }
   ]
}
```

サンプル API 応答

API は、200 OK 応答コードと、一致結果を含む JSON ファイルを返します。

一致ファイルのインポート

一致ファイルのインポート REST API は、ファイルをインポートする前に、新しいビジネスエンティティを既存のビジネスエンティティと照合します。インポート操作が完了した後、一致したビジネスエンティティは新しいデータで更新されます。

重複ビジネスエンティティ、一意のビジネスエンティティまたは重複および一意のビジネスエンティティをインポートできます。

この API は POST メソッドを使用します。

要求 URL

［一致したファイルのインポート］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<orsId>/<aftermatch>
```

［一致したファイルのインポート］URL に対して、次の HTTP POST 要求を行います。

```
POST http://<host>:<port>/<context>/<orsId>/<aftermatch>
```

パラメータ

ファイルをインポートする前に、新しいビジネスエンティティを既存のビジネスと一致させるためのパラメータを指定します。

次の表に、これらのパラメータについて説明します。

パラメータ	タイプ	説明
orsId	パス	既存のレコードの一致後、ファイルをインポートするオペレーショナルリファレンスストアデータベース ID。
filter	クエリ	オプション。次のフィルタ値を使用します: withMatches: レコードが既存のレコードと一致する場合は true を入力します。 withoutMatches: レコードが既存のレコードと一致しない場合は false を入力します。
jobGroupControlId	クエリ	オプション。一致ジョブグループ ID。
systemName	クエリ	オプション。ソースシステム名。

サンプル API 要求

次のサンプル要求は、ファイルをインポートする前に、新しいビジネスエンティティを既存のビジネスエンティティと照合します。

```
POST /cmx/beimport/{ors}/aftermatch
{
    jobGroupControlId: "...",
    systemName: "SFA",
    filter: {
      withMatches: true,
      withoutMatches: false
    }
}
```

サンプル API 応答

API は、新しいビジネスエンティティを既存のビジネスエンティティと正常に照合し、ファイルをインポートした後、200 OK 応答コードを返します。応答本文は空です。

ソースシステムメタデータの取得

ソースシステムメタデータの取得 REST API は、ソースシステムのメタデータを返します。

この API は GET メソッドを使用します。

要求 URL

［ソースシステムメタデータの取得］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<orsId>/sourceSystem
```

［ソースシステムメタデータの取得］URL に対して次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/<context>/<orsId>/sourceSystem
```

パラメータ

応答で返すソースシステムを指定します。

次の表に、要求本文内のパラメータを示します。

パラメータ	タイプ	説明
orsId	パス	ソースシステムを取得するためのオペレーショナルリファレンスストア ID。
entityId	パス	ビジネスエンティティの ID。
entityName	パス	ビジネスエンティティの名前。
action	クエリ	API で実行するアクション。デフォルト値は次のとおりです。 `getSourceSystems`
aggregate	クエリ	オプション。複数の XREF 間の集計データ。使用する値は次のとおりです。 - RECORD - NODE - ENTITY
children	クエリ	オプション。子ノード名のリスト。指定した場合、応答には子ノード名が含まれます。
compact	クエリ	オプション。圧縮結果を返すことを指定します。
contentMetadata	クエリ	オプション。履歴イベントのリストのメタデータ。使用する値は次のとおりです。 XREF PENDING_XREF DELETED_XREF HISTORY MATCH BVT TRUST
contributorsOnly	クエリ	オプション。すべての XREF または最善データ（BVT）に貢献する XREF を数えます。
depth	クエリ	オプション。返す子レベルの数。
describe	クエリ	オプション。ソースシステムの説明を追加します。

パラメータ	タイプ	説明
fields	クエリ	オプション。ビジネスエンティティフィールドのカンマ区切りリスト。指定した場合、応答にはリストされたフィールドのみが含まれます。
filter	クエリ	オプション。フィルタ式。
firstRecord	クエリ	オプション。結果の最初の行を指定します。
order	クエリ	オプション。フィールド名のカンマ区切りリスト。オプションでプレフィックス+または-を指定できます。プレフィックス+は結果を昇順でソートし、プレフィックス-は結果を降順でソートすることを指定します。デフォルトは+です。複数のパラメータを指定した場合、結果セットは、指定した最初のパラメータでまずソートされてから、次のパラメータでソートされます。
recordStates	クエリ	オプション。履歴イベントのリストで返されるレコードの状態。カンマ区切りのリストを指定します。 以下の値を使用できます。 - ACTIVE - PENDING - DELETED
recordstoReturn	クエリ	オプション。返す行の数を指定します。
recordsTotal	クエリ	オプション。true に設定すると、結果内のレコードの数が返されます。デフォルトは false です。

サンプル API 要求

次のサンプル要求は、使用可能なソースシステムメタデータを返します。

```
GET /cmx/metadata/localhost-orcl-MDM_SAMPLE/sourceSystem HTTP/1.1
Host: localhost:8080
```

サンプル API 応答

次のサンプル応答は、使用可能なソースシステムメタデータを返します。

```
HTTP/1.1 200 OK
Server: JBoss-EAP/7
X-Powered-By: Undertow/1
Content-Type: application/json;charset=UTF-8
Content-Length: 328

{
  "totalCount" : 7,
  "firstRecord" : 1,
  "pageSize" : 7,
  "sourceSystems" : [ {
    "name" : "Product"
  }, {
    "name" : "Informatica Data Director"
  }, {
    "name" : "Legacy"
  }, {
    "name" : "SFA"
  }, {
    "name" : "Lookups"
```

```
  },{
    "name" : "OrgDataInc"
  },{
    "name" : "Admin"
  } ]
}
```

レコード履歴イベントの取得

［レコード履歴イベントの取得］REST API は、レコードに関連付けられている履歴イベントまたは履歴イベントグループのリストを返します。要求本文でレコード ID を送信します。

この API は、GET メソッドを使用して、履歴イベントグループごとに次のデータを返します。

- グループの開始日と終了日
- グループのイベント数

この API は、履歴イベントごとに次のデータを返します。

- 履歴イベント ID
- 変更日
- 変更したユーザー
- 変更の影響を受ける履歴テーブルのリスト
- 変更の影響を受けるレコードノードのリスト

要求 URL

［レコード履歴イベントの取得］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<rowId>?action=listHistoryEvents
```

［レコード履歴イベントの取得］URL に対して、次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/<context>/<database ID>/<business entity>/<rowId>?action=listHistoryEvents
```

クエリパラメータ

レコードの ID は必須パラメータです。この API は、レコード ID を使用して、関連するすべての履歴イベントを検索します。

次の表にクエリパラメータを示します。

パラメータ	説明
startDate と endDate	オプション。データの取得対象となる日付範囲。日付範囲を指定すると、応答にはその範囲内のイベントのみが含まれます。
granularity	オプション。履歴イベントをグループ化する詳細のレベル。指定すると、応答で履歴イベントがグループ化されます。指定しないと、応答で履歴イベントはグループ化されません。 以下の値を使用します。 - YEAR - QUARTER - MONTH - WEEK - DAY - HOUR - MINUTE - AUTO
recordStates	オプション。履歴イベントのリストで返されるレコードの状態。カンマ区切りのリストを指定します。 以下の値を使用できます。 - ACTIVE - PENDING - DELETED
contentMetadata	オプション。履歴イベントのリストのメタデータ。カンマ区切りのリストを指定します。 以下の値を使用できます。 - XREF - PENDING_XREF - DELETED_XREF - HISTORY - MATCH - BVT - TRUST
children	オプション。子ノード名のカンマ区切りリスト。指定した場合、応答には子ノード名が含まれます。
order	オプション。フィールド名のカンマ区切りリスト。オプションでプレフィックス+または-を指定できます。プレフィックス+は結果を昇順でソートし、プレフィックス-は結果を降順でソートすることを指定します。デフォルトは+です。複数のパラメータを指定した場合、結果セットは、指定した最初のパラメータでまずソートされてから、次のパラメータでソートされます。
fields	オプション。ビジネスエンティティフィールドのカンマ区切りリスト。指定した場合、応答にはリストされたフィールドのみが含まれます。
filter	オプション。フィルタ式。
depth	オプション。返す子レベルの数。

パラメータ	説明
recordsToReturn	オプション。返す行の数を指定します。
searchToken	オプション。前の要求で返された検索トークンを指定します。
returnTotal	オプション。true に設定すると、結果内のレコードの数が返ります。デフォルトは false です。
firstRecord	オプション。結果の最初の行を指定します。
changeType	オプション。結果で返される変更のタイプを指定します。カンマ区切りのリストを指定します。 以下の値を使用できます。 - BO - XREF - BVT - MERGE - MERGE_AS_SOURCE - MERGE_AS_TARGET - UNMERGE_AS_SOURCE - UNMERGE_AS_TARGET

関連項目：

- 「UTC での日付と時刻の形式」 (ページ 34)

サンプル API 要求

次のサンプル要求は、あるレコードの 2000 年 1 月 1 日以降のすべてのマージを年別に分類して返します:

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/123?
action=listHistoryEvents&startDate=2010-01-01&granularity=YEAR&depth=2&changeType=MERGE
```

サンプル API 応答

次のサンプル応答は、指定したレコードの 2000 年 1 月 1 日以降のマージをリストしています:

```
{
    firstRecord: 1,
    recordCount: 2
    item: [
        {
            link: [ // you can use links directly to get event list
                {rel: "events", href: "/Person/123?
action=listHistoryEvents&startDate=2000-01-01&endDate=2001-01-01&depth=2&changeType=MERGE"}
            ],
            startDate: "2000-01-01",
            endDate: "2001-01-01",
            eventCount: 123
        },
        // no events in 2001, 2002, ... 2009
        {
            link: [
                {rel: "events", href: "/Person/123?
action=listHistoryEvents&startDate=2010-01-01&endDate=2011-01-01&depth=2&changeType=MERGE"}
            ],
            startDate: "2010-01-01",
            endDate: "2011-01-01",
            eventCount: 23
        }
        // no events in 2011, ..., 2016
```

```
    ]
}
```

イベント詳細の取得

［イベント詳細の取得］REST API は、レコードに関連付けられている特定の履歴イベントの詳細を返します。この API は、行われた変更のタイプ、変更前後の値などの詳細を返します。レコード ID と履歴イベント ID を要求本文で送信します。

この API は GET メソッドを使用します。

要求 URL

［イベント詳細の取得］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<rowId>?action=getHistoryEventDetails
```

［イベント詳細の取得］URL に対して、次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/<context>/<database ID>/<business entity>/<rowId>?action=getHistoryEventDetails
```

クエリパラメータ

始める前に、［レコード履歴イベントの取得］API を使用して、レコードに関連付けられている履歴イベントをリストします。返された結果から、履歴イベント ID とレコード ID をクエリパラメータとして使用します。

次の表にクエリパラメータを示します。

パラメータ	説明
eventID	必須。［レコード履歴イベントの取得］API は、履歴イベントのイベント ID を返します。
recordStates	オプション。履歴イベントのリストで返されるレコードの状態。カンマ区切りのリストを指定します。 以下の値を使用できます。 - ACTIVE - PENDING - DELETED
contentMetadata	オプション。履歴イベントのリストのメタデータ。カンマ区切りのリストを指定します。 以下の値を使用できます。 - XREF - PENDING_XREF - DELETED_XREF - HISTORY - MATCH - BVT - TRUST
children	オプション。子ノード名のカンマ区切りリスト。指定した場合、応答には子ノード名が含まれます。

パラメータ	説明
order	オプション。フィールド名のカンマ区切りリスト。オプションでプレフィックス+または-を指定できます。プレフィックス+は結果を昇順でソートし、プレフィックス-は結果を降順でソートすることを指定します。デフォルトは+です。複数のパラメータを指定した場合、結果セットは、指定した最初のパラメータでまずソートされてから、次のパラメータでソートされます。
fields	オプション。ビジネスエンティティフィールドのカンマ区切りリスト。指定した場合、応答にはリストされたフィールドのみが含まれます。
filter	オプション。フィルタ式。
depth	オプション。返す子レベルの数。
recordsToReturn	オプション。返す行の数を指定します。
searchToken	オプション。前の要求で返された検索トークンを指定します。
returnTotal	オプション。true に設定すると、結果内のレコードの数が返されます。デフォルトは false です。
firstRecord	オプション。結果の最初の行を指定します。

サンプル API 要求

次のサンプル要求は、履歴イベントの情報を返します。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/123?
action=getHistoryEventDetails&eventId=2016-07-14T02%3A01%3A24.529%2B0000
```

サンプル API 応答

次のサンプル応答では、指定したイベントの詳細が示されています。

```
{
  "eventId": "2016-07-14T02:01:24.529+0000",
  "eventDate": "2016-07-14T02:01:24.529Z",
  "user": "admin",
  "changeType": [
    "BVT",
    "BO",
    "UNMERGE_AS_TARGET"
  ],
  "businessEntity": {
    "Person": {
      "rowidObject": "438243        ",
      "creator": "datasteward1",
      "createDate": "2016-07-08T20:42:47.402Z",
      "updatedBy": "admin",
      "lastUpdateDate": "2016-07-14T01:42:50.841Z",
      "consolidationInd": 1,
      "lastRowidSystem": "SYSO          ",
      "hubStateInd": 1,
      "label": "BE,AC",
      "partyType": "Person",
      "lastName": "BE",
      "displayName": "AC BE",
      "firstName": "AC"
    }
```

```
  }
}
```

リストレポート

レポートのリスト REST API は、登録されたレポートとその設定のリストを返します。

この API は GET メソッドを使用します。

要求 URL

［レポートの一覧表示］REST URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/list
```

この URL に対して次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/<context>/<database ID>/list
```

クエリパラメータ

show パラメータを使用して、すべてのレポートを一覧表示するか、ルートレポートのみを一覧表示するかを指定できます。

ルートレポートおよびドリルダウンレポートを含むすべてのレポートを返すには、次のクエリパラメータを使用します。

```
list?show=all
```

ルートレポートのみを返すには、次のクエリパラメータを使用します。

```
list?show=root
```

注: show パラメータを指定しない場合、API はルートレポートを返します。

応答本文

応答本文には、各レポートの詳細を含むレポートのリストが含まれます。

次の表に、応答本文内のレポート設定パラメータを示します。

パラメータ	説明
ROWID_RPT_CONFIG	レポートの ID。
DIMENSION_NAME_1	レポート内のデータの次元の名前。次元は例えば、ユーザー名、レビュータスクタイプ、ビジネスエンティティタイプなどです。
DIMENSION_NAME_2	オプション。レポート内のデータの次元の名前。次元は例えば、ユーザー名、レビュータスクタイプ、ビジネスエンティティタイプなどです。
TIMEPERIOD_NAME	オプション。データが適用される期間の名前。例えば、「月」と指定できます。
RPT_NAME	レポートの名前。
METRIC_ NAME	レポートによって収集されたデータのタイプのラベル。デフォルトでは、このパラメータに指定された値は、グラフの y 軸の名前として表示されます。例えば、ユーザーがクローズしたタスクに関する情報を収集するレポートがあるとします。メトリック名に「クローズされたタスクの数」を使用できます。

パラメータ	説明
RPT_DESC	レポートの説明。
RPT_TYPE	レポートがドリルダウンレポートかどうかを示します。値が null の場合、レポートはルートレポートです。値が null 以外の場合、レポートはドリルダウンレポートです。

サンプル API 要求

次のサンプル要求は、ルートレポートとドリルダウンレポートを含むすべての登録済みレポートを返します。

```
GET http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/list?show=all
```

サンプル API 応答

次のサンプル応答は、登録済みレポートのリストを示しています。

```
{
  "ROWID_RPT_CONFIG": "7 ",
  "DIMENSION_NAME_1": "Country",
  "DIMENSION_NAME_2": "Users",
  "TIMEPERIOD_NAME": "Month",
  "RPT_NAME": "Top 10 Countries",
  "METRIC_NAME": "Number Of Customers",
  "RPT_DESC": "Top 10 Countries",
  "RPT_TYPE": "null"
},
{
  "metadata": {
  "ROWID_RPT_CONFIG": "12 ",
  "DIMENSION_NAME_1": "City code",
  "DIMENSION_NAME_2": "null",
  "TIMEPERIOD_NAME": "null",
  "RPT_NAME": "Customer by City",
  "METRIC_NAME": "Customer by City",
  "RPT_DESC": "Customer by City",
  "RPT_TYPE": "null"
}
```

レポート設定およびデータの取得

レポート設定およびデータの取得 REST API は、レポート設定およびデータを返します。

この API は GET メソッドを使用します。

要求 URL

［レポート設定とデータの取得］REST URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/data/<report ID>
```

この URL に対して次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/<context>/<database ID>/data/<reportId>
```

クエリパラメータ

レポート ID は必須 URL パラメータです。レポート ID を使用して、設定およびデータが必要なレポートを指定します。

応答本文

応答には、レポート設定とデータが含まれます。

次の表に、応答本文内のパラメータを示します。

パラメータ	説明
metadata	レポート設定が含まれます。
ROWID_RPT_CONFIG	レポートの ID。
DIMENSION_NAME_1	レポート内のデータの次元の名前。次元は例えば、ユーザー名、レビュータスクタイプ、ビジネスエンティティタイプなどです。
DIMENSION_NAME_2	オプション。レポート内のデータの次元の名前。次元は例えば、ユーザー名、レビュータスクタイプ、ビジネスエンティティタイプなどです。
TIMEPERIOD_NAME	オプション。データが適用される期間の名前。例えば、「月」と指定できます。
RPT_NAME	レポートの名前。
METRIC_ NAME	レポートによって収集されたデータのタイプのラベル。デフォルトでは、このパラメータに指定された値は、グラフの y 軸の名前として表示されます。例えば、ユーザーがクローズしたタスクに関する情報を収集するレポートがあるとします。メトリック名に「クローズされたタスクの数」を使用できます。
RPT_DESC	レポートの説明。
RPT_TYPE	レポートがドリルダウンレポートかどうかを示します。値が null の場合、レポートはルートレポートです。値が null 以外の場合、レポートはドリルダウンレポートです。
fieldsMetadata	レポートデータ配列のフィールドを一覧表示します。値には、DIMENSION_VALUE_1、DIMENSION_VALUE_2、TIMEPERIOD_VALUE、METRIC_VALUE、DRILLDOWN_RPT_ID が含まれます。
data	レポートデータ配列の各フィールドの値を一覧表示します。

サンプル API 要求

次のサンプル要求は、レポート設定とレポートのデータを返します。

```
GET http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/data/MDM.RPT.1
```

サンプル API 応答

次のサンプル応答は、レポート設定とレポートのデータを示しています。

```
{
  "metadata": {
    "ROWID_RPT_CONFIG": "11 ",
    "DIMENSION_NAME_1": "Country code",
    "DIMENSION_NAME_2": "null",
    "TIMEPERIOD_NAME": "null",
    "RPT_NAME": "Customer by Region",
    "METRIC_NAME": "Customer by Region",
    "RPT_DESC": "Customer by Region",
    "RPT_TYPE": "null",
    "fieldsMetadata": ["DIMENSION_VALUE_1", "DIMENSION_VALUE_2", "TIMEPERIOD_VALUE", "METRIC_VALUE",
"DRILLDOWN_RPT_ID"]
  },
```

```
  "data": [
    [
      "CA ",
      "null",
      "null",
      "9",
      "123"
  ],
    [
      "IN ",
      "null",
      "null",
      "19",
      "456"
  ],
    [
      "US ",
      "United States",
      "null",
      "9",
      "678"
  ],
    [
      "IN ",
      "India ",
      "null",
      "19",
      null
  ]
  ]
}
```


レポート設定およびドリルダウンレポートの取得

レポート設定およびドリルダウンレポートの取得 REST API は、レポート設定および関連付けられたドリルダウンレポートを返します。

この API は GET メソッドを使用します。

要求 URL

［レポート設定とドリルダウンレポートの取得］REST URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/meta/<report ID>
```

この URL に対して次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/<context>/<database ID>/meta/<report ID>
```

クエリパラメータ

レポート ID は必須 URL パラメータです。レポート ID を使用して、設定および関連するドリルダウンレポートの対象となるレポートを指定します。

応答本文

応答本文には、次のパラメータが含まれます。

パラメータ	説明
ROWID_RPT_CONFIG	レポートの ID。
DIMENSION_NAME_1	レポート内のデータの次元の名前。次元は例えば、ユーザー名、レビュータスクタイプ、ビジネスエンティティタイプなどです。
DIMENSION_NAME_2	オプション。レポート内のデータの次元の名前。次元は例えば、ユーザー名、レビュータスクタイプ、ビジネスエンティティタイプなどです。
TIMEPERIOD_NAME	オプション。データが適用される期間の名前。例えば、「月」と指定できます。
RPT_NAME	レポートの名前。
METRIC_ NAME	レポートによって収集されたデータのタイプのラベル。デフォルトでは、このパラメータに指定された値は、グラフの y 軸の名前として表示されます。例えば、ユーザーがクローズしたタスクに関する情報を収集するレポートがあるとします。メトリック名に「クローズされたタスクの数」を使用できます。
RPT_DESC	レポートの説明。
RPT_TYPE	レポートがドリルダウンレポートかどうかを示します。値が null の場合、レポートはルートレポートです。値が null 以外の場合、レポートはドリルダウンレポートです。
DRILLDOWN	ドリルダウンレポートを一覧表示します。一覧表示されているレポートの順序は、ルートレポートに対するレポートのレベルを反映しています。
RPT_NAME	ドリルダウンレポートの名前。
DETAILED_RPT_IDS	レポート詳細の ID を一覧表示します。ドリルダウンレベルのドリルダウンレポートのレポート詳細 ID が含まれます。
CONFIG_RPT_IDS	ドリルダウンレベルのドリルダウンレポートの ID を一覧表示します。

サンプル API 要求

次のサンプル要求は、レポート設定と関連するドリルダウンレポートを返します。

```
GET http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/meta/SVR1.2X9NO
```

サンプル API 応答

次のサンプル応答は、レポート設定と関連するドリルダウンレポートを示しています。

```
{
  "ROWID_RPT_CONFIG": "7 ",
  "DIMENSION_NAME_1": "Country",
  "DIMENSION_NAME_2": "Users",
  "TIMEPERIOD_NAME": "Month",
  "RPT_NAME": "Top 10 Countries",
  "METRIC_NAME": "Number Of Customers",
  "RPT_DESC": "Top 10 Countries",
  "RPT_TYPE": "null",
  "DRILLDOWN":
  [
```

```
    {
      "RPT_NAME": "Customer by region - First level drilldown",
      "DETAILED_RPT_IDS":["1", "2", "3"],
      "CONFIG_RPT_IDS":["1"]
    },
    {

      "RPT_NAME": "Customer by city - Second level drilldown",
      "DETAILED_RPT_IDS":["5", "6", "7", "8", "9", "10"],
      "CONFIG_RPT_IDS":["2"]
    }
  ]
}
```

登録レポート

登録レポート REST API は、カスタムレポートを登録します。API は、他の API で使用できるレポートのレポート ID を返します。

この API は POST メソッドを使用します。

追加設定なしで使用できるレポートの登録については、「追加設定なしで使用できるレポートの管理」 (ページ 342)を参照してください。

要求 URL

［レポートの登録］REST URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/list
```

URL に対して次の HTTP POST 要求を行います。

```
POST http://<host>:<port>/<context>/<database ID>/list
```

要求本文

レポート設定を指定します。

次の表では、要求本文のレポート詳細のパラメータについて説明します。

パラメータ	説明
ROWID_RPT_CONFIG	レポートの ID。
DIMENSION_NAME_1	レポート内のデータの次元の名前。次元は例えば、ユーザー名、レビュータスクタイプ、ビジネスエンティティタイプなどです。
DIMENSION_NAME_2	オプション。レポート内のデータの次元の名前。次元は例えば、ユーザー名、レビュータスクタイプ、ビジネスエンティティタイプなどです。
TIMEPERIOD_NAME	オプション。データが適用される期間の名前。例えば、「月」と指定できます。
RPT_NAME	レポートの名前。
METRIC_ NAME	レポートによって収集されたデータのタイプのラベル。デフォルトでは、このパラメータに指定された値は、グラフの y 軸の名前として表示されます。例えば、ユーザーがクローズしたタスクに関する情報を収集するレポートがあるとします。メトリック名に「クローズされたタスクの数」を使用できます。

パラメータ	説明
RPT_DESC	レポートの説明。
RPT_TYPE	レポートがドリルダウンレポートかどうかを示します。値が null の場合、レポートはルートレポートです。値が null 以外の場合、レポートはドリルダウンレポートです。

サンプル API 要求

次のサンプル要求は、レポートを登録します。

```
POST http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/list
{
  "DIMENSION_NAME_1": "Country",
  "DIMENSION_NAME_2": "Users",
  "TIMEPERIOD_NAME": "null",
  "RPT_NAME": "Top 10 Countries",
  "METRIC_NAME": "Number Of Customers",
  "RPT_DESC": "Top 10 Countries"
 }
```

サンプル API 応答

次のサンプルは、レポートを正常に登録したときの応答を示しています。

```
{
  "ROWID_RPT_CONFIG": "SVR1.2X9N0",
  "DIMENSION_NAME_1": "Subject area",
  "DIMENSION_NAME_2": "Source System",
  "TIMEPERIOD_NAME": "Month",
  "RPT_NAME": "randomname750",
  "RPT_DESC": "Source system Metrics",
  "METRIC_NAME": "Number Of Records",
  "RPT_TYPE": null
}
```

API は、ROWID_RPT_CONFIG パラメータでレポート ID を返します。

レポート設定の更新

レポート設定の更新 REST API は、レポートの設定を更新します。

この API は POST メソッドを使用します。

要求 URL

［レポート設定の更新］REST URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/list/<report ID>
```

URL に対して次の HTTP POST 要求を行います。

```
POST http://<host>:<port>/<context>/<database ID>/list/<report ID>
```

クエリパラメータ

レポート ID は必須 URL パラメータです。レポート ID を使用して、設定を更新するレポートを指定します。

要求本文

更新するレポート設定を指定します。

次の表に、要求本文内のパラメータを示します。

パラメータ	説明
DIMENSION_NAME_1	レポート内のデータの次元の名前。次元は例えば、ユーザー名、レビュータスクタイプ、ビジネスエンティティタイプなどです。
DIMENSION_NAME_2	オプション。レポート内のデータの次元の名前。次元は例えば、ユーザー名、レビュータスクタイプ、ビジネスエンティティタイプなどです。
TIMEPERIOD_NAME	オプション。データが適用される期間の名前。例えば、「月」と指定できます。
RPT_NAME	レポートの名前。
METRIC_ NAME	レポートによって収集されたデータのタイプのラベル。デフォルトでは、このパラメータに指定された値は、グラフの y 軸の名前として表示されます。例えば、ユーザーがクローズしたタスクに関する情報を収集するレポートがあるとします。メトリック名に「クローズされたタスクの数」を使用できます。
RPT_DESC	レポートの説明。

サンプル API 要求

次のサンプル要求は、レポートの設定を更新します。

```
POST http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/list/SVR1.2X9NO
{
  "DIMENSION_NAME_1": "Country",
  "DIMENSION_NAME_2": "Users",
  "TIMEPERIOD_NAME": "null",
  "RPT_NAME": "Top 10 Countries",
  "METRIC_NAME": "Number Of Customers",
  "RPT_DESC": "Top 10 Countries"
}
```

サンプル API 応答

レポート設定が正常に更新された場合、API は応答を返しません。

レポートデータの追加または更新

レポートデータの追加または更新 REST API は、レポートのデータエントリを追加または更新します。

注: レポートにレポートデータが含まれる場合、API はレポートデータをオーバーライドします。

この API は POST メソッドを使用します。

要求 URL

［レポートデータの追加または更新］REST URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/data/<report ID>
```

URL に対して次の HTTP POST 要求を行います。

```
POST http://<host>:<port>/<context>/<database ID>/data/<report ID>
```

クエリパラメータ

レポート ID は必須 URL パラメータです。レポート ID を使用して、データエントリを追加または更新するレポートを指定します。

要求本文

データエントリを追加または更新するレポートを指定します。

次の表では、要求本文のレポートデータのパラメータについて説明します。

パラメータ	説明
DIMENSION_NAME_1	レポート内のデータの次元の名前。次元は例えば、ユーザー名、レビュータスクタイプ、ビジネスエンティティタイプなどです。
DIMENSION_NAME_2	オプション。レポート内のデータの次元の名前。次元は例えば、ユーザー名、レビュータスクタイプ、ビジネスエンティティタイプなどです。
TIMEPERIOD_NAME	オプション。データが適用される期間の名前。例えば、「月」と指定できます。
METRIC_VALUE	データの次元を表す数値。例えば、ユーザーがクローズしたタスクに関する情報を収集するレポートがあるとします。レポートの 1 つの次元はユーザー John Smith で、別の次元はレビュータスクタイプです。メトリック値は 5 となる場合があります。これは、John Smith がレビュータスクタイプの 5 つのタスクを閉じたことを意味します。
DRILLDOWN_RPT_ID	ドリルダウンレポートの ID。値が null の場合、設定されたドリルダウンレポートはありません。

サンプル API 要求

次のサンプル要求は、レポートのデータエントリを追加または更新します。

```
POST http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/data/SVR1.2X9N0
[
  {
    "DIMENSION_VALUE_1": "11:Org,ex.Phone",
    "DIMENSION_VALUE_2": "Cross Source Matches",
    "TIMEPERIOD_VALUE": "null",
    "METRIC_VALUE": "5",
    "DRILLDOWN_RPT_ID": null
 },
  {
    "DIMENSION_VALUE_1": "9:Org Name,City,Zip",
    "DIMENSION_VALUE_2": "Cross Source Matches",
    "TIMEPERIOD_VALUE": "null",
    "METRIC_VALUE": "40",
    "DRILLDOWN_RPT_ID": null
 },
 {
    "DIMENSION_VALUE_1": "12:Org,ex.Email",
    "DIMENSION_VALUE_2": "CRM Matches",
    "TIMEPERIOD_VALUE": "null",
    "METRIC_VALUE": "7",
    "DRILLDOWN_RPT_ID": null
 }
]
```

サンプル API 応答

レポートのデータエントリが正常に追加または更新された場合、API は応答を返しません。

レポートの削除

レポートの削除 REST API は、レポートを削除します。

この API は DELETE メソッドを使用します。

要求 URL

［レポートの削除］REST URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/data/<report ID>
```

［削除］URL に対して、次の HTTP 要求を行います。

```
DELETE http://<host>:<port>/<context>/<database ID>/data/<report ID>
```

クエリパラメータ

レポート ID は必須 URL パラメータです。レポート ID を使用して削除するレポートを指定します。

サンプル API 要求

次のサンプル要求は、レポートを削除します。

```
DELETE http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/data/SVR1.2X9NO
```

サンプル API 応答

レポートが正常に削除された場合、API は応答を返しません。

レポート更新ジョブの実行

［レポート更新ジョブの実行］REST API は、追加設定なしで使用できるレポートに関連付けられたバッチジョブを開始します。

この API は GET メソッドを使用します。

要求 URL

［レポート更新ジョブの実行］REST URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/data/collect/<report ID>
```

この URL に対して次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/<context>/<database ID>/data/collect/<report ID>
```

クエリパラメータ

レポート ID は必須 URL パラメータです。レポート ID を使用して、更新ジョブを実行するレポートを指定します。

応答本文

応答の status パラメータには、次のいずれかの値が含まれる場合があります。

値	説明
PROCESSING	ジョブがトリガされました。ジョブが完了すると、ジョブ ID が返されます。
COMPLETED_SUCCESSFULLY	レポートデータは、ジョブを実行せずにすぐに計算されました。
DONE	ジョブは開始されず、同期モードで計算が実行されました。
FAILED	エラーが発生したため、ジョブは開始されませんでした。errorMessage および errorCode 属性はエラーを説明します。

サンプル API 要求

次のサンプル要求は、レポートの更新ジョブを実行します。

```
GET http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/data/collect/SVR1.2X9NO
```

サンプル API 応答

次のサンプルは、レポートの更新ジョブを正常に実行したときの応答を示しています。

```
{
  "status": "PROCESSING"
}
```

レポートの更新ジョブのステータスの取得

レポートの更新ジョブのステータスの取得 REST API は、レポートの更新ジョブのステータスを返します。

この API は GET メソッドを使用します。

要求 URL

［レポート更新ジョブのステータスの取得］REST URL の形式は次のとおりです。

```
 http://<host>:<port>/<context>/<database ID>/data/collect/<report ID>/status
```

この URL に対して次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/<context>/<database ID>/data/collect/<report ID>/status
```

クエリパラメータ

レポートの ID は必須 URL パラメータです。report ID パラメータを使用して、更新ジョブのステータスを取得するレポートを指定します。

応答本文

応答には、次のパラメータが含まれます。

パラメータ	説明
status	レポート更新ジョブのステータス。値は COMPLETED_SUCCESSFULLY、PROCESSING、FAILED、または ERROR です。
lastUpdateDate	レポートの最終更新日。COMPLETED_SUCCESSFULLY ステータスを使用してレポート更新ジョブに対して定義されています。
errorCode	オプション。レポート更新ジョブでのエラーのエラーコード。ERROR ステータスを使用してレポート更新ジョブに対して定義されています。
errorMessage	オプション。レポート更新ジョブでのエラーのエラーメッセージ。ERROR ステータスを使用してレポート更新ジョブに対して定義されています。

サンプル API 要求

次のサンプル要求は、レポート更新ジョブのステータスを返します。

```
GET http://<host>:<port>/<context>/<database ID>/data/collect/SVR1.2X9N1/status
```

サンプル API 応答

次のサンプルは、レポート更新ジョブのステータスをチェックしたときの応答を示しています。

```
{
  "status": "COMPLETED_SUCCESSFULLY",
  "jobId": "SVR1.2X9N1 ",
  "lastUpdateDate": "2019-10-31T14:37:11.846-04:00",
  "startRunDate": "2019-10-31T14:37:09.120-04:00"
}
```

ジョブグループのリスト

ジョブグループのリスト REST API は、既存のジョブグループおよびステータスのリストを返します。ジョブグループタイプでジョブをフィルタできます。例えば、すべてのインポートジョブのみを表示できます。

この API は GET メソッドを使用します。

要求 URL

［ジョブのリストの取得］URL の形式は次のとおりです。

```
http://<host>:<port>/<jobcontrol>/<orsId>/<group?jobGroupType>
```

［ジョブのリストの取得］URL に対して、次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/<jobcontrol>/<orsId>/<group?jobGroupType>
```

パラメータ

既存のジョブグループおよびステータスのリストを返すためのパラメータを指定します。

次の表に、これらのパラメータについて説明します。

パラメータ	パス	説明
jobGroupControlId	パス	ジョブグループの ID。
orsId	パス	既存のジョブグループおよびステータスのリストを返すためのオペレーショナルリファレンスストアデータベース ID。
detailed	クエリ	オプション。以下のいずれかの値を使用します。 True: 既存のジョブグループおよびステータスの詳細なリストを表示する場合。 False: 既存のジョブグループおよびステータスの簡単なリストを表示する場合。
firstRecord	クエリ	オプション。既存のジョブグループのリストの最初のレコード。
jobGroupType	クエリ	オプション。ジョブグループのタイプごとにジョブをリストします。以下のいずれかの値を使用します。 Import Match Import_After_Match Find_Replace
groupBy	クエリ	オプション。ビジネスエンティティに基づきジョブをグループ化します。
recordsToReturn	クエリ	オプション。返す必要があるレコードの数。
returnTotal	クエリ	オプション。True に設定します。返すレコードの合計数。

サンプル API 要求

次のサンプル要求は、既存のジョブグループのリストとステータスを返します。

```
GET /cmx/jobcontrol/localhost-orcl-MDM_SAMPLE/group/SVR1.1G7UY?groupBy=owningTable HTTP/1.1
Host: localhost:8080
```

サンプル API 応答

次のサンプル応答は、既存のジョブグループとステータスのリストを返します。

```
HTTP/1.1 200 OK
Server: JBoss-EAP/7
Content-Disposition: inline;filename=f.txt
X-Powered-By: Undertow/1
Content-Type: application/json;charset=UTF-8
Content-Length: 1551

{
  "jobGroupControlId" : "SVR1.1G7UY",
  "jobGroup" : {
    "type" : "IMPORT",
```

```
    "name" : "BE_Import_RH_LE_PersonView[1]_16f9ffd55b5"
  },
  "startDate" : "2020-01-13T13:06:26.649-05:00",
  "endDate" : "2020-01-13T13:06:40.842-05:00",
  "status" : "COMPLETED_WITH_ERRORS",
  "statusMessage" : "Completed with errors/warnings",
  "errorMessages" : [ "Completed with errors/warnings", "Completed with errors/warnings", "Completed with
errors/warnings", "Completed with errors/warnings", "Completed with errors/warnings", "Completed with errors/
warnings" ],
  "jobItems" : [ {
    "startDate" : "2020-01-13T13:06:36.150-05:00",
    "endDate" : "2020-01-13T13:06:40.808-05:00",
    "status" : "COMPLETED_WITH_ERRORS",
    "owningTable" : "PersonView",
    "errorMessages" : [ "Completed with errors/warnings", "Completed with errors/warnings", "Completed with
errors/warnings", "Completed with errors/warnings", "Completed with errors/warnings", "Completed with errors/
warnings" ],
    "progress" : 100,
    "metrics" : {
      "200" : 6,
      "300" : 5,
      "201" : 5,
      "301" : 4,
      "202" : 1,
      "302" : 1
    }
  } ],
  "parameters" : {
    "mappingId" : "SVR1.1G7UX",
    "fileName" : "my.pdf",
    "systemName" : "Admin",
    "fileId" : "DB_SVR1.1G7UW"
  },
  "metricLabels" : {
    "200" : "Total lines in file",
    "201" : "Converted lines",
    "300" : "Total records",
    "202" : "Rejected lines",
    "301" : "Loaded records",
    "302" : "Rejected records"
  },
  "progress" : 100
}
```


ジョブグループのリスト

ジョブグループのリスト REST API は、バッチグループの各ジョブのデータを返します。

この API は GET メソッドを使用します。

要求 URL

[ジョブグループの一覧表示] URL の形式は次のとおりです。

```
http://<host>:<port>/<jobcontrol>/<orsId>/<groupBy=>
```

[ジョブグループの一覧表示] URL に対して、次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/<jobcontrol>/<orsId>/<groupBy=>
```

パラメータ

バッチグループの各ジョブのデータを返すためのパラメータを指定します。

次の表に、要求本文内のパラメータを示します。

パラメータ	説明
orsID	マッピングを取得する元のオペレーショナルリファレンスストア ID。
groupBy	オプション。要求されたジョブのタイプごとにジョブをグループ化します。
jobGroupControlId	バッチジョブグループ制御 ID を指定します。

サンプル API 要求

次のサンプル要求は、バッチグループ内の各ジョブのデータを返します。

```
GET /cmx/jobcontrol/localhost-orcl-MDM_SAMPLE/group?jobGroupType=IMPORT HTTP/1.1
Host: localhost:8080
```

サンプル API 応答

次のサンプル応答は、バッチグループ内の各ジョブのデータを示しています。

```
HTTP/1.1 200 OK
Server: JBoss-EAP/7
X-Powered-By: Undertow/1
Content-Type: application/json;charset=UTF-8
Content-Length: 564

{
  "records" : [ {
    "jobGroupControlId" : "SVR1.1G7UY",
    "startDate" : "2020-01-13T13:06:26.649-05:00",
    "status" : "PENDING",
    "parameters" : { },
    "metricLabels" : { }
  }, {
    "jobGroupControlId" : "SVR1.1G61Y",
    "startDate" : "2020-01-13T12:39:42.152-05:00",
    "endDate" : "2020-01-13T12:40:03.221-05:00",
    "status" : "COMPLETED_WITH_ERRORS",
    "statusMessage" : "Completed with errors/warnings",
    "parameters" : { },
    "metricLabels" : { }
  } ],
  "totalCount" : 2,
  "firstRecord" : 1,
  "pageSize" : 10
}
```

DaaS メタデータの取得

［DaaS メタデータの取得］REST API は、名前、タイプ、使用するビジネスエンティティ、必須フィールドのリストなど、Daas プロバイダについての情報を返します。

この API は GET メソッドを使用します。

要求 URL

［DaaS メタデータの取得］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/meta/daas/<providerName>
```

［DaaS メタデータの取得］URL に対して、次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/<context>/<database ID>/meta/daas/<providerName>
```

クエリパラメータ

providerName パラメータは必須パラメータです。このパラメータは、構成されている DaaS プロバイダの名前です。

サンプル API 要求

次のサンプル要求は、dcp DaaS プロバイダのメタデータ情報を返します。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/meta/daas/dcp
```

次のサンプル要求は、ondemand DaaS プロバイダのメタデータ情報を返します。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/meta/daas/ondemand
```

サンプル API 応答

次の例は、dcp DaaS プロバイダのメタデータ情報を JSON 形式で示しています。

```
{
                    providerName: "dcp",
                    providerType: "READ",
                    businessEntity: "Organization",
                    systemName: "SFA",
                    requiredFields: [
                                  "dunsNumber"
                    ]
}
```

次の例は、ondemand DaaS プロバイダのメタデータ情報を JSON 形式で示しています。

```
{
                    providerName: "ondemand",
                    providerType: "SEARCH",
                    businessEntity: "Organization",
                    systemName: "SFA",
                    requiredFields: [
                                "displayName"
                    ]
}
```

DaaS 検索

［DaaS 検索］REST API は、ビジネスエンティティのいくつかの入力フィールドを使用して外部 DaaS サービスを呼び出し、応答をレコードのリストに変換します。

この API は POST メソッドを使用します。

要求 URL

［DaaS 検索］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/daas/search/<daas provider>
```

［DaaS 検索］URL に対して、次の HTTP POST 要求を行います。

POST http://<host>:<port>/<context>/<database ID>/daas/search/<daas provider>

注: 要求本文で、必須フィールドを使用してビジネスエンティティの詳細を提供します。

要求本文

要求本文には、urn:co-ors.informatica.mdm 名前空間からのタイプ Organization または OrganizationView の XML 要素または JSON 要素を含める必要があります。

サンプル API 要求

次の例では、表示名 INFA を持つ organization ビジネスエンティティを検索するよう、DaaS プロバイダ ondemand に要求しています。

```
POST http://localhost:8080/cmx/cs/localhost-orcl-MDM_SAMPLE/daas/search/ondemand
{
  "Organization": {
    "displayname": "INFA"
  }
}
```

サンプル API 応答

次のサンプル応答は、表示名 INFA を持つ Organization ビジネスエンティティについて、DaaS プロバイダが返した検索結果を示しています。

```
{
  "link": [],
  "item": [
    {
      "businessEntity": {
        "Organization": {
          "displayName": "INFA LAB INC",
          "dunsNumber": "001352574",
          "TelephoneNumbers": {
            "link": [],
            "item": [
              {
                "phoneNum": "9736252265"
              }
            ]
          },
          "Addresses": {
            "link": [],
            "item": [
              {
                "Address": {
                  "cityName": "ROCKAWAY",
                  "addressLine1": "11 WALL ST",
                  "postalCd": "07866",
                  "stateCd": {
                    "stateAbbreviation": "NJ"
                  }
                }
              }
            ]
          }
        }
      },
      "label": "001352574-INFA LAB INC",
      "systemName": "SFA"
    },
    {
      "businessEntity": {
        "Organization": {
```

```
        "displayName": "INFA INC",
        "dunsNumber": "007431013",
        "TelephoneNumbers": {
          "link": [],
          "item": [
            {
              "phoneNum": "5629019971"
            }
          ]
        },
        "Addresses": {
          "link": [],
          "item": [
            {
              "Address": {
                "cityName": "LONG BEACH",
                "addressLine1": "3569 GARDENIA AVE",
                "postalCd": "90807",
                "stateCd": {
                  "stateAbbreviation": "CA"
                }
              }
            }
          ]
        }
      }
    },
    "label": "007431013-INFA INC",
    "systemName": "SFA"
  },
  {
    "businessEntity": {
      "Organization": {
        "displayName": "INFA INC",
        "dunsNumber": "020591086",
        "TelephoneNumbers": {
          "link": [],
          "item": [
            {
              "phoneNum": "7186248777"
            }
          ]
        },
        "Addresses": {
          "link": [],
          "item": [
            {
              "Address": {
                "cityName": "BROOKLYN",
                "addressLine1": "16 COURT ST STE 2004",
                "postalCd": "11241",
                "stateCd": {
                  "stateAbbreviation": "NY"
                }
              }
            }
          ]
        }
      }
    },
    "label": "020591086-INFA INC",
    "systemName": "SFA"
  },
  {
    "businessEntity": {
      "Organization": {
        "displayName": "INFA INC",
        "dunsNumber": "604057286",
        "TelephoneNumbers": {
          "link": [],
          "item": [
```

```
          {
            "phoneNum": "8473580802"
          }
        ]
      },
      "Addresses": {
        "link": [],
        "item": [
          {
            "Address": {
              "cityName": "PALATINE",
              "addressLine1": "THE HARRIS BANK BLDG STE 614,800E NW HWY",
              "postalCd": "60074",
              "stateCd": {
                "stateAbbreviation": "IL"
              }
            }
          }
        ]
      }
    }
  },
  "label": "604057286-INFA INC",
  "systemName": "SFA"
},
{
  "businessEntity": {
    "Organization": {
      "displayName": "INFA INC",
      "dunsNumber": "032785606",
      "TelephoneNumbers": {
        "link": [],
        "item": [
          {
            "phoneNum": "5629019971"
          }
        ]
      },
      "Addresses": {
        "link": [],
        "item": [
          {
            "Address": {
              "cityName": "SIMI VALLEY",
              "addressLine1": "3962 HEMWAY CT",
              "postalCd": "93063",
              "stateCd": {
                "stateAbbreviation": "CA"
              }
            }
          }
        ]
      }
    }
  },
  "label": "032785606-INFA INC",
  "systemName": "SFA"
},
{
  "businessEntity": {
    "Organization": {
      "displayName": "INFA",
      "dunsNumber": "045228877",
      "TelephoneNumbers": {
        "link": [],
        "item": [
          {
            "phoneNum": "3304410033"
          }
        ]
      },
```

```
      "Addresses": {
        "link": [],
        "item": [
          {
            "Address": {
              "cityName": "NORTON",
              "addressLine1": "4725 ROCK CUT RD",
              "postalCd": "44203",
              "stateCd": {
                "stateAbbreviation": "OH"
              }
            }
          }
        ]
      }
    }
  },
  "label": "045228877-INFA",
  "systemName": "SFA"
},
{
  "businessEntity": {
    "Organization": {
      "displayName": "INFA INC",
      "dunsNumber": "609028209",
      "TelephoneNumbers": {
        "link": [],
        "item": [
          {
            "phoneNum": "9084394655"
          }
        ]
      },
      "Addresses": {
        "link": [],
        "item": [
          {
            "Address": {
              "cityName": "CALIFON",
              "addressLine1": "21 FAIRMOUNT RD W",
              "postalCd": "07830",
              "stateCd": {
                "stateAbbreviation": "NJ"
              }
            }
          }
        ]
      }
    }
  },
  "label": "609028209-INFA INC",
  "systemName": "SFA"
},
{
  "businessEntity": {
    "Organization": {
      "displayName": "INFA INC",
      "dunsNumber": "195271796",
      "TelephoneNumbers": {
        "link": [],
        "item": [
          {
            "phoneNum": "7137824181"
          }
        ]
      },
      "Addresses": {
        "link": [],
        "item": [
          {
            "Address": {
```

```
                    "cityName": "HOUSTON",
                    "addressLine1": "1800 AUGUSTA DR STE 200",
                    "postalCd": "77057",
                    "stateCd": {
                      "stateAbbreviation": "TX"
                    }
                  }
                }
              }
            ]
          }
        }
      },
      "label": "195271796-INFA INC",
      "systemName": "SFA"
    },
    {
      "businessEntity": {
        "Organization": {
          "displayName": "INFA INC",
          "dunsNumber": "098605830",
          "Addresses": {
            "link": [],
            "item": [
              {
                "Address": {
                  "cityName": "BELLFLOWER",
                  "postalCd": "90707",
                  "stateCd": {
                    "stateAbbreviation": "CA"
                  }
                }
              }
            ]
          }
        }
      },
      "label": "098605830-INFA INC",
      "systemName": "SFA"
    }
  ]
}
```


DaaS 読み取り

[DaaS 読み取り] REST API は、ビジネスエンティティのいくつかのフィールドを使用して外部 DaaS サービスを要求し、応答を完全なレコードに変換します。

この API は POST メソッドを使用します。DaaS プロバイダへの要求で必須フィールドを指定します。

要求 URL

[DaaS 読み取り] URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/daas/read/<daas provider>
```

[DaaS 読み取り] URL に対して、次の POST 要求を行います。

```
POST http://<host>:<port>/<context>/<database ID>/daas/read/<daas provider>
```

注: 要求本文で、必須フィールドを使用してレコードの詳細を提供します。

クエリパラメータ

次の表に、使用できるクエリパラメータを示します。

パラメータ	説明
logChanges	オプション。true に設定した場合、変換されたレコードには、書き込みビジネスエンティティサービスに渡される（サービスデータオブジェクト）SDO 変更サマリが含まれます。デフォルトは false です。

要求本文

要求本文には、urn:coors.informatica.mdm 名前空間からのタイプ OrganizationView の XML 要素または JSON 要素を含める必要があります。

サンプル API 要求

次の例では、表示名 INFA を持つ organization ビジネスエンティティを検索するよう、DaaS プロバイダ ondemand に要求しています。

```
POST http://localhost:8080/cmx/cs/localhost-orcl-MDM_SAMPLE/daas/search/ondemand
{
  "Organization": {
    "displayname": "INFA"
  }
}
```

サンプル API 応答

次のサンプル応答は、D-U-N-S 番号が 001352574 の組織について、DaaS プロバイダが返した詳細を示しています。

```
{
  "Organization": {
    "displayName": "Infa Lab Inc",
    "dunsNumber": "001352574",
    "TelephoneNumbers": {
      "link": [],
      "item": [
        {
          "phoneNum": "09736250550"
        }
      ]
    },
    "Addresses": {
      "link": [],
      "item": [
        {
          "Address": {
            "cityName": "Rockaway",
            "addressLine1": "11 WALL ST"
          }
        }
      ]
    }
  }
}
```

WriteMerge

［WriteMerge］REST API は DaaS プロバイダから取得したレコードのリストを受け入れて、適切なソースシステムを使用して別個の XREF で保持し、単一のレコードにマージします。すべての XREF は同じレコードに属します。

この API は POST メソッドを使用します。

要求 URL

［WriteMerge］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/daas/write-merge/<business entity name>
```

［WriteMerge］URL に対して、次の HTTP POST 要求を行います。

```
POST http://<host>:<port>/<context>/<database ID>/daas/write-merge/<business entity name>
```

要求本文

要求本文には、urn:cobase.informatica.mdm 名前空間からのタイプ DaaSEntity.Pager の XML または JSON 要素を含める必要があります。

応答ヘッダー

応答に成功すると、API は応答ヘッダーで interactionId と processId を返します。

サンプル API 要求

次のサンプル要求は、DaaS 検索の結果を入力として使用して、Organization ビジネスエンティティタイプのレコードを作成します。

```
POST http://localhost:8080/cmx/cs/localhost-orcl-MDM_SAMPLE/daas/write-merge/Organization
{
  "item": [
    {
      "businessEntity": {
        "Organization": {
          "displayName": "INFA LAB INC",
          "dunsNumber": "001352574",
          "TelephoneNumbers": {
            "item": [
              {
                "phoneNum": "9736252265"
              }
            ]
          }
        }
      },
      "systemName": "DNB"
    },
    {
      "businessEntity": {
        "Organization": {
          "displayName": "INFA INC",
          "dunsNumber": "007431013",
          "TelephoneNumbers": {
            "item": [
              {
                "phoneNum": "5629019971"
              }
            ]
          }
        }
      },
```

```
      "systemName": "Admin"
    }
  ]
}
```

サンプル API 応答

次のサンプル応答は、レコードのリストを単一レコードに正常にマージした後の、応答ヘッダーと応答本文を示しています:

```
{
  "Organization": {
    "key": {
      "rowid": "121921",
      "sourceKey": "140000000000"
    },
    "rowidObject": "121921",
    "TelephoneNumbers": {
      "link": [],
      "item": [
        {
          "key": {
            "rowid": "21721",
            "sourceKey": "140000001000"
          },
          "rowidObject": "21721"
        }
      ]
    }
  }
}
```

Daas インポート

［DaaS インポート］REST API は、データを XML 形式で受け取って、レコードに変換します。

この API は POST メソッドを使用します。

要求 URL

［DaaS データのインポート］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/daas/import/FamilyTreeMemberOrganizationToOrgView
```

［DaaS データのインポート］URL に対して、次の HTTP POST 要求を行います。

```
POST http://<host>:<port>/<context>/<database ID>/daas/import/FamilyTreeMemberOrganizationToOrgView
```

クエリパラメータ

ソースシステムの名前は必須パラメータです。

次の表に、要求で使用できるパラメータを示します。

パラメータ	説明
systemName	必須。データ変更を実行するソースシステムの名前。
interactionId	オプション。すべての変更に割り当てる相互作用 ID。一般に、呼び出しの結果として保留中の変更が作成される際に、Hub で ID が生成されます。

パラメータ	説明
effectivePeriod	オプション。開始日と終了日が含まれます。レコードが有効である期間を指定します。タイムラインが有効なレコードにこれらのパラメータを指定します。
validateOnly	オプション。TRUE に設定した場合、変更されたレコードに検証ルールが適用されるだけで、変更内容は保持されません。
recordState	オプション。作成または更新されたレコードの Hub 状態。サポートされるレコードの状態は、ACTIVE、および PENDING です。
processId	オプション。タスクを含むワークフロープロセスの ID。サービス呼び出しの結果としてワークフローが開始されると、パラメータには開始されたワークフロープロセスの識別子が含められます。

関連項目：

- 「UTC での日付と時刻の形式」 (ページ 34)

要求本文

要求本文には、urn:co-ors.informatica.mdm 名前空間からのタイプ DaaSChangeFamilyTreeMemberOrganizationToOrgView の XML 要素または JSON 要素を含める必要があります。

応答ヘッダー

応答が正常な場合、API は応答ヘッダー内の interactionId および processId と、応答本文内のレコード詳細を返します。

プロセスが相互作用 ID を生成し、それをレコードの作成に使用する場合、API はその相互作用 ID を返します。プロセスが、レコードを直接データベースに保存せず、ワークフローを開始する場合、API はワークフロープロセスの ID であるプロセス ID を返します。

次の例は、相互作用 ID とプロセス ID が含まれる応答ヘッダーを示しています。

```
BES-interactionId: 72200000242000
BES-processId: 15948
```

サンプル API 要求

この要求は常に XML 形式です。

次のサンプル要求は、リンケージ名前空間からのタイプ ChildAssociation の XML データを示します。

```
POST http://localhost:8080/cmx/cs/localhost-orcl-MDM_SAMPLE/daas/import/linkage2org?systemName=Admin
<FamilyTreeMemberOrganization xmlns="http://services.dnb.com/LinkageServiceV2.0" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance" xsi:type="ChildAssociation">
    <AssociationTypeText>ParentSubsidiary</AssociationTypeText>
    <OrganizationName>
        <OrganizationPrimaryName>
            <OrganizationName>INFORMATICA JAPAN K.K.</OrganizationName>
        </OrganizationPrimaryName>
    </OrganizationName>
    <SubjectHeader>
        <DUNSNumber>697557825</DUNSNumber>
    </SubjectHeader>
    <Location>
        <PrimaryAddress>
            <StreetAddressLine>
                <LineText>2-5-1, ATAGO</LineText>
            </StreetAddressLine>
```

```
        <StreetAddressLine>
            <LineText>ATAGO GREEN HILLS MORI TOWER 26F.</LineText>
        </StreetAddressLine>
        <PrimaryTownName>MINATO-KU</PrimaryTownName>
        <CountryISOAlpha2Code>JP</CountryISOAlpha2Code>
        <TerritoryAbbreviatedName>TKY</TerritoryAbbreviatedName>
        <PostalCode>105-0002</PostalCode>
        <TerritoryOfficialName>TOKYO</TerritoryOfficialName>
    </PrimaryAddress>
</Location>
<OrganizationDetail>
    <FamilyTreeMemberRole>
        <FamilyTreeMemberRoleText>Subsidiary</FamilyTreeMemberRoleText>
    </FamilyTreeMemberRole>
    <FamilyTreeMemberRole>
        <FamilyTreeMemberRoleText>Headquarters</FamilyTreeMemberRoleText>
    </FamilyTreeMemberRole>
    <StandaloneOrganizationIndicator>false</StandaloneOrganizationIndicator>
</OrganizationDetail>
<Linkage>
    <LinkageSummary>
        <ChildrenSummary>
            <ChildrenQuantity>1</ChildrenQuantity>
            <DirectChildrenIndicator>false</DirectChildrenIndicator>
        </ChildrenSummary>
        <ChildrenSummary>
            <ChildrenTypeText>Affiliate</ChildrenTypeText>
            <ChildrenQuantity>29</ChildrenQuantity>
            <DirectChildrenIndicator>false</DirectChildrenIndicator>
        </ChildrenSummary>
        <ChildrenSummary>
            <ChildrenTypeText>Branch</ChildrenTypeText>
            <ChildrenQuantity>1</ChildrenQuantity>
            <DirectChildrenIndicator>true</DirectChildrenIndicator>
        </ChildrenSummary>
        <SiblingCount>29</SiblingCount>
    </LinkageSummary>
    <GlobalUltimateOrganization>
        <DUNSNumber>825320344</DUNSNumber>
    </GlobalUltimateOrganization>
    <DomesticUltimateOrganization>
        <DUNSNumber>697557825</DUNSNumber>
    </DomesticUltimateOrganization>
    <ParentOrganization>
        <DUNSNumber>825320344</DUNSNumber>
    </ParentOrganization>
    <FamilyTreeMemberOrganization>
        <AssociationTypeText>HeadquartersBranch</AssociationTypeText>
        <OrganizationName>
            <OrganizationPrimaryName>
                <OrganizationName>INFORMATICA JAPAN K.K.</OrganizationName>
            </OrganizationPrimaryName>
        </OrganizationName>
        <SubjectHeader>
            <DUNSNumber>692640710</DUNSNumber>
            <SubjectHandling>
                <SubjectHandlingText DNBCodeValue="11028">De-listed</SubjectHandlingText>
            </SubjectHandling>
        </SubjectHeader>
        <Location>
            <PrimaryAddress>
                <StreetAddressLine>
                    <LineText>2-2-2, UMEDA, KITA-KU</LineText>
                </StreetAddressLine>
                <PrimaryTownName>OSAKA</PrimaryTownName>
                <CountryISOAlpha2Code>JP</CountryISOAlpha2Code>
                <TerritoryAbbreviatedName>OSK</TerritoryAbbreviatedName>
                <PostalCode>530-0001</PostalCode>
                <TerritoryOfficialName>OSAKA</TerritoryOfficialName>
            </PrimaryAddress>
        </Location>
```

```
            <OrganizationDetail>
                <FamilyTreeMemberRole>
                    <FamilyTreeMemberRoleText>Branch</FamilyTreeMemberRoleText>
                </FamilyTreeMemberRole>
                <StandaloneOrganizationIndicator>false</StandaloneOrganizationIndicator>
            </OrganizationDetail>
            <Linkage>
                <GlobalUltimateOrganization>
                    <DUNSNumber>825320344</DUNSNumber>
                </GlobalUltimateOrganization>
                <DomesticUltimateOrganization>
                    <DUNSNumber>697557825</DUNSNumber>
                </DomesticUltimateOrganization>
                <HeadquartersOrganization>
                    <DUNSNumber>697557825</DUNSNumber>
                </HeadquartersOrganization>
                <FamilyTreeHierarchyLevel>2</FamilyTreeHierarchyLevel>
            </Linkage>
        </FamilyTreeMemberOrganization>
        <FamilyTreeHierarchyLevel>2</FamilyTreeHierarchyLevel>
    </Linkage>
</FamilyTreeMemberOrganization>
```

サンプル API 応答

次のサンプル応答は、Organization ビジネスエンティティをインポートおよび作成した後の、応答ヘッダーと本文を示しています。

```
BES-interactionId: 72202100242034
BES-processId: 156048
{
  "Organization": {
    "key": {
      "rowid": "101921",
      "sourceKey": "697557825"
    },
    "rowidObject": "101921"
  }
}
```

DaaS 更新

［DaaS 更新］REST API は、変更前後にデータを XML 形式で受け取ります。この API は、変更をレコードに適用します。

この API は POST メソッドを使用します。

要求 URL

［DaaS 更新］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/daas/update/FamilyTreeMemberOrganizationToOrgView
```

［DaaS 更新］URL に対して、次の HTTP POST 要求を行います。

```
POST http://<host>:<port>/<context>/<database ID>/daas/update/FamilyTreeMemberOrganizationToOrgView
```

クエリパラメータ

ソースシステムの名前は必須パラメータです。

次の表に、要求で使用できるパラメータを示します。

パラメータ	説明
systemName	必須。データ変更を実行するソースシステムの名前。
interactionId	オプション。すべての変更に割り当てる相互作用 ID。一般に、呼び出しの結果として保留中の変更が作成される際に、Hub で ID が生成されます。
effectivePeriod	オプション。開始日と終了日が含まれます。レコードが有効である期間を指定します。タイムラインが有効なレコードにこれらのパラメータを指定します。
validateOnly	オプション。TRUE に設定した場合、変更されたレコードに検証ルールが適用されるだけで、変更内容は保持されません。
recordState	オプション。作成または更新されたレコードの Hub 状態。サポートされるレコードの状態は、ACTIVE、および PENDING です。
processId	オプション。タスクを含むワークフロープロセスの ID。サービス呼び出しの結果としてワークフローが開始されると、パラメータには開始されたワークフロープロセスの識別子が含められます。

関連項目：

- 「UTC での日付と時刻の形式」 (ページ 34)

要求本文

要求本文には、urn:co-ors.informatica.mdm 名前空間からのタイプ DaaSChangeFamilyTreeMemberOrganizationToOrgView の XML 要素または JSON 要素を含める必要があります。

応答ヘッダー

応答が正常な場合、API は応答ヘッダー内の interactionId および processId と、応答本文内のレコード詳細を返します。

プロセスが相互作用 ID を生成し、それをレコードの作成に使用する場合、API はその相互作用 ID を返します。プロセスが、レコードを直接データベースに保存せず、ワークフローを開始する場合、API はワークフロープロセスの ID であるプロセス ID を返します。

次の例は、相互作用 ID とプロセス ID が含まれる応答ヘッダーを示しています。

```
BES-interactionId: 72200000242000
BES-processId: 15948
```

サンプル API 要求

この API は、変更前と変更後の 2 つの応答を XML 形式で受け入れます。次の要求では、組織に新しい電話番号が追加されます。before XML データには電話番号がなく、after XML データには電話番号があります。

次の要求には、新たに追加された電話番号が含まれます。

```
POST http://localhost:8080/cmx/cs/localhost-orcl-MDM_SAMPLE/daas/update/linkage2org?systemName=Admin
<urn:DaaSChangelinkage2org xmlns:urn="urn:cs-ors.informatica.mdm" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:type="urn:DaaSChangelinkage2org">
    <urn:before xmlns="http://services.dnb.com/LinkageServiceV2.0">
```

```
        <SubjectHeader>
            <DUNSNumber>697557825</DUNSNumber>
        </SubjectHeader>
    </urn:before>


    <urn:after xmlns="http://services.dnb.com/LinkageServiceV2.0">
        <SubjectHeader>
            <DUNSNumber>697557825</DUNSNumber>
        </SubjectHeader>
        <Telecommunication>
            <TelephoneNumber>
                <TelecommunicationNumber>09736250550</TelecommunicationNumber>
                <InternationalDialingCode>1</InternationalDialingCode>
                <UnreachableIndicator>true</UnreachableIndicator>
            </TelephoneNumber>
        </Telecommunication>
    </urn:after>
</urn:DaaSChangelinkage2org>
```

サンプル API 応答

次の例は、新たに作成した組織の電話番号の行 ID を示しています。

```
{
  "Organization": {
    "key": {
      "rowid": "101921",
      "sourceKey": "697557825"
    },
    "rowidObject": "101921",
    "TelephoneNumbers": {
      "link": [],
      "item": [
        {
          "key": {
            "rowid": "1722",
            "sourceKey": "09736250550"
          },
          "rowidObject": "1722"
        }
      ]
    }
  }
}
```

第 4 章

Data Director の REST API

REST API を使用して、Data Director のブランドを変更し、ログインページの免責事項を設定できます。

Data Director のブランド変更用 REST API

REST API を使用して、Data Director のブランドを変更し、アプリケーションのルックアンドフィールをカスタマイズできます。Data Director アプリケーションのデフォルトのカラーテーマ、ログイン背景、およびロゴを変更できます。

次の表に、Data Director のブランドを変更するための REST API を示します。

REST API	**説明**
ロゴファイルのアップロード	ロゴ画像ファイルをアップロードします。POST メソッドをサポートします。
ロゴファイルの削除	ロゴ画像ファイルを削除します。DELETE メソッドをサポートします。
ログイン BG ファイルのアップロード	ログインページの背景画像ファイルをアップロードします。POST メソッドをサポートします。
ログイン BG ファイルの削除	ログインページの背景画像ファイルを削除します。DELETE メソッドをサポートします。
変数の取得	Data Director アプリケーションのボタン、メニュー、テキストの色などのユーザーインタフェース設定を読み取ります。GET メソッドをサポートします。
変数の更新	Data Director アプリケーションのボタン、メニュー、テキストの色などのユーザーインタフェース設定を更新します。POST メソッドをサポートします。
カラースキームの削除	デフォルトのユーザーインタフェース設定を復元します。DELETE メソッドをサポートします。

ログイン BG ファイルのアップロード

ログイン BG ファイルのアップロード REST API は、Data Director アプリケーションのログインページの背景画像をアップロードします。ファイル名を指定するには、POST 要求の X-File-Name ヘッダーに追加します。REST 要求本文で画像データを送信します。Content-Type ヘッダー値は `application/octet-stream` にする必要があります。

この API は POST メソッドを使用します。

画像のファイルサイズは、cmxserver.properties ファイルで指定された cmx.file.max_file_size_mb プロパティの値以下である必要があります。API は、ブラウザがサポートする画像ファイルタイプをサポートします。サポート対象ブラウザバージョンの詳細については、次の場所にある Product Availability Matrix（PAM）を参照してください。https://network.informatica.com/community/informatica-network/product-availability-matrices

要求 URL

［ログイン BG の更新］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<path>
```

［ログイン BG の更新］URL に対して、次の HTTP POST 要求を行います。

```
POST http://<host>:<port>/cmx/ui/theme/login_bg_image
```

サンプル API **要求**

次のサンプル要求は、ログインページの背景画像ファイルをアップロードします。

```
POST http://localhost:8080/cmx/ui/theme/login_bg_image
```

サンプル API **応答**

Data Director ログインページの新しい背景画像が正常にアップロードされると、API は 200 OK 応答コードを返します。応答本文は空です。

ログイン BG ファイルの削除

ログイン BG ファイルの削除 REST API は、Data Director アプリケーションのログインページからカスタム背景画像を削除します。

この API は DELETE メソッドを使用します。

カスタム背景画像を削除すると、ログインページにデフォルトの背景画像が表示されます。

要求 URL

［ログイン BG ファイルの削除］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<path>
```

［ログイン BG ファイルの削除］URL に対して、次の HTTP DELETE 要求を行います。

```
DELETE http://<host>:<port>/cmx/ui/theme/login_bg_image
```

サンプル API **要求**

次のサンプル要求は、Data Director アプリケーションのログインページから背景画像を削除します。

```
DELETE http://localhost:8080/cmx/ui/theme/login_bg_image
```

サンプル API **応答**

Data Director アプリケーションログインページから背景画像が正常に削除されると、API は 200 OK 応答コードを返します。応答本文は空です。

ロゴファイルのアップロード

ロゴファイルのアップロード REST API は、Data Director アプリケーションの新しいロゴ画像をアップロードします。ファイル名を指定するには、POST 要求の X-File-Name ヘッダーに追加します。REST 要求本文で画像データを送信します。Content-Type ヘッダー値は application/octet-stream にする必要があります。

この API は POST メソッドを使用します。

画像のファイルサイズは、幅 130 ピクセル、高さ 33 ピクセル以内にする必要があります。API は、ブラウザがサポートする画像ファイルタイプをサポートします。サポート対象ブラウザバージョンの詳細については、次の場所にある Product Availability Matrix（PAM）を参照してください。https://network.informatica.com/community/informatica-network/product-availability-matrices

要求 URL

［ロゴファイルのアップロード］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<path>
```

［ロゴファイルのアップロード］URL に対して、次の HTTP POST 要求を行います。

```
POST http://localhost:8080/cmx/ui/theme/logo
```

サンプル API **要求**

次のサンプル要求は、アプリケーションのロゴ画像をアップロードします。

```
POST http://localhost:8080/cmx/ui/theme/logo HTTP/1.1
Host: localhost:8080
X-File-Name: logo_dockers.jpg
Content-Type: application/octet-stream
```

サンプル API **応答**

ロゴ画像が正常にアップロードされると、API は 200 OK 応答コードを返します。応答本文は空です。

ロゴファイルの削除

ロゴファイルの削除 REST API は、カスタムロゴ画像ファイルをアプリケーションから削除します。

この API は DELETE メソッドを使用します。

注: カスタムロゴ画像を削除すると、ログインページにデフォルトのロゴ画像が表示されます。

要求 URL

［ロゴファイルの削除］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<path>
```

［ロゴファイルの削除］URL に対して、次の HTTP DELETE 要求を行います。

```
DELETE http://<host>:<port>/cmx/ui/theme/logo
```

サンプル API **要求** URL

次のサンプル要求は、Data Director ロゴファイルを削除します。

```
DELETE http://localhost:8080/cmx/ui/theme/logo
```

サンプル API **応答**

ロゴ画像ファイルが正常に削除されると、API は 200 OK 応答コードを返します。応答本文は空です。

変数の取得

変数の取得 REST API は、Data Director の現在のカラースキーマプロパティを返します。

この API は GET メソッドを使用します。

要求 URL

［変数の取得］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<path>
```

［変数の取得］URL に対して、次の HTTP GET 要求を行います。

```
GET http://<host>:<port>/cmx/ui/theme/variables
```

サンプル API **要求**

次の要求は、現在のカラーテーマのプロパティを返します。

```
GET http://localhost:8080/cmx/ui/theme/variables
```

サンプル API **応答**

カラースキームプロパティが正常に取得されると、API は 200 OK 応答コードを返します。


```
    "variables": {
        "menu_text_color": "black",
        "dropdown_menu_item_checked_hover_bg_color": "Gold",
        "button_default_selected_border_color": "blue",
        "dropdown_menu_item_checked_hover_text_color": "Coral",
        "button_success_text_color": "white",
        "login_form_text_color": "green",
        "menu_hover_text_color": "#07fc03",
        "dropdown_menu_item_checked_text_color": "LightSalmon",
        "button_danger_text_color": "black",
        "menu_hover_bg_color": "#00349d",
        "header_search_text_color": "#36270a",
        "button_default_border_color": "red",
        "dropdown_menu_item_text_color": "#daf97b",
        "button_danger_bg_color": "pink",
        "button_default_bg_color": "yellow",
        "menu_sel_text_color": "#26cfe0",
        "button_success_border_color": "olive",
        "dropdown_menu_text_color": "#945d29",
        "dropdown_menu_bg_color": "#a4c7f5",
        "primary_button_text_color": "white",
        "header_search_bg_color": "#a4c7f5",
        "dropdown_menu_item_checked_bg_color": "LightGoldenrodYellow",
        "button_default_selected_bg_color": "orange",
        "primary_button_bg_color": "green",
        "entity_view_label_font_wight": "bold",
        "primary_button_selected_bg_color": "pink",
        "primary_button_selected_border_color": "red",
        "header_text_color": "#36270a",
        "button_danger_selected_text_color": "green",
        "primary_button_border_color": "white",
        "menu_sel_bg_color": "#636260",
        "button_success_selected_border_color": "red",
        "login_form_background_color": "#fcd483",
        "dropdown_menu_item_hover_text_color": "OrangeRed",
        "header_background_color": "#c4922d",
        "button_success_selected_bg_color": "PaleVioletRed",
        "button_danger_border_color": "purple",
        "button_default_text_color": "red",
        "button_success_bg_color": "orange",
        "button_danger_selected_bg_color": "yellow",
        "button_danger_selected_border_color": "red",
        "primary_button_selected_text_color": "red",
        "workspace_bg": "#cadee0",
        "button_default_selected_text_color": "blue",
        "dropdown_menu_item_hover_bg_color": "RoyalBlue",
        "menu_background_color": "#cdc1bb",
        "button_success_selected_text_color": "black"
    }

}
```


Data Director のカラースキームプロパティの詳細については、「デフォルトのカラースキーム」 (ページ 257)を参照してください。

変数の更新

変数の更新 REST API は、Data Director の現在のカラープロパティを更新します。JSON 形式を使用して、要求本文でデータを送信します。

この API は POST メソッドを使用します。
注: API 要求は、アプリケーション全体で一貫してカラープロパティを更新しない場合があります。例えば、ボタンとメニューのカラープロパティを更新する要求を送信した後、［保存］ボタンは画面ごとに異なり、一部のメニューでは背景色のみが変更される場合があります。

要求 URL

［変数の更新］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<path>
```

［変数の更新］URL に対して、次の HTTP POST 要求を行います。

```
POST http://<host>:<port>/cmx/ui/theme/variables
```

サンプル API **要求**

次の要求は、Data Director アプリケーションのすべてのカラープロパティを更新します。

```
POST http://localhost:8080/cmx/ui/theme/variables

{
  "variables": {
    "menu_text_color": "#FFF",
    "button_default_selected_border_color": "orange",
    "button_success_text_color": "white",
    "login_form_text_color": "red",
    "button_danger_text_color": "black",
    "menu_hover_bg_color": "#00349D",
    "button_default_border_color": "yellow",
    "dropdown_menu_item_text_color": "white",
    "button_danger_bg_color": "pink",
    "button_default_bg_color": "yellow",
    "button_success_border_color": "olive",
    "dropdown_menu_text_color": "white",
    "dropdown_menu_bg_color": "olive",
    "primary_button_text_color": "white",
    "button_default_selected_bg_color": "orange",
    "primary_button_bg_color": "green",
    "entity_view_label_font_wight": "normal",
    "primary_button_selected_bg_color": "silver",
    "primary_button_selected_border_color": "black",
    "header_text_color": "#444444",
    "button_danger_selected_text_color": "black",
    "primary_button_border_color": "green",
    "menu_sel_bg_color": "#001E60",
    "button_success_selected_border_color": "black",
    "login_form_background_color": "#E5EDF8",
    "header_background_color": "#ffd900",
    "button_success_selected_bg_color": "silver",
    "button_danger_border_color": "pink",
    "button_default_text_color": "black",
    "button_success_bg_color": "olive",
    "button_danger_selected_bg_color": "orange",
    "button_danger_selected_border_color": "red",
    "primary_button_selected_text_color": "black",
    "workspace_bg": "#E5EDF8",
    "button_default_selected_text_color": "black",
    "dropdown_menu_item_hover_bg_color": "navy",
    "menu_background_color": "#004fb6",
    "button_success_selected_text_color": "black"
  }
}
```

次の要求は、Data Director アプリケーションのプライマリボタンのカラープロパティを更新します。

```
{
  "variables":{
     "primary_button_text_color": "white",
        "primary_button_bg_color": "green",
        "primary_button_border_color": "white",
        "primary_button_selected_bg_color": "pink",
        "primary_button_selected_border_color": "red",
        "primary_button_selected_text_color": "red"
    }
}
```

サンプル API 応答

指定されたカラープロパティが正常に更新されると、API は 200 OK 応答コードを返します。応答本文は空です。

デフォルトのカラースキーム

組織のニーズに応じて、Data Director アプリケーションのデフォルトのカラープロパティを変更できます。一部のプロパティは、特定の Data Director ユーザーインタフェースコンポーネントに影響します。ボタンのプロパティなどの他のプロパティは、複数のユーザーインタフェースコンポーネントに影響します。

ログインページ

次の表に、ログインページで変更できるプロパティを示します。

プロパティ	説明	デフォルト
login_form_text_color	ログインページのフィールドラベルのテキストの色。	#333333
login_form_background_color	ログインページの背景色。	#fff

ページヘッダー

次の表に、Data Director ページで変更できるプロパティを示します。

プロパティ	説明	デフォルト
header_text_color	ページヘッダーのテキストの色。	#fff
header_search_text_color	検索フィールドの左側のページヘッダーに表示されるビジネスエンティティの名前の色。	#000
header_background_color	ページヘッダーの背景色。	#000

ナビゲーションバー

次の表に、左側のナビゲーションバーで変更できるプロパティを示します。

プロパティ	説明	デフォルト
menu_text_color	ユーザーが選択したりカーソルを合わせたりしていないナビゲーションバー項目のテキストの色。	#fff
menu_sel_text_color	選択したナビゲーションバー項目のテキストの色。	#fff
menu_hover_text_color	ユーザーがカーソルを合わせたナビゲーションバー項目のテキストの色。	#fff
menu_background_color	ユーザーが選択したりカーソルを合わせたりしていないナビゲーションバー項目のテキストの背景色。	#333
menu_hover_bg_color	ユーザーがカーソルを合わせたナビゲーションバー項目の背景色。	#000
menu_sel_bg_color	選択したナビゲーションバー項目の背景色。	#000

ワークスペース

次の表に、ワークスペースまたはビジネスエンティティレコードビューで変更できるプロパティを示します。

プロパティ	説明	デフォルト
workspace_bg	ナビゲーションバーの右側に表示されるワークスペースの背景色。	#fff
entity_view_label_font_weight	読み取り専用ビューのフィールドラベルのフォントの太さ。	ノーマル

ドロップダウンリスト

次の表に、アプリケーションのドロップダウンリストで変更できるプロパティを示します。

プロパティ	説明	デフォルト
dropdown_menu_text_color	リストアイテムのテキストの色。	#000
dropdown_menu_bg_color	ドロップダウンリストの背景色。	#fff

プロパティ	説明	デフォルト
dropdown_menu_item_hover_text_color	ユーザーが選択またはカーソルを合わせたリスト項目のテキストの色。	#2f3342
dropdown_menu_item_hover_bg_color	ユーザーが選択またはカーソルを合わせたリスト項目の背景色。	#fafbfc

アプリケーションボタン

アプリケーションのボタンは、次のカテゴリに属しています。

- 1. 操作をキャンセルしたときにデータが失われる可能性のあるボタン。例えば、ビジネスエンティティレコードビューの**［編集のキャンセル］**ボタン。
- 2. データを送信または保存するボタン。例えば、ビジネスエンティティレコードビューの**［編集］**または**［保存］**ボタン。
- 3. ユーザーインタフェースで主要なアクションを実行するボタン。例えば、**［フォームに戻る］**ボタン。
- 4. デフォルトのボタン。ほとんどのページとダイアログボックスにあるボタン。例えば、**［OK］**ボタン。

次の表に、変更できるボタンのスタイルとプロパティを示します。

ボタンスタイル	プロパティ	説明	デフォルト
1	button_danger_text_color	ボタンのテキストの色。	#3df
1	button_danger_bg_color	ボタンの背景色。	#ddd
1	button_danger_border_color	ボタンの境界の色。	#2373e4
1	button_danger_selected_border_color	ユーザーが選択またはカーソルを合わせたボタンの境界の色。	#2373e4
1	button_danger_selected_text_color	ユーザーが選択またはカーソルを合わせたボタンのテキストの色。	#3df
1	button_danger_selected_bg_color	ユーザーが選択またはカーソルを合わせたボタンの背景色。	#ddd
2	button_success_text_color	ボタンのテキストの色。	#3df
2	button_success_bg_color	ボタンの背景色。	#ddd
2	button_success_border_color	ボタンの境界の色。	#2373e4
2	button_success_selected_text_color	ユーザーが選択またはカーソルを合わせたボタンのテキストの色。	#3df
2	button_success_selected_bg_color	ユーザーが選択またはカーソルを合わせたボタンの背景色。	#ddd

ボタンスタイル	プロパティ	説明	デフォルト
2	button_success_selected_border_color	ユーザーが選択またはカーソルを合わせたボタンの境界の色。	#2373e4
3	primary_button_text_color	ボタンのテキストの色。	#fff
3	primary_button_bg_color	ボタンの背景色。	#4b91ea
3	primary_button_border_color	ボタンの境界の色。	#3483e7
3	primary_button_selected_text_color	ユーザーが選択またはカーソルを合わせたボタンのテキストの色。	#fff
3	primary_button_selected_bg_color	ユーザーが選択またはカーソルを合わせたボタンの背景色。	#4b91ea
3	primary_button_selected_border_color	ユーザーが選択したりカーソルを合わせたりしていないボタンの境界の色。	#3483e7
4	button_default_text_color	ボタンのテキストの色。	#333
4	button_default_bg_color	ボタンの背景色。	#fff
4	button_default_border_color	ボタンの境界の色。	#eaeaea
4	button_default_selected_text_color	ユーザーが選択またはカーソルを合わせたボタンのテキストの色。	#333
4	button_default_selected_bg_color	ユーザーが選択またはカーソルを合わせたボタンの背景色。	#fff
4	button_default_selected_border_color	ユーザーが選択またはカーソルを合わせたボタンの境界の色。	#eaeaea

カラースキームの削除

カラースキームの削除 REST API は、カスタムカラープロパティを削除し、Data Director アプリケーションのデフォルトのカラープロパティを復元します。

この API は DELETE メソッドを使用します。

要求 URL

［カラースキームの削除］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/
```

［カラースキームの削除］URL に対して、次の HTTP DELETE 要求を行います。

```
DELETE http://<host>:<port>/cmx/ui/theme
```

サンプル API **要求**

次のサンプル要求は、カスタムカラープロパティを削除し、デフォルトのプロパティを復元します。

```
DELETE http://localhost:8080/cmx/ui/theme
```

要求の本文が空です。

サンプル API **応答**

API は、Data Director アプリケーションから現在のカラープロパティを正常に削除し、デフォルトのプロパティを復元した後、200 OK の応答コードを返します。応答本文は空です。

免責事項を設定するための REST API

REST API を使用して、Data Director、プロビジョニングツール、および Hub コンソールのログインページの免責事項を設定できます。例えば、アプリケーションの許可された使用法や、組織によって実施されているセキュリティポリシーについて説明できます。ユーザーは、ログインする前に免責事項を確認する必要があります。

次の表に、ログインページの免責事項を設定するための REST API を示します。

REST API	**説明**
リーガルメッセージ設定の読み取り	免責事項ファイルの情報を読み取ります。例えば、現在の免責事項を取得します。GET メソッドをサポートします。
リーガルメッセージ設定	免責事項ファイルを作成または更新します。例えば、現在の免責事項を更新します。POST メソッドをサポートします。
リーガルメッセージ設定の削除	免責事項ファイルを削除します。例えば、現在の免責事項を削除します。DELETE メソッドをサポートします。

免責事項のパラメータ

免責事項のテキスト、タイトル、およびユーザーがログインを試みるたびに免責事項を表示するかどうかを指定できます。ログインページに免責事項を表示するアプリケーションを指定することもできます。

次の表に、指定できる免責事項のパラメータを示します。

パラメータ	**説明**
messageContent	免責事項のテキスト。
messageTitle	免責事項のタイトル。
acceptLabel	ユーザーが免責事項に同意するためにクリックするボタンのテキスト。
rejectLabel	ユーザーが免責事項を拒否するためにクリックするボタンのテキスト。ユーザーは免責事項に同意せずにログインすることはできません。ユーザーが免責事項を拒否した場合、ログイン画面が再表示されます。

パラメータ	説明
showOnce	ユーザーが初めてログインしたときに免責事項が表示されるよう指定します。デフォルトは true です。ユーザーがログインを試みるたびに免責事項を表示するには、値を false に設定します。
enabledApplications	ユーザーがログインしようとしたときに免責事項を表示するアプリケーション。次の値を指定できます。 - e360。Data Director アプリケーションを指定します。 - provisioning。プロビジョニングツールを指定します。 - hubconsole。Hub コンソールを指定します。

リーガルメッセージ設定の読み取り

リーガルメッセージ設定の読み取り REST API は、現在のログインページの免責事項を削除します。

この API は GET メソッドを使用します。

要求 URL

［リーガルメッセージ設定の読み取り］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<path>
```

［リーガルメッセージ設定の読み取り］URL に対して、次の HTTP GET 要求を行います。

```
GET http://localhost:8080/cmx/ui/legal/config
```

サンプル API **要求**

次の要求は、現在のカラーテーマのプロパティを返します。

```
GET http://localhost:8080/cmx/ui/legal/config
Content-Type: application/json
```

サンプル API **応答**

現在のログインページの免責事項が正常に取得されると、API は 200 OK 応答コードを返します。

例えば、次の応答を受信します。

```
{
   "messageContent":"The use of this systems may be monitored and recorded. If you access this system,
you expressly consent to such monitoring and the use of the security policies enforced by the company.",
   "messageTitle":"Legal Disclaimer",
   "acceptLabel":"Accept",
   "rejectLabel":"Reject",
   "showOnce":true,
   "enabledApplications":[
      "e360",
      "provisioning",
      "hubconsole"
   ]
}
```

リーガルメッセージ設定

リーガルメッセージ設定 REST API は、ログインページの免責事項を作成または更新します。JSON 形式を使用して、要求本文でデータを送信します。

この API は POST メソッドを使用します。

要求 URL

［リーガルメッセージ設定］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<path>
```

［リーガルメッセージ設定］URL に対して、次の HTTP POST 要求を行います。

```
POST http://localhost:8080/cmx/ui/legal/config
```

サンプル API **要求**

次の要求本文には、サンプルの免責事項が含まれています。

```
{
    "messageContent": "The use of this systems may be monitored and recorded. If you access this system,
you expressly consent to such monitoring and the use of the security policies enforced by the company.",
    "messageTitle": "Legal Disclaimer",
    "acceptLabel": "Accept",
    "rejectLabel": "Reject",
    "showOnce": true,
    "enabledApplications": [
        "e360",
        "provisioning",
        "hubconsole"
    ]
}
```

サンプル API **応答**

ログインページの免責事項が正常に作成または更新されると、API は 200 OK 応答コードを返します。応答本文は空です。

リーガルメッセージ設定の削除

リーガルメッセージ設定の削除 REST API は、ログインページの免責事項の設定を削除します。

この API は DELETE メソッドを使用します。

要求 URL

［リーガルメッセージ設定の削除］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<path>
```

［リーガルメッセージ設定の削除］URL に対して、次の HTTP DELETE 要求を行います。

```
DELETE http://<host>:<port>/cmx/ui/legal/config
```

サンプル API **要求**

次のサンプル要求は、ログインページの免責事項の設定を削除します。

```
DELETE http://localhost:8080/cmx/ui/legal/config
```

サンプル API **応答**

ユーザーは、標準のログイン手順を使用してログインできます。ログインページの免責事項の設定が正常に削除されると、API は 200 OK 応答コードを返します。応答本文は空です。

第 5 章

SOAP ビジネスエンティティサービス呼び出し

この章では、以下の項目について説明します。

SOAP ビジネスエンティティサービス呼び出しの概要

Simple Object Access Protocol（SOAP）エンドポイント呼び出しを行うと、すべてのビジネスエンティティサービスが Web サービスとして使用できるようになります。SOAP 呼び出しを介して、ビジネスエンティティ内にレコードを作成したり、ビジネスエンティティ内のレコードを削除、更新、検索したりできます。レコードのマージ、マージ解除、照合などの操作も実行できます。SOAP 呼び出しは、タスクの作成、更新、検索、実行にも利用できます。

ビジネスエンティティサービスの SOAP エンドポイントは、Web Services Security（WS-Security）UsernameToken を使用してユーザーを認証します。

注: SOAP API を使用してビジネスエンティティサービスを呼び出す前に、オペレーショナル参照ストアを検証します。

認証方法

ビジネスエンティティサービスへのすべての SOAP 呼び出しはユーザー認証を必要とします。Web サービス要求の SOAP メッセージヘッダーにユーザー名とパスワードを指定します。

SOAP ヘッダー要素 Security には、セキュリティ関連のデータが含まれています。Security 要素には、以下の子要素を持つ UsernameToken 要素が含まれています。

username

トークンに関連付けられたユーザー名。

password

トークンに関連付けられたユーザー名に対するパスワード。

UsernameToken 要素にユーザー名とパスワードを指定して送信します。

次の例は、SOAP メッセージの Security ヘッダー要素を示しています。

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:cs-
ors.informatica.mdm">
   <soapenv:Header>
   <Security xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
    <UsernameToken>
      <Username>admin</Username>
      <Password>admin</Password>
    </UsernameToken>
  </Security>
  </soapenv:Header>
<soapenv:Body>
.....
</soapenv:Body>
</soapenv:Envelope>
```

サードパーティアプリケーションからログインする場合の認証クッキー

認証 Cookie を使用して MDM Hub ユーザーを認証し、Postman などのサードパーティアプリケーションからビジネスエンティティサービスを呼び出します。認証 Cookie を使用すると、ユーザー名とパスワードをハードコーディングする必要がありません。

認証 Cookie を使用するプロセスには、次の手順が含まれます。

1. 認証されたユーザーの資格情報を使用して、認証 Cookie を取得します。
2. 認証 Cookie を保存します。
3. 認証 Cookie を使用して SOAP API を呼び出します。

すべての MDM セッションは、Informatica CSRF トークン（ICT）を使用します。ICT は HTTP ヘッダーに表示されます。

ユーザーがセッション ID を要求すると、応答は HTTP ヘッダーの一部として ICT を返します。外部アプリケーションの場合、外部アプリケーションが使用できる認証用の URL があります。この URL は、ヘッダーにある ICT と認証 Cookie を返します。認証 Cookie は MDM セッション ID です。

以降のビジネスエンティティサービスの呼び出しでは、呼び出しに認証 Cookie と ICT が含まれている必要があります。後続の要求では、セッションが期限切れになるまで ICT および MDM セッション ID を使用できます。

認証 Cookie と Informatica CSRF トークンの取得

次の例は、認証 Cookie と Informatica CSRF トークン（ICT）を取得する方法を示しています。

```
POST: https://<host>:<port>/cmx/auth/<database ID>

Authorization:   No Auth

Body:
{
    "username" : "admin",
    "password" : {
        "encrypted" : false,
        "password" : "admin"
    },
    "userInfo" : {
        "sessionTimeout":300,
        "role":"Manager"
    }
}
```

次の例は、ICT がヘッダーにどのように表示されるかを示しています。

```
ICT:  0ffa3b95c67e111b9c14c140a70273a15db266993d0c763d555135de4c4d6110
```

次の例は、認証 Cookie が応答にどのように表示されるかを示しています。

```
Name: mdmsessionid
Value: vppOTOhJLKMjYJf7Diil
Domain: 10.xx.xx.xx
Path: /cmx
Expires: Session
HttpOnly: true
Secure: false
```

認証 Cookie と Informatica CSRF トークンを使用した SOAP の例

次の例は、API 要求ヘッダーで認証 Cookie と Informatica CSRF トークン（ICT）を使用する方法を示しています。

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:cs-
ors.informatica.mdm">
  <soapenv:Header/>
   <soapenv:Body>
      <urn:listAssignableUsers>
         <!--Optional:-->
         <urn:parameters>
            <!--Optional:-->
            <urn:taskType>Update</urn:taskType>
            <!--Optional:-->
            <urn:businessEntity>Person</urn:businessEntity>
         </urn:parameters>
      </urn:listAssignableUsers>
   </soapenv:Body>
</soapenv:Envelope>

Headers:
Cookie:  mdmsessionid: vppOTOhJLKMjYJf7Diil
ICT:  0ffa3b95c67e111b9c14c140a70273a15db266993d0c763d555135de4c4d6110
```

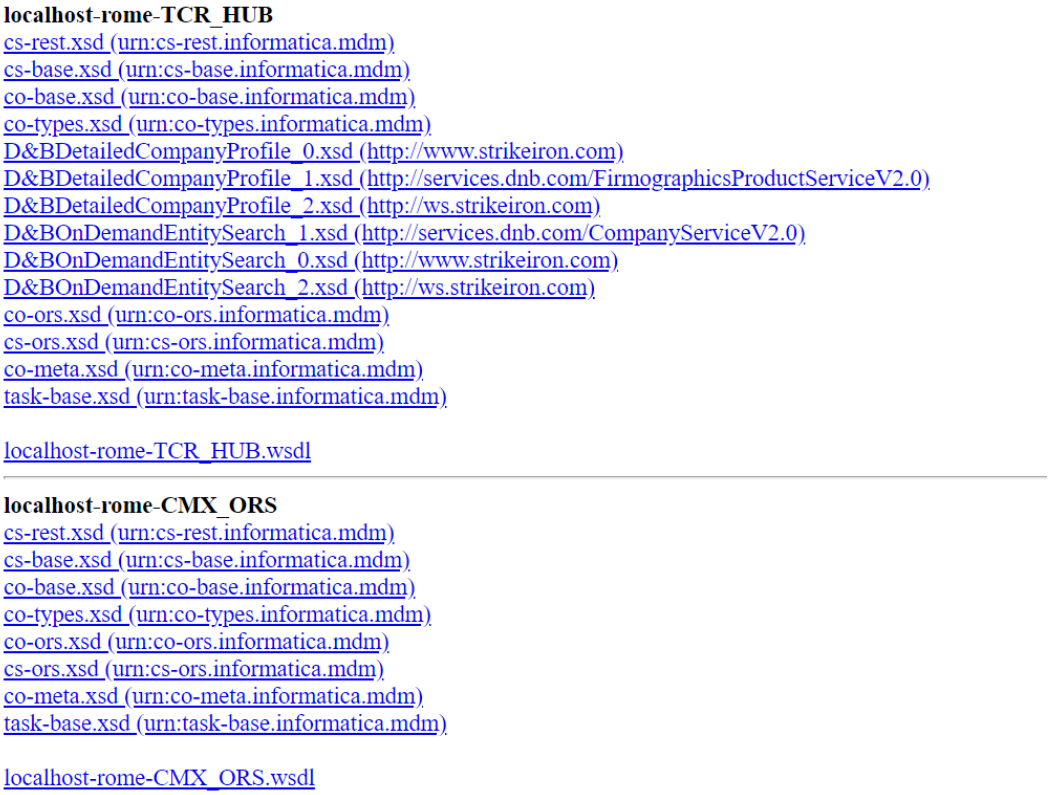
Web サービス記述言語ファイル

Web サービス記述言語（WSDL）ファイルには、Web サービス、SOAP 要求および応答の形式、およびすべてのパラメータの XML 記述が含まれています。MDM Hub は、オペレーショナル参照ストアごとに WSDL ファイルを生成します。

各オペレーショナル参照ストアの WSDL ファイルは次の場所にあります。

`http:://<host: ホスト>:<port: ポート>/cmx/csfiles`

次の図は、オペレーショナル参照ストアの WSDL ファイルをダウンロードできる場所を示しています。



利用可能なオペレーショナルリファレンスストアの WSDL ファイルをダウンロードするには、リンクをクリックします。

SOAP URL

SOAP URL は、SOAP サーバーへの接続に使用するアドレスです。

SOAP URL の構文は次のとおりです。

`http://<host: ホスト>:<port: ポート>/<context: コンテキスト>/<database ID: データベース ID>`

この URL には以下のフィールドがあります。

host

データベースを実行するホスト。

ポート

データベースリスナが使用するポート番号。

context

context は常に cmx/services/BEServices です。

データベース ID

Hub コンソールのデータベースツールで登録された ORS の ID。

次のサンプルは SOAP URL を示しています。

```
http://localhost:8080/cmx/services/BEServices/localhost-orcl-DS_UI1
```

SOAP 要求と SOAP 応答

SOAP クライアントを介してビジネスエンティティサービスに要求を送信する場合と、クライアントでビジネスエンティティサービスから応答を受け取る場合は、SOAP XML メッセージ形式を使用します。SOAP 要求の形式と SOAP 応答の形式は同じです。

SOAP メッセージには次の要素が含まれます。

エンベロープ

メッセージの開始と終了を定義します。

ヘッダー

オプション。付加的な属性（メッセージを処理するための認証詳細など）が含まれます。ヘッダー要素を含める場合は、この要素をエンベロープ要素の最初の子要素として含める必要があります。

本文

クライアントまたは Web サービスが処理する XML データが含まれます。

SOAP メッセージの形式は次のとおりです。

```
<?xml version="1.0"?>
<env:Envelope  xmlns:env="http://schemas.xmlsoap.org/soap/envelope/" >

<env:Header>
</env:Header>

<env:Body>
</env:Body>

</env:Envelope>
```

SOAP 要求の形式は次のとおりです。

```
POST /<host>:<port>/<context>/<database ID> HTTP/1.0
Content-Type: text/xml; charset=utf-8

<?xml version="1.0"?>
<env:Envelope  xmlns:env="http://schemas.xmlsoap.org/soap/envelope/" >

<env:Header>
</env:Header>

<env:Body>
</env:Body>

</env:Envelope>
```

SOAP 応答の形式は次のとおりです。

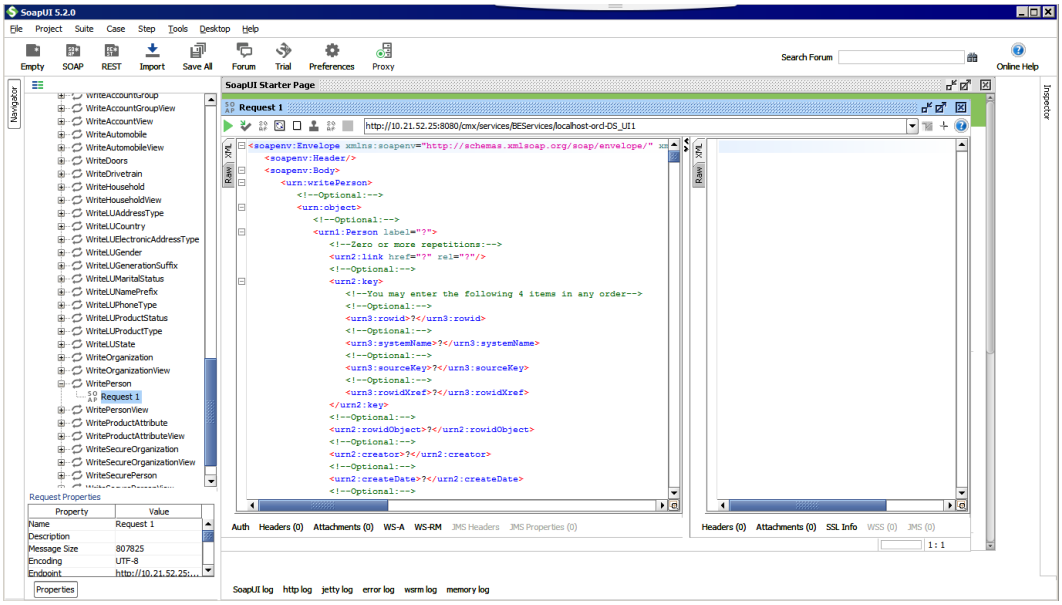
```
HTTP/1.0 200 OK
Content-Type: text/xml; charset=utf-8

<?xml version="1.0"?>
<env:Envelope  xmlns:env="http://schemas.xmlsoap.org/soap/envelope/" >

<env:Header>
</env:Header>

<env:Body>
</env:Body>

</env:Envelope>
```

入力パラメータと出力パラメータの表示

SoapUI などの有用なテストツールを使用し、SOAP API の入力パラメータと出力パラメータを確認できます。

SOAP プロジェクトを作成し、そのプロジェクトに WSDL ファイルをインポートします。SoapUI では、ビジネスエンティティサービスを使用して実行できる操作がノードとして表示されます。各操作には、要求メッセージ形式と応答メッセージ形式があります。SoapUI は、WSDL ファイルがインポートされた時点で、各操作のサンプル要求を作成します。

プロジェクトを開き、要求をダブルクリックして要求エディタを開きます。次の図は、SoapUI に表示された、WritePerson SOAP API の入力パラメータを示しています。



SOAP API リファレンス

このセクション「ビジネスエンティティサービス用の SOAP API リファレンス」では、SOAP API をリストアップし、各 API について説明しています。ビジネスエンティティサービスの説明については、WSDL ファイルも参照してください。

ビジネスエンティティの操作には次の SOAP API を使用します。

メタデータの取得

ビジネスエンティティのデータ構造を返します。

レコードの一覧表示

外部キー値のルックアップ値のリストを返します。

レコードの読み取り

ビジネスエンティティ内のルートレコードの詳細を返します。

レコードの作成

指定されたビジネスエンティティ内にレコードを作成します。

レコードの更新

指定されたルートレコードとその子レコードを更新します。

レコードの削除

ビジネスエンティティ内のルートレコードを削除します。

レコードの検索

検索可能なルートビジネスエンティティ内で文字列値を検索し、その検索条件に一致するルートレコードを返します。

昇格のプレビュー

変更要求の相互作用 ID に基づいて保留中の変更の昇格が行われた場合、結果として生成されるレコードのプレビューを返します。

昇格

レコードの保留中の変更をすべて、変更要求の相互作用 ID に基づいて昇格させます。

昇格の削除

レコードの保留中の変更をすべて、変更要求の相互作用 ID に基づいて削除します。

マージのプレビュー

2 つ以上のルートレコードをマージすると、この API は統合されたルートレコードのプレビューを返します。

レコードのマージ

2 つ以上のルートレコードをマージして単一の統合されたレコードを作成します。

レコードのマージ解除

ルートレコードのマージを解除し、レコードがマージされる前に存在した個々のルートレコードの状態にします。

関連するレコードの取得

階層マネージャで設定されるリレーションに基づいて、関連するレコードのリストを返します。

一致するレコードの読み取り

指定されたルートレコードに一致するレコードを返します。

一致するレコードの更新

マッチテーブルのレコードを作成または更新します。

一致レコードの削除

一致するレコードをマッチテーブルから削除します。

BPM **メタデータの取得**

タスクタイプと、ワークフローアダプタが Get Task Lineage サービスと管理サービスを実行できるかどうかを示す 2 つのインジケータを返します。

タスクの一覧表示

ワークフロータスクのリストを返します。

タスクの読み取り

タスクの詳細を返します。

タスクの作成

タスクを作成してワークフローを開始します。

タスクの更新

1 つのタスクを更新します。

タスクアクションの実行

タスクアクションを 1 つ実行し、さらに処理ができるようにそのタスクをワークフローに戻します。

割り当て可能なユーザーの一覧表示

特定のタスクタイプに属するタスクの割り当て対象として指定できるユーザーのリストを返します。

タスクの完了

ワークフロー内のすべてのタスクを完了した後、タスクワークフローを閉じます。

サンプル SOAP 要求とサンプル SOAP 応答

SOAP API を理解するには、次のサンプル SOAP 要求とサンプル SOAP 応答を確認してください。

割り当て可能なユーザーの一覧表示のサンプル

次のサンプル SOAP 要求は、割り当て可能なユーザーのリストを取得します。

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:cs-
ors.informatica.mdm">
   <soapenv:Header>
   <Security xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
    <UsernameToken>
      <Username>admin</Username>
      <Password>admin</Password>
    </UsernameToken>
  </Security>
  </soapenv:Header>
   <soapenv:Body>
      <urn:listAssignableUsers>
         <!--Optional:-->
         <urn:parameters>
            <!--Optional:-->
            <urn:taskType>Update</urn:taskType>
            <!--Optional:-->
```

```
            <urn:businessEntity>Person</urn:businessEntity>
         </urn:parameters>
      </urn:listAssignableUsers>
   </soapenv:Body>
</soapenv:Envelope>
```

次のサンプル SOAP 応答には、割り当て可能なユーザーのリストが含まれます。

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/
2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
      <ns6:listAssignableUsersReturn xmlns:ns1="urn:cs-base.informatica.mdm" xmlns:ns2="urn:co-
base.informatica.mdm" xmlns:ns3="urn:co-ors.informatica.mdm" xmlns:ns4="urn:co-meta.informatica.mdm"
xmlns:ns5="urn:task-base.informatica.mdm" xmlns:ns6="urn:cs-ors.informatica.mdm">
         <ns6:object>
            <ns1:users>
                <user>
                 <userName>admin</userName>
                </user>
            </users>
            <ns1:roles/>
         </ns6:object>
      </ns6:listAssignableUsersReturn>
   </soapenv:Body>
</soapenv:Envelope>
```

ビジネスエンティティの読み取りサンプル

次のサンプル SOAP 要求は、Person ビジネスエンティティの情報を取得します。

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:cs-
ors.informatica.mdm" xmlns:urn1="urn:cs-base.informatica.mdm">
   <soapenv:Header/>
   <soapenv:Body>
      <urn:readEntity>
         <urn:parameters>
            <urn:coFilter>
               <urn1:object name="Person" depth="3" suppress="false">
                  <urn1:key>
                     <urn1:rowid>931</urn1:rowid>
                  </urn1:key>
                  <urn1:fields>firstName,lastName,middleName,gender</urn1:fields>
                  <urn1:recordState>ACTIVE</urn1:recordState>
                  <urn1:recordState>PENDING</urn1:recordState>
                  <urn1:contentMetadata>XREF</urn1:contentMetadata>
               </urn1:object>
            </urn:coFilter>

            <urn:outputFilter>
               <urn1:object name="PersonView" depth="2" suppress="false">
               </urn1:object>
            </urn:outputFilter>
         </urn:parameters>
      </urn:readEntity>
   </soapenv:Body>
</soapenv:Envelope>
```

ビジネスエンティティの作成サンプル

次のサンプル SOAP 要求は、Person ビジネスエンティティを作成します。

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:cs-
ors.informatica.mdm" xmlns:urn1="urn:co-base.informatica.mdm" xmlns:urn2="urn:cs-base.informatica.mdm"
xmlns:urn3="urn:co-ors.informatica.mdm" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Header/>
    <soapenv:Body>
        <urn:writeEntity>
            <urn:object xmlns:urn3="urn:co-ors.informatica.mdm" xsi:type="urn3:Person">
```

```
                <urn3:Person label="Person">
                    <urn3:partyType>Person</urn3:partyType>
                    <urn3:lastName>test1</urn3:lastName>
                    <urn3:middleName>test1</urn3:middleName>
                    <urn3:firstName>test1</urn3:firstName>
                </urn3:Person>
                <urn3:changeSummary create="#/urn:writeEntity/urn:object/urn1:Person" logging="false"
xmlns:sdo="commonj.sdo">
                    <urn:object sdo:ref="#/urn:writeEntity/urn:object" sdo:unset="Person"/>
                </urn3:changeSummary>
            </urn:object>
            <urn:parameters>
                <urn:systemName>Admin</urn:systemName>
            </urn:parameters>
        </urn:writeEntity>
    </soapenv:Body>
</soapenv:Envelope>
```

第 6 章

相互参照レコードと BVT 計算サービス

この章では、以下の項目について説明します。

相互参照レコードと BVT 計算サービスの概要

相互参照レコードおよびベストバージョンオブトゥルース（BVT）計算用のサービスを使用して、ソースデータがマスタレコードを形成する方法を学ぶことができます。

これらのサービスを使用して、次のタスクを実行できます。

- ソースデータについての情報を収集する
- ベストバージョンオブトゥルースが決定された方法を特定する
- BVT 計算をオーバーライドして、マスタレコードにベストバージョンオブトゥルースが含まれるようにする

相互参照データの取得および BVT 計算の調査

MDM Hub のマスタレコードではベストバージョンオブトゥルース（BVT）が保持されます。MDM Hub は複数のソースシステムから最も信頼性の高いデータを各マスタレコードに統合し、ベストバージョンオブトゥルースを確立します。MDM Hub には相互参照レコードのソースデータが保存されます。ビジネスエンティティサービスを使用して、相互参照レコードからデータを読み取って、BVT の計算方法を決定できます。

相互参照レコードの取得

ビジネスエンティティサービスを使用して、特定のマスタレコードの相互参照レコードを取得できます。

相互参照レコードを取得するための REST API URL の形式は、次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<row ID>?contentMetadata=XREF
```

次のサンプル要求は、行 ID 123 を持つ Person ビジネスエンティティレコードの相互参照レコードを取得します。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-ORS/Person/123?contentMetadata=XREF
```

.

相互参照レコード応答の取得

次の例は、行 ID 123 を持つ Person レコードについて返された相互参照レコードを示しています。

```
GET /Person/123?contentMetadata=XREF
```

```
{
  "firstName": "Joe",
  "lastName": "Smith",
  "XREF": {
    "item": [
      {
        "rowidXref": 111,
        "firstName": "Joe",
        "lastName": "Smith",
      },
      {
        "rowidXref": 222,
        "firstName": "John",
        "lastName": "Smith"
      }
    ]
  }
}
```

マスタレコードへのコントリビュータの特定

ビジネスエンティティサービスを使用して、マスタレコードにデータを提供した相互参照レコードフィールドを確認できます。各フィールドにデータを提供したレコードは、相互参照レコードの行 ID によって識別されます。

マスタレコードへのコントリビュータを特定する REST API URL の形式は、次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<row ID>?contentMetadata=BVT
```

次のサンプル要求は、行 ID 123 を持つ Person レコードの BVT 情報を取得します。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-ORS/Person/123?contentMetadata=BVT
```

マスタレコード応答へのコントリビュータの特定

次の例に、マスタレコードの各フィールドにデータを提供した相互参照レコードを示します。

```
{
  "firstName": "Joe",
  "lastName": "Smith",
  "BVT": {
    "firstName": {
      "rowidXref": 111
    },
    "lastName": {
      "rowidXref": 222
    }
  },
}
```

提供元の相互参照レコードフィールドの信頼スコアの取得

ビジネスエンティティサービスを使用して、マスタレコードにデータを提供する相互参照レコードフィールドの信頼スコアを取得します。

コントリビュータを特定して信頼スコアを取得する REST API URL の形式は、次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<row ID>?contentMetadata=TRUST
```

次のサンプル要求は、行 ID 123 を持つ Person レコードの信頼スコアを提供します。

```
GET http://localhost:8080/cmx/cs/ors/Person/123?contentMetadata=TRUST
```

提供元の相互参照レコードフィールド応答の信頼スコアの取得

次の応答例では、Person ビジネスエンティティレコードの各フィールドの信頼スコアを示します。

```
{
  "firstName": "John",
  "lastName": "Smith",
  "TRUST": {
    "firstName": {
      "score": 75.0,
     "valid": true,
     "trustSetting" :{
         // custom settings
     }
    },
  },
}
```

すべての相互参照レコードフィールドの信頼スコアの取得

XREF_TRUST に設定された contentMetadata パラメータとともに REST API を使用して、すべての相互参照レコードフィールドの信頼スコアとダウンロード率を取得します。

コントリビュータを特定して信頼スコアを取得する、［読み取り］REST API の要求 URL:

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<row ID>?contentMetadata=XREF_TRUST
```

次のサンプル要求は、行 ID 123 を持つ Person レコードの相互参照データを取得します。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-ORS/Person/123?contentMetadata=XREF_TRUST
```

すべての相互参照レコードフィールド応答の信頼スコアの取得

次の例は、Person ビジネスエンティティのすべての相互参照レコードフィールドの信頼スコアとダウンロード率を示します。

```
{
    "firstName": "Sergey",
    "lastName": "Ivanov",
    "XREF": {
        "item": [{
            "rowidXref": 111,
            "firstName": "Sergey",
            "lastName": "Petrov",
            "TRUST": {
                "firstName": {
                    "score": 75.0,
                    "valid": true
                },
                "lastName": {
                    "score": 60.0,
                    "valid": false,
                    "downgradePerCent": 20.0
                }
```

```
            }
        }, {
            "rowidXref": 222,
            "firstName": "Sergey",
            "lastName": "Ivanov",
            "TRUST": {
                "firstName": {
                    "score": 10.0,
                    "valid": true
                },
                "lastName": {
                    "score": 80.0,
                    "valid": true
                }
            }
        }]
    }
}
```

ソースシステムについての情報の取得

相互参照データの取得元ソースシステムの情報、およびレコード全体に対してソースシステムが提供した相互参照レコード数を、ノードごとおよびレコードごとに取得できます。

要求では、次のパラメータを指定できます。

describe

true に設定すると、ソースシステムの説明が返されます。true または false を指定できます。デフォルトは false です。

aggregate

ソースシステム情報を返すレベルを指定します。ENTITY、NODE、および RECORD を指定できます。デフォルトは ENTITY です。

recordStates

レコードを返すレコードの状態を指定します。ACTIVE、PENDING、または DELETED を指定できます。デフォルトは ACTIVE です。

compact

no に設定すると、aggregate パラメータが ENTITY および他の集計レベルに指定されている場合、エンティティレベルデータが返されることを指定します。yes または no を指定できます。REST API 要求のみで使用できます。デフォルトは yes です。

ソースシステムについての情報の取得例

次のサンプル要求は、行 ID 123 を持つ Person ビジネスエンティティのエンティティレベルとノードレベルのソースシステム情報を取得します。

```
GET http://localhost:8080/cmx/cs/ors/Person/123?action=getSourceSystems&aggregate=ENTITY,NODE&compact=no
```

次のサンプル要求は、行 ID 456 を持つ Person ビジネスエンティティの、レコードレベルのソースシステム情報とソースシステムの説明を取得します。

```
GET http://localhost:8080/cmx/cs/ors/Person/123/Address/456?
action=getSourceSystems&aggregate=ENTITY,NODE&compact=no
```

ソースシステム応答についての情報の取得

次の例は、Person ビジネスエンティティのエンティティレベルとノードレベルの情報を示しています。

```
{
      "name": "Admin",
```

```
      "xrefCount": 120
    }
  ],
  Person: {
    "rowidObject": "456",
    "sourceSystem":
      {
        "name": "Admin",
        "xrefCount": 30
      }
    ]
  }
}
```

応答のフィルタリングおよびページ区切り

応答で返すフィールドを選択し、複数の条件で結果をフィルタリングし、結果をページ区切りできます。

フィルタリング要求の例

次のテーブルに、さまざまなフィルタが適用された Person ビジネスエンティティに対するサンプル要求と、応答で返された結果についての説明を示します。

要求	返された結果についての説明
/Person/123	すべてのユーザー定義フィールド
/Person/123?readSystemFields=true	すべてのユーザー定義フィールドとすべてのシステムフィールド
/Person/123?fields=firstName	1 つのユーザー定義フィールド
/Person/123?fields=updatedBy	1 つのシステムフィールド
Person/123?fields=firstName,updatedBy	1 つのユーザー定義フィールドと 1 つのシステムフィールド
/Person/123? fields=firstName&readsystemFields=true	1 つのユーザー定義フィールドとすべてのシステムフィールド

ベストバージョンオブトゥルースの確立

相互参照レコードのソースデータを調べた後、データスチュワードはマスタレコードがベストバージョンオブトゥルースを表すように、ソースデータの統合方法を調整できます。

ビジネスエンティティサービスを使用して次のアクションを実行し、ベストバージョンオブトゥルースを確立できます。

- 信頼設定を更新する
- 一致しないソースデータを削除する

- 正しい提供元フィールドを選択する
- マスタレコードに正しい値を書き込む

正しい提供元フィールドの選択

信頼スコアが最も高いフィールドにベストバージョンオブトゥルースが含まれていない場合、データスチュワードはマスタレコードにデータを提供する正しいデータが含まれるフィールドを選択できます。

システム名とソースキーに基づいて正しい提供元フィールドを選択する URL と要求本文の形式は、次のとおりです。

```
POST http://<host>:<port>/<context>/<database ID>/<business entity>/<row ID>?systemName=<source system name>
{
    BVT: {
        <field name>: {
            systemName: "<source system name>",
            sourceKey: "<source key>"
        }
    }
}
```

相互参照レコード ID に基づいて正しい提供元フィールドを選択する URL と要求本文の形式は、次のとおりです。

```
POST http://<host>:<port>/<context>/<database ID>/<business entity>/<row ID>?systemName=<source system name>
{
    BVT: {
        <field name>: {
            rowidXref: "<row ID>"
        }
    }
}
```

正しい提供元フィールドの選択例

次の URL および要求本文は、ソースキー 0001 の Sales ソースシステムから相互参照レコードの名フィールドを選択して、マスタレコードにデータを提供します。

```
POST http://localhost:8080/cmx/cs/localhost-orcl-ORS/Person/123?systemName=Admin
{
    BVT: {
        firstName: {
            systemName: "Sales",
            sourceKey: "0001"
        }
    }
}
```

次の URL および要求本文は、行 ID 789 の相互参照レコードの名フィールドを選択して、マスタレコードにデータを提供します。

```
POST http://localhost:8080/cmx/cs/localhost-orcl-ORS/Person/123?systemName=Admin
{
    BVT: {
        firstName: {
            rowidXref: "789"
        }
    }
}
```

マスタレコードに正しい値を書き込む

ビジネスエンティティサービス呼び出しを使用して、正しい値をマスタレコードの書き込む際、値の信頼設定を指定することもできます。信頼設定を指定しない場合、MDM Hub は管理者システム設定を使用します。

管理者信頼設定で正しい値を書き込むための URL と要求本文の形式は、次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<row ID>?systemName=<source system providing
the correct value>{
    "<field name>": "<correct value>",
     "$original": {
        "<field name>": "<current value>",

    },
    "TRUST": {
        "<field name>": {
            "trustSetting" : {
                custom: false
            }
        }
    }
}
```

定義済みの信頼設定で正しい値を書き込むための URL と要求本文の形式は、次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<row ID>?systemName=<source system providing
the correct value>{
    "<field name>": "<correct value>",
     "$original": {
        "<field name>": "<current value>",

    },
    "TRUST": {
        "firstName": {
            "trustSetting" : {
                custom: true, // if custom=true, all other trustSetting fields
                              //are mandatory. If they are not set,
                              //the service will return an error.
                minimumTrust: <minimum trust percent>,
                maximumTrust: <maximum trust percent>,
                timeUnit: "<units for measuring trust decay>",
                maximumTimeUnits: <number of units>,
                graphType: "<name of graph type>"
            }
        }
    }
}
```

信頼パラメータ

次の信頼パラメータを定義できます。

minimumTrust

データ値が（減衰期間の経過後に）「古く」なったときに移行する信頼レベル。この値は最大信頼度以下である必要があります。

注: 最大信頼度と最小信頼度が同じ場合、減衰曲線は平らになり、減衰期間と減衰タイプは影響しなくなります。

maximumTrust

データ値が変更された場合の信頼レベル。例えば、ソースシステム X で電話番号フィールドが 555-1234 から 555-4321 に変更された場合、新しい値では、電話番号フィールドに対してシステム X の最大信頼度レベルが与えられます。最大信頼度レベルを高く設定することによって、ソースシステムにおける変更がベースオブジェクトに適用されるようにすることができます。

timeUnit

減衰期間の計算に使用する単位（日、週、月、四半期、または年）を指定します。

maximumTimeUnits

減衰期間の計算に使用する（日、週、月、四半期、または年の）数を指定します。

graphType

減衰は、減衰期間中に信頼レベルが低下するパターンをたどります。グラフタイプには、次のいずれかの減衰パターンを指定できます。

グラフタイプパラメータ	説明
LINEAR	最も単純な減衰。 減衰は、最大信頼度から最小信頼度への直線をたどります。
RISL	低下のほとんどが減衰期間の最初に発生します。 減衰は凹曲線をたどります。 ソースシステムがこのグラフタイプである場合、システムからの新しい値はおそらく信頼されますが、この値は上書きされる可能性があります。
SIRL	低下のほとんどが減衰期間の最後に発生します。 減衰は凸曲線をたどります。 ソースシステムがこのグラフタイプである場合、値が減衰期間の最後に近付くまで、他のシステムがこの値をマスタレコードで上書きする可能性は比較的低くなります。

マスタレコードに正しい値を書き込む例

例 1

マスタレコード内の名前を Sam Brown から John Smith に変更するとします。変更は Sales ソースシステムによるものです。信頼設定は、管理者信頼設定に設定されます。

次のコードは、例 1 の URL とコマンドを示しています。

```
POST http://localhost:8080/cmx/cs/localhost-orcl-ORS/Person/123?systemName=Sales
{
    "firstName": "John",
    "lastName": "Smith"
    "$original": {
        "firstName": "Sam",
        "lastName": "Brown"
    },
    "TRUST": {
        "firstName": {
            "trustSetting" : {
                custom: false
            }
            {
        "lastName": {
            "trustSetting" : {
                custom: false
            }
        }
    }
}
```

例 2

マスタレコード内の名前を Sam Brown から John Smith に変更するとします。変更は SFA ソースシステムによるものです。信頼設定は最小信頼度 60 および最大信頼度 90 に設定され、信頼は 3 か月の減衰期間にわたって直線的に減衰します。

次のコードは、例 2 の URL とコマンドを示しています。

```
POST http://localhost:8080/cmx/cs/localhost-orcl-ORS/Person/123?systemName=SFA
{
    "firstName": "John",
    "lastName": "Smith"
    "$original": {
        "firstName": "Sam",
        "lastName": "Brown"
    },
    "TRUST": {
        "firstName": {
            "trustSetting" : {
                custom: true,
                minimumTrust: 60,
                maximumTrust: 90,
                timeUnit: "Month",
                maximumTimeUnits: 3,
                graphType: "LINEAR"
            }
            {
        "lastName": {
            "trustSetting" : {
                custom: true,
                minimumTrust: 60,
                maximumTrust: 90,
                timeUnit: "Month",
                maximumTimeUnits: 3,
                graphType: "LINEAR"
            }
        }
    }
}
```

一致しないソースデータの削除

相互参照レコードを特定のマスタレコードに不適切に関連付けられている場合、データスチュワードはその相互参照レコードをマージ解除できます。マージ解除した相互参照レコードから新しいマスタレコードが作成されます。

マージ解除呼び出しでは、1 つの相互参照レコードのみマージ解除できます。複数の相互参照レコードをマージ解除する必要がある場合は、相互参照レコードごとにマージ解除呼び出しを行います。

マージ解除イベント用のトリガが構成されている場合、マージ解除タスクが作成されます。そうでない場合、相互参照レコードはマージ解除されます。

システム名とソースキーに基づいてレコードをマージ解除する URL およびコマンドの形式は、次のとおりです。

```
POST http://<host>:<port>/<context>/<database ID>/<business entity>/<row ID>?action=unmerge&systemName=<source
system name>
{
name: "<object name>",
key: {rowid: "<rowid value>", sourcekey: "<source key>", systemName: "<source system name>" }
}
```

相互参照レコード ID に基づいてレコードをマージ解除する URL およびコマンドの形式は、次のとおりです。

```
POST http://<host>:<port>/<context>/<database ID>/<business entity>/<row ID>?action=unmerge&systemName=<source
system name>
{
name: "<object name>",
key: {rowid: "<rowid value>", rowidXref: "<row ID of xref>"}
}
```

一致しないソースデータの削除例

REST API **の例**

次のコードは、住所レコードから子レベルで相互参照レコードをマージ解除する URL とコマンドを示しています。

```
POST http://localhost:8080/cmx/cs/localhost-orcl-MDM_SAMPLE/Person/181921?action=unmerge&systemName=Admin
{
"name":"Person.Address",
"key":{
"rowid":"41721 ",
"rowidXref":41722
}
}
```

説明：

- マージ解除する相互参照レコードの行 ID は 41722 です
- 相互参照レコードをマージ解除するマスタレコードの行 ID は 41721 です
- ルートレコードの行 ID は 181921 です

SOAP/EJB **の例**

次のコードは、住所レコードから子レベルで相互参照レコードをマージ解除する URL とコマンドを示しています。

```
<ns9:UnMerge xmlns:ns2="urn:co-base.informatica.mdm" xmlns:ns7="urn:co-meta.informatica.mdm"
xmlns:ns3="http://services.dnb.com/LinkageServiceV2.0" xmlns:ns8="urn:task-base.informatica.mdm"
xmlns:ns6="urn:co-ors.informatica.mdm" xmlns:ns1="urn:cs-base.informatica.mdm" xmlns:ns9="urn:cs-
ors.informatica.mdm" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="ns9:UnMerge">
   <ns9:parameters>
      <ns9:businessEntityKey name="Person">
         <ns1:key>
            <ns1:rowid>181921</ns1:rowid>
        </ns1:key>
      </ns9:businessEntityKey>
      <ns9:unmergeKey name="Person.TelephoneNumbers">
         <ns1:key>
            <ns1:rowid>41721 </ns1:rowid>
            <ns1:rowidXref>41722</ns1:rowidXref>
         </ns1:key>
      </ns9:unmergeKey>
      <ns9:treeUnmerge>true</ns9:treeUnmerge>
   </ns9:parameters>
</ns9:UnMerge>
```

説明：

- マージ解除する相互参照レコードの行 ID は 41722 です
- 相互参照レコードをマージ解除するマスタレコードの行 ID は 41721 です
- ルートレコードの行 ID は 181921 です

マージ解除応答

マージ解除応答には、マージ解除された相互参照レコードから作成されたベースオブジェクトの行 ID が含まれます。

応答例 1

次の例は、Person ルートノードから相互参照レコードをマージ解除するときの応答を示しています。

```
{
    Person: {
        rowidObject: "7777"
```

```
    }
}
```

応答例 2

次の例は、Address 子ノードから相互参照レコードをマージ解除するときの応答を示しています。

```
{
    Person: {
        Address: {
            item: [
                rowidObject: "55555"
            ]
        }
    }
}
```

第 7 章

企業リンケージサービスのサポート

この章では、以下の項目について説明します。

概要

Duns & Bradstreet （D&B）の企業リンケージサービスは、要求された組織の親とそのすべての関連エンティティを返します。D&B のリンケージサービスを使用して、組織のすべての支社および部門の情報を取得できます。リンケージサービスからのデータを使用して、レコードを作成および更新できます。

企業リンケージデータを MDM Hub にインポートできます。エンティティビューの DaaS プロバイダカスタムコンポーネントからリンケージサービスを使用できる、カスタムアプリケーションを開発する必要があります。

D&B サービスからデータをインポートしてそのデータでレコードを作成するビジネスエンティティサービスが必要です。外部ストレージでデータが変更されたら、レコードで対応する変更を行うことができる必要があります。D&B は、データの変更を通知する監視サービスを提供しています。変更前後のデータを受け入れて、対応するレコードに変更を適用するサービスが必要です。

DaaS インポートおよび更新用のビジネスエンティティサービス

DaaS インポートビジネスエンティティサービスは、リンケージサービスから XML 形式でデータを受け取って、レコードに変換します。DaaS 更新ビジネスエンティティサービスは、外部サービスから 2 つの XML ファ

イル形式のデータを受け取ります。この 2 つの XML ファイルは、変更前データと変更後データに対応します。更新サービスは、対応するレコードに変更を適用します。

関連項目：

- 「Daas インポート」 (ページ 246)
- 「DaaS 更新」 (ページ 249)

リンケージサポートの構成

D&B のリンケージサービスを使用してレコードを作成および更新するには、プロビジョニングツールで構成を追加し、カスタムアプリケーションを作成してリンケージサービスからの応答を分割する必要があります。

以下のタスクを実行して、D&B のリンケージサービスのサポートを構成します。

1. プロビジョニングツールを使用して、リンケージサービスの WSDL をアップロードします。
2. プロビジョニングツールを使用して、XML ドキュメントからビジネスエンティティへのトランスフォーメーションを構成し、それをサービスとして公開します。トランスフォーメーションをサービスとして公開すると、プロセスで DaaS インポートおよび更新ビジネスエンティティサービスが作成されます。
3. リンケージサービスのデータを要求し、応答をレコード詳細とリンケージ詳細に分割できる、カスタムアプリケーションを作成します。
4. 要求をカスタムアプリケーションに送信するユーザーインタフェースを開発します。

注: WSDL をアップロードし、XML からビジネスエンティティへのトランスフォーメーションを設定する方法の詳細については、『*Multidomain MDM のプロビジョニングツールガイド*』の「Data as a Service の統合」を参照してください。

リンケージデータ分割用のカスタムアプリケーション

D&B のリンケージサービスを使用するには、リンケージ情報をレコード詳細とリンケージ詳細に分割できるカスタムアプリケーションを設計する必要があります。

カスタムアプリケーションでは、次の関数を実行する必要があります。

1. エンティティビューからリンケージサービスに対する要求を受け入れる。
2. D&B に要求を送信して、応答を受信する
3. 応答を XML に変換する
4. 応答をレコード詳細とリンケージ詳細に分割する
5. XML 情報をビジネスエンティティサービスに送信して、データベースにレコードとして保存する
6. データの変更を監視して、外部サービスの List Change Notice 関数を呼び出す

第 8 章

データをクレンジング、分析、変換するための外部呼び出し

この章では、以下の項目について説明します。

概要

外部プロバイダは、レコードデータをクレンジング、分析、変換するための Web サービスを提供します。レコードを追加するときに、住所フィールドが空白かどうかをチェックするなどのカスタム検証で外部 Web サービスを使用します。レコードデータを変換するカスタムロジックに外部 Web サービスを使用します。例えば、2 つのレコードをマージするとき、住所はマージするが電話番号はマージしないようにすることができます。

外部 Web サービスでは、ビジネスエンティティサービスが呼び出すことのできる 1 つ以上の操作が公開されます。各操作には、要求タイプと応答タイプがあります。ビジネスエンティティサービスは、必要なサービスパラメータとともにレコードデータを外部サービスに送信します。実行ロジックの特定のステップで外部 Web サービスを呼び出すように設定できます。実装したロジックに基づいて、Data Director から要求が送信され、レコードデータが更新されます。必要に応じて、外部サービスはデータを変更できます。

プロビジョニングツールで、外部サービスを呼び出すビジネスエンティティとイベントを構成します。プロビジョニングツールで、外部サービス用の WSDL ファイルをアップロードして、SOAP サービスと操作を登録します。サービスを特定のビジネスエンティティとイベントにバインドします。

リソースキットの WSDL ファイルを使用して、サービスメソッドで交換するサービス、操作、メソッド、およびデータ型を把握します。外部 Web サービス用の custom-logic-service.wsdl ファイルは、次のリソースキットの場所にあります：C:\<infamdm installation directory>\hub\resourcekit\samples\BESExternalCall\source\resources\webapp\WEB-INF\wsdl\

リソースキットには、カスタムロジックと検証を実装するサンプルコードが含まれます。リソースキットをインストールすると、サンプルのカスタムロジックおよび検証用の bes-external-call.ear ファイルがアプリケーションサーバーにデプロイされます。

サービスフェーズ

ビジネスエンティティサービスは複数のサービスフェーズで構成されます。外部呼び出しを行い、サポートされているイベントをトリガするようにサービスフェーズを設定できます。

サービスフェーズは次の順序でトリガされます。

1. BeforeEverything。検証、ローカリゼーション、プロジェクションなどのサービスのロジックを実行する前に、外部呼び出しを行い、サポートされているイベントをトリガします。サンプルのユースケースは、このサービスフェーズを設定して、デフォルトのマージ動作を変更し、2 つのビジネスエンティティの属性のマージを防止したい場合などです。例えば、2 つ以上の Person レコードのマージ中に、親 Person レコードの子電話番号レコードのマージを防止することができます。

   注: プロジェクションは、特定のビジネスエンティティサービスの応答に相互参照情報を追加するプロセスです。プロジェクションは、特定のサービスフェーズの前後に発生する、可能な追加のステップと見なすことができます。

2. BeforeValidate。入力データを検証する前に、外部呼び出しを行い、サポートされているイベントをトリガします。サンプルのユースケースは、ビジネスエンティティを保持するときに入力データを検証する前に、ロジックを検証するようにこのサービスフェーズを設定したい場合などです。例えば、Person レコードを保持する前に、外部のサードパーティサービスに対して行われる住所または電話番号の検証。

3. AfterValidate。入力データを検証した後に、外部呼び出しを行い、サポートされているイベントをトリガします。サンプルのユースケースは、ビジネスエンティティを保持するときに入力データを検証した後に、ロジックを検証するようにこのサービスフェーズを設定したい場合などです。例えば、Person レコードを保持した後に、外部のサードパーティサービスに対して行われる住所または電話番号の検証。

   注: AfterValidate サービスがトリガされると、前のサービスフェーズでデータが変更または処理されたため、入力データが異なる場合があります。

4. AfterEverything。検証、ローカリゼーション、プロジェクションなどのサービスのロジックを実行した後に、外部呼び出しを行い、サポートされているイベントをトリガします。サンプルのユースケースは、サービスがすべてのロジックを実行した後に通知をトリガするようにこのサービスフェーズを設定したい場合などです。例えば、2 つ以上の Person レコードのマージが成功した後に電子メール通知を送信します。

   注: AfterEverything サービスフェーズがトリガされると、外部呼び出しは、入力データではなく、サービス応答からのデータにアクセスできます。

サービスフェーズをサポートするサービス

サービスフェーズをサポートする次の内部サービスを使用できます。

WriteCO

ビジネスエンティティを保持および更新します。WriteCO サービスは、次のサービスフェーズをサポートします。

- BeforeEverything
- BeforeValidate
- AfterValidate
- AfterEverything

これらのサービスフェーズは、ビジネスエンティティの保持前、保持中、および保持後にトリガされます。

WriteView

ビジネスエンティティビューを保持および更新します。WriteView サービスは、次のサービスフェーズをサポートします。

- BeforeEverything
- BeforeValidate
- AfterValidate
- AfterEverything

これらのサービスフェーズは、ビジネスエンティティの保持前、保持中、および保持後にトリガされます。

WriteView サービスは、サポートされているすべての WriteCO サービスフェーズを次の順序でトリガします。

1. WriteView.BeforeEverything
2. WriteView.BeforeValidate
3. WriteView.AfterValidate
4. WriteCO.BeforeEverything
5. WriteCO.BeforeValidate
6. WriteCO.AfterValidate
7. WriteCO.AfterEverything
8. WriteView.AfterEverything

ReadCO

ビジネスエンティティを取得します。ReadCO サービスは、次のサービスフェーズをサポートします。

- BeforeEverything
- AfterEverything

これらのサービスフェーズは、ビジネスエンティティの読み取りの前後にトリガされます。

ReadView

ビジネスエンティティビューを取得しています。ReadView サービスは、次のサービスフェーズをサポートします。

- BeforeEverything
- AfterEverything

これらのサービスフェーズは、ビジネスエンティティビューの読み取りの前後にトリガされます。

注: ReadView サービスは、サポートされているすべての ReadCO サービスフェーズを次の順序でトリガします。

1. ReadView.BeforeEverything
2. ReadCO.BeforeEverything
3. ReadCO.AfterEverything
4. ReadView.AfterEverything

PreviewMergeCO

複数のビジネスエンティティのマージプレビュー結果を取得します。PreviewMergeCO サービスは、次のサービスフェーズをサポートします。

- BeforeEverything

- AfterEverything

これらのサービスフェーズは、マージされたビジネスエンティティのプレビューの生成の前後にトリガされます。

MergeCO

複数のビジネスエンティティをマージします。MergeCO サービスは、次のサービスフェーズをサポートします。

- BeforeEverything
- AfterEverything

これらのサービスフェーズは、ビジネスエンティティのマージの前後にトリガされます。

プロセスメソッドパラメータ

外部呼び出しのビジネスロジックを実装する場合、プロセスメソッドパラメータを使用できます。

カスタムロジック Java インタフェースで次のパラメータを使用します。

helperContext

Service Data Objects（SDO）操作に必要なヘルパー実行コンテキストを定義します。

inputSdo

親レコードと子レコードを含む、ビジネスエンティティを表す Service Data Object を指定します。

inParams

前の要求からの入力プロパティを指定し、外部呼び出し内で定義されたサービスフェーズに直接依存します。inParams パラメータには、前の呼び出しからの outParams パラメータを含めることもできます。

例えば、validateOnly は入力プロパティであり、親レコード、子レコード、またはビジネスエンティティレベルで検証を実行することを示します。

validateOnly プロパティには次の値を設定することができます。

- true。親または子レコードの検証を実行します。例えば、このプロパティを指定して、10 個の電話番号を個別に検証できます。
- null または false。ビジネスエンティティレベルで検証を実行します。例えば、このプロパティを指定して、ユーザーがレコードを保存するときにビジネスエンティティの検証を実行できます。

もう 1 つの例は、外部呼び出しの名前を指定する servicePhase プロパティです。

outParams

サポートされている inputSdo および inParams パラメータを追加または上書きします。例えば、Merge_CO_Before_Everything イベントを含む外部呼び出しは、keysAndOverrides パラメータを更新して、ビジネスエンティティキーを追加したり、マージのキーを削除したりする場合があります。

戻り値では、inputSdo にデータ変更があると、外部呼び出しは次のいずれかの値を返します。

- null。inputSdo パラメータにデータ変更がない場合。
- 変更された inputSdo パラメータ。inputSdo パラメータにデータ変更があり、ビジネスエンティティの変更を反映する場合。

次のサンプルコードは、プロセスメソッドパラメータを含むカスタムロジック Java インタフェースを示しています。

```
public class ValidateOrgWriteBeforeEverything implements CustomLogic {
    @Override
    public DataObject process(HelperContext helperContext, DataObject inputSdo,
            Map<String, Object> inParams,
            Map<String, Object> outParams) throws StepException {
                    // your implementation
                    return null;
                              //or
                    return inputSdo;
            }
     }
```

外部呼び出しのベストプラクティス

外部呼び出しを実装するときは、次のベストプラクティスを考慮してください。

- データをクレンジング、分析、または変換する既存の関数が存在しない場合は、外部呼び出しを実装します。
- 外部呼び出しを実装するときは、呼び出しが MDM Hub およびビジネスエンティティサービスの呼び出し応答のパフォーマンスに影響を与える可能性があるため、注意が必要です。
- 外部呼び出しをいつ使用するかを特定します。例えば、ユーザーが 1 つの子レコードから別の子レコードに移動したとき、またはユーザーが **[保存]** をクリックしたときに、外部呼び出しを実行します。
- 外部呼び出しごとに個別のクラスを定義します。
- inParams プロセスメソッドパラメータを validateOnly プロパティとともに外部呼び出しに含める場合は、validateOnly プロパティを次のいずれかの値に設定していることを確認してください。
  - false または null。ユーザーが **[保存]** をクリックした後に外部呼び出しを実行します。
  - true。ユーザーが親または子レコードを更新した後、外部呼び出しを実行してから、別の子レコードに移動します。
- AfterEverything イベントまでの後続の外部呼び出しにカスタムプロパティを含める場合は、outParams プロセスメソッドパラメータにカスタムプロパティを追加します。
- 保存操作中の最後の WriteCO.AfterEverything イベントは、親レコードと子レコードの行 ID を返します。他のデータが必要な場合は、ビジネスエンティティサービスの読み取りを呼び出します。
- 読み取り操作は次の操作中に発生するため、読み取り操作の外部呼び出しは実装しないでください。
  - 検索
  - 書き込む前に読み取り
  - 書き込んだ後に読み取り
- クレンジング関数を使用してレコードを検証します。
- 追加の検証を実装する場合は、validateOnly プロパティが次のいずれかの値に設定されているときに、WriteCO.AfterEverything イベントで追加の検証を実行します。
  - false または null。ベースオブジェクトの場合。
  - true。子レコードの場合。
- ビジネスエンティティサービス呼び出しを使用してデータの読み取りまたは書き込みを行う外部アプリケーションと同様に、外部呼び出しからビジネスエンティティサービス呼び出しを行います。

外部呼び出しの作成

ビジネスエンティティサービスにはサービス手順があります。入力要求は各サービス手順を通過します。ビジネスエンティティサービス実行ロジックの特定の手順に対して、外部サービスへの外部呼び出しを作成できます。データをクレンジング、分析、または変換する関数が存在しない場合は、外部呼び出しを作成します。

外部呼び出しを作成するには、次の手順を実行します。

1. サンプルの Web サービスを理解してデプロイします。
2. プロビジョニングツールにログインします。
3. WSDL ファイルをアップロードします。
4. SOAP サービスを登録します。
5. 外部呼び出しを設定します。
6. 設定をパブリッシュします。

外部呼び出しを作成した後、呼び出しが期待どおりに機能することを確認するために、呼び出しをテストします。

これらの手順を含むサンプルの外部呼び出しを表示するには、「サンプル外部呼出し」 (ページ 292)を参照してください。

サンプル外部呼出し

外部呼び出しを作成する方法については、Resource Kit に含まれている BESExternalCall サンプルを確認してください。次のセクションには、BESExternalCall サンプルを作成およびテストするために必要な手順に関する情報が含まれています。

BESExternalCall サンプルは、外部呼び出しのセットアップを成功させるための最も簡単な基盤を実装しています。

BESExternalCall サンプルのコード構造は次のとおりです。

- ValidatePerson.java ファイルと MergePerson.java ファイルの CustomLogic 実装
- CustomLogicFactorImpl.java ファイルの CustomLogicFactory 実装
- CustomLogicService.java ファイルの WebService 実装

ValidatePerson.java ファイルと MergePerson.java ファイルの CustomLogic 実装

外部呼び出しの CustomLogic 実装には、追加のデータ検証や操作などのカスタムビジネスロジックが含まれます。

CustomLogic 実装には、mdm-spi.jar ファイルで使用できる com.informatica.mdm.spi.externalcall.CustomLogic インタフェースを含める必要があります。このインタフェースは、カスタムビジネスロジックを定義できるプロセスメソッドを公開します。次の ValidatePerson.java クラスコードはサンプルです。

```
public class ValidatePerson implements CustomLogic {
    private CompositeServiceClient besClient;
    private CallContext callContext;
    public ValidatePerson(CompositeServiceClient besClient, CallContext callContext) {
        this.besClient = besClient;
        this.callContext = callContext;
    }
    @Override
```

```
    public DataObject process(HelperContext helperContext, DataObject inputSdo,
                              Map < String, Object > inParams,
                              Map < String, Object > outParams) throws StepException {
            try {
                List < ValidationError > errorList = new ArrayList < > ();
                DataFactory dataFactory = helperContext.getDataFactory();

                // Ensure that the Person SDO has at least one Address item
                List<?> list = inputSdo.getList("Person/Addresses/item");
                if ((list == null) || list.isEmpty()) {
                    String personId = inputSdo.getString("Person/rowidObject");

                    if(personId != null) { // Verify if an address already exists
                        DataObject readEntity = dataFactory.create(ReadEntity.class);
                        DataObject personFilter =

readEntity.createDataObject("parameters").createDataObject("coFilter").createDataObject("object");
                        personFilter.setString("name", "Person");
                        personFilter.createDataObject("key").setString("rowid", personId);
                        DataObject addressFilter = personFilter.createDataObject("object");
                        addressFilter.setString("name", "Addresses");
                        addressFilter.createDataObject("pager").setInt("recordsToReturn", 1);
                        DataObject readEntityReturn = besClient.process(callContext, readEntity);
                        List existingList = readEntityReturn.getList("object/Person/Addresses/item");
                        if (existingList == null || existingList.isEmpty()) {
                            errorList.add(createValidationError(dataFactory, "CUSTOM-00001", "You must enter
at least one Address.", "Person.Addresses"));
                        }
```

CustomLogicFactorImpl.java ファイルの CustomLogicFactory 実装

CustomLogicFactory 実装は、期待されるフェーズで実行するか、ターゲットビジネスエンティティに対して実行するか、またはその両方を行う CustomLogic インスタンスを提供します。

CustomLogicFactory 実装には、mdm-spi.jar ファイルで使用できる com.informatica.mdm.spi.externalcallCustomLogicFactory インタフェースを含める必要があります。このインタフェースは、外部呼び出し要求が初期段階で呼び出す create メソッドを公開します。このメソッドで期待されるロジックは、外部呼び出し要求をフィルタリングして、期待されるカスタムロジック実装に転送します。次の ValidatePerson.java クラスコードはサンプルです。

```
public class CustomLogicFactoryImpl implements CustomLogicFactory {

            public static final String PERSON = "Person";

            private static final CustomLogic EMPTY_LOGIC = new EmptyLogic();

            private CompositeServiceClient besClient;

            public CustomLogicFactoryImpl(CompositeServiceClient besClient) { this.besClient = besClient;
            }

            @Override
            public CustomLogic create(ExternalCallRequest externalCallRequest, CallContext callContext)
throws StepException {
             // Interest is in the Person business entity. The logic can handle a few service phases.
                Trigger trigger = externalCallRequest.getTrigger();
                String businessEntity = trigger.getBusinessEntity();
                ServicePhase phase = trigger.getServicePhase();

                switch (phase) {
                    case WRITE_CO_BEFORE_VALIDATE:
                        if (PERSON.equals(businessEntity)) {
                            return new ValidatePerson(besClient, callContext);
                        }
                        break;
                    case PREVIEW_MERGE_CO_BEFORE_EVERYTHING:
                    case MERGE_CO_BEFORE_EVERYTHING:
                        if (PERSON.equals(businessEntity)) {
                            return new MergePerson();
```

```
                    }
                    break;
                default:
                    //
            }
            return EMPTY_LOGIC; // This one does nothing
        }
```

CustomLogicFactory 実装には、次の create メソッドパラメータを含めることができます。

externalCallRequest

bes-external-call.xsd ファイルで定義されている外部呼び出しを表す要求のインスタンス。

callContext

CompositeServiceClient 要求がカスタムロジック実装で使用する複合サービスコンテキストのインスタンス。

CustomLogicService.java ファイルの WebService 実装

WebService 実装は、外部呼び出しが呼び出す完全なサービスを公開し、外部呼び出しフローのエントリポイントになります。

次の CustomLogicService.java サンプルコードは、bes-external-call.xsd ファイルが Java のネイティブ WebService API に基づいて定義する Web サービスを実装します。

```
/**
* Web service implementation.
* It must accept a {urn:bes-external-call.informatica.mdm}ExternalCallRequest as an operation input
* and return a {urn:bes-external-call.informatica.mdm}ExternalCallResponse as output.
*/
@WebServiceProvider(
        targetNamespace = "http://cs.sample.mdm.informatica.com/",
        serviceName = "CustomLogicService",
        portName = "CustomLogicServicePort",
        wsdlLocation = "WEB-INF/wsdl/custom-logic-service.wsdl"
)
@ServiceMode(Mode.PAYLOAD)
public class CustomLogicService implements Provider<Source> {

    @Override
    public Source invoke(Source request) {

        CompositeServiceClient compositeServiceClient =
createCompositeServiceClient();
        CustomLogicFactory customLogicFactory = new
CustomLogicFactoryImpl(compositeServiceClient);
        String appName = "<trusted_user>"; // replace with proper application user name

        // Create a processor instance and let the instance perform the job.
        // Provide a custom logic factory implementation.
        ExternalCallProcessor externalCallProcessor =
                new ExternalCallProcessor(compositeServiceClient, appName, customLogicFactory);

 return externalCallProcessor.invoke(request);
}
```

ExternalCallProcessor インスタンスの invoke メソッドは、受信要求を処理します。

注: ExternalCallProcessor インスタンスは、信頼できるアプリケーションユーザーに依存しています。信頼できるアプリケーションユーザーの詳細については、『*Multidomain MDM のセキュリティガイド*』を参照してください。

BESExternalCall の例

Web サービスへの BESExternalCall 外部呼び出しは、次のアクションを実行できます。

- 人物を作成または更新するとき、呼び出しはその人物が少なくとも 1 つの住所を持っているかどうかを検証します。検証が失敗した場合、呼び出しはカスタマイズされたエラーメッセージを返します。
- マージされたレコードのプレビューを生成する場合、呼び出しは電話番号をマージしません。
- レコードをマージする場合、呼び出しは電話番号をマージしません。

次の表に、BESExternalCall の例をサポートするファイルとファイルの場所を示します。

ファイル名	説明	場所
bes-external-call.xsd	すべての外部呼び出しの要求と応答のタイプを定義します。外部 Web サービスの操作では、入力要素および出力要素として、要求タイプおよび応答タイプを使用する必要があります。	<MDM installation directory>/hub/server/lib/mdm-spi.jar
custom-logic-service.wsdl	サンプル Web サービスの WSDL ファイル。サービスメソッドが交換する外部サービス、操作、メソッド、およびデータ型を定義します。	<MDM installation directory>/hub/resourcekit/samples/BESExternalCall/source/resources/webapp/WEB-INF/wsdl/
ValidatePerson.java および MergePerson.java	サンプルの Web サービスで実行されるカスタムの検証およびマージ操作の例。 **注**: この例では.java ファイルを使用していますが、外部 Web サービスで別のコーディング標準が必要になる場合があります。	<MDM installation directory>/hub/resourcekit/samples/BESExternalCall/source/java/
build.xml	bes-external-call.ear ファイルを作成します。	<MDM installation directory>/hub/resourcekit/samples/BESExternalCall/
bes-external-call.ear	この例を実行するためにアプリケーションサーバーにデプロイする.ear ファイル。	<MDM installation directory>/hub/resourcekit/samples/BESExternalCall/build/

手順 1.サンプルの Web サービスの理解とデプロイ

BESExternalCall の例には、WSDL ファイルと、外部呼び出しが使用するカスタムロジックを使用した Java ファイルが含まれています。例に含まれるファイルを確認してください。この例を使用するには、.ear ファイルを作成して、MDM Hub を実行するアプリケーションサーバーにこのファイルをデプロイします。

外部呼び出しを設計する場合、最初に外部サービスのフレームワークでカスタムロジックを開発します。次に、外部サービス、操作、メソッドと、サービスメソッドが交換するデータ型を定義する WSDL ファイルを作成します。WSDL ファイルが bes-external-call.xsd ファイルで定義されている要求と応答のタイプを使用していることを確認してください。

1. 要求タイプと応答タイプの要件を理解するには、bes-external-call.xsd ファイルを確認してください。
2. サンプル Web サービスの要求タイプと応答タイプの実装を理解するには、 custom-logic-service.wsdl ファイルを確認してください。
3. サンプル Web サービスが使用するカスタムロジックを理解するには、Resource Kit の Java ファイルを確認してください。

4. サンプルの.ear ファイルを作成するには、ANT 作成ツールで build.xml ファイルを使用します。

   EAR ファイルの作成の詳細については、『*Multidomain MDM のリソースキットガイド*』を参照してください。

5. bes-external-call.ear ファイルを MDM Hub を実行するアプリケーションサーバーにデプロイします。

   EAR ファイルのデプロイの詳細については、『*Multidomain MDM のリソースキットガイド*』を参照してください。

手順 2.プロビジョニングツールへのログイン

プロビジョニングツールで Web サービスと外部呼び出しを設定します。

ログインするには、プロビジョニングツールの URL とユーザー資格情報が必要です。URL には MDM Hub Server のホスト名とポート番号が含まれています。この情報がない場合は、MDM Hub の管理者に問い合わせてください。

プロビジョニングツールと同じアプリケーションサーバーで MDM Hub を実行する必要があります。プロビジョニングツールの詳細については、『*Multidomain MDM のプロビジョニングツールガイド*』を参照してください。

1. サポートされているブラウザを開きます。
2. プロビジョニングツールの URL を入力します。

   URL の形式は次のとおりです。

   - セキュリティ保護されている接続。 https://<MDM Hub Server host name>:<MDM Hub Server port number>/provisioning/
   - セキュリティ保護されていない接続。 http://<MDM Hub Server host name>:<MDM Hub Server port number>/provisioning/

   [ログイン] ページが開きます。
3. ユーザ名とパスワードを入力します。
4. **[ログイン]** をクリックします。
5. ORS のプロンプトが表示されたら、サンプル MDM ORS を選択します。

   プロビジョニングツールが開き、**[ホーム]** ページが表示されます。

手順 3.サンプル WSDL ファイルのアップロード

BESExternalCall のサンプル Web サービスの WSDL ファイルをアップロードします。

WSDL ファイルのアップロードの詳細については、『*Multidomain MDM のプロビジョニングツールガイド*』を参照してください。

1. プロビジョニングツールで、**[ビジネスエンティティ]** > **[拡張]** をクリックします。

   [拡張] ページが開きます。
2. **[WSDL]** を選択し、**[作成]** をクリックします。

3. プロパティペインで次のプロパティを指定します。

プロパティ	説明	値の例
名前	サンプル Web サービスを説明する名前。	besexternal
URL	WSDL ファイルの URL。	`https://<host>:<port>/bes-external-call/customLogicService?wsdl`

4. **［適用］**をクリックします。

   ［ツリービュー］パネルに詳細が表示されます。

手順 4.SOAP サービスの登録

WSDL が示す SOAP サービスと操作を登録します。

SOAP サービスの登録の詳細については、『*Multidomain MDM のプロビジョニングツールガイド*』を参照してください。

1. **［拡張］**ページで**［SOAP サービス］**を選択し、**［作成］**をクリックします。

   プロパティパネルにプロパティが表示されます。

2. プロパティパネルで以下のプロパティを指定します。

プロパティ	説明	例値
名前	SOAP サービスの名前。	userexit
WSDL	サンプル WSDL ファイルをアップロードしたときに指定した名前。 Web サービスの詳細は、［名前空間］フィールド、［サービス名］フィールド、［ポート名］フィールド、および［エンドポイントアドレス］フィールドに表示されます。	besexternal
SOAP ヘッダー	ユーザー名やパスワードなど、Web サービスの実行に必要な追加情報。	BESExternalCall の例では、SOAP ヘッダーはありません。

3. **［適用］**をクリックします。

   ［ツリービュー］パネルに詳細が表示されます。

手順 5.外部呼び出しの設定

外部呼び出しは、設定された SOAP サービスと操作を使用し、呼び出しは 3 つのサービスフェーズでトリガされます。

外部呼出しの設定の詳細については、『*Multidomain MDM のプロビジョニングツールガイド*』を参照してください。

1. **［拡張］**ページで**［外部呼び出し］**を選択し、**［作成］**をクリックします。

2. **［プロパティ］**パネルで次のプロパティを設定します。

プロパティ	説明	例値
名前	外部呼び出しの名前。	validateperson
SOAP サービス	設定した SOAP サービス。	userexit
SOAP 操作	実行する SOAP 操作。	validate

3. **［適用］**をクリックします。

   ［ツリービュー］パネルは、SOAP サービスと操作のコンテンツに基づいて入力されます。
4. **［ツリービュー］**パネルで**［ビジネスエンティティ］**を選択し、**［作成］**をクリックして、**［Person］**を選択します。
5. **［サービスフェーズ］**を選択し、**［作成］**をクリックして、サンプルコードに対して次のサービスフェーズを選択します。
   - WriteCO.BeforeValidate
   - PreviewMergeCO.BeforeEverything
   - MergeCO.BeforeEverything

   注: カスタムロジックに基づいて、サービスフェーズを選択できます。
6. **［適用］**をクリックします。

手順 6.設定のパブリッシュ

WSDL ファイル、SOAP サービス、および外部呼び出しの設定は、キャッシュに格納されます。設定を MDM Hub にパブリッシュします。

プロビジョニングツールで、**［パブリッシュ］**をクリックします。

パブリッシュプロセスにより、設定が MDM Hub に保存されます。

アドレス検証のテスト

REST API を使用して、検証ステップが機能することを確認します。Person レコードを住所なしで作成します。外部呼び出しの設定が正しい場合は、Person レコードに少なくとも 1 つの住所が必要であることを示す検証エラーが表示されます。

REST URL を使用するには、MDM サンプル ORS のデータベース ID と MDM ユーザー名が必要です。

1. MDM サンプル ORS のデータベース ID を見つけます。
   a. Hub コンソールにログインします。
   b. **［設定］**ワークベンチで、**［データベース］**をクリックします。
   c. MDM サンプルデータベースを選択します。
   d. **［データベースのプロパティ］**パネルで、**［データベース ID］**の値をコピーします。この値を REST URL で使用します。
2. ［作成］API を使用して、Person ビジネスエンティティを住所なしで作成します。

```
POST https://<host>:<port>/cmx/cs/<database id>/Person?systemName=<user name>
{
        firstName: "John"
    }
```

検証テストは失敗し、エラーが表示されます。Person レコードは作成されません。

3. 住所を持つ Person ビジネスエンティティを作成します。

```
POST https://<host>:<port>/cmx/cs/<database id>/Person?systemName=<user name>
{
        firstName: "John",
        Addresses: {
            item: [
                {
                    cityName: "Toronto"
                }
            ]
        }
    }
```

検証テストはエラーなしで成功します。Person レコードが作成されます。

マージロジックのテスト

電話番号のマージを防止するマージロジックをテストするには、MergePreview API を使用します。

1. 2 つの Person ビジネスエンティティを作成するには、作成 API を使用します。

```
POST https://<host>:<port>/cmx/cs/<ors>/Person?systemName=Admin
    {
        firstName: "John",
        Addresses: {
            item: [
                {
                    cityName: "Toronto"
                }
            ]
        },
        TelephoneNumbers: {
            item:[
                {
                    phoneNum: "111-11-11"
                }
            ]
        }
    }
POST https://<host>:<port>/cmx/cs/<ors>/Person?systemName=Admin
    {
        firstName: "John",
        Addresses: {
            item: [
                {
                    cityName: "Ottawa"
                }
            ]
        },
        TelephoneNumbers: {
            item:[
                {
                    phoneNum: "222-22-22"
                }
            ]
        }
    }
```

応答には、次のサンプル行 ID が含まれます。

- Person: 161923、161924
- 住所: 2123、2124
- 電話番号: 101723、101724

2. マージされた Person ビジネスエンティティをプレビューするには、PreviewMerge API を実行します。

```
POST https://<host>:<port>/cmx/cs/<ors>/Person/161923?action=previewMerge&depth=2
{
        keys: [
            {
                rowid: "161924"
            }
        ]
    }
```

応答には、2 つの住所と 2 つの電話番号が含まれます。


```
{
  "Person": {
    "rowidObject": "161923        ",
    "creator": "admin",
    "createDate": "2016-10-20T09:50:35.878-04:00",
    "updatedBy": "admin",
    "lastUpdateDate": "2016-10-20T09:50:35.879-04:00",
    "consolidationInd": 4,
    "lastRowidSystem": "SYSO          ",
    "hubStateInd": 1,
    "label": "Person: , Bill",
    "partyType": "Person",
    "displayName": "Bill",
    "firstName": "Bill",
   "TelephoneNumbers": {
      "link": [],
      "firstRecord": 1,
      "recordCount": 2,
      "pageSize": 2,
      "item": [
        {
          "rowidObject": "101723        ",
          "creator": "admin",
          "createDate": "2016-10-20T09:50:35.904-04:00",
          "updatedBy": "admin",
          "lastUpdateDate": "2016-10-20T09:50:35.905-04:00",
          "consolidationInd": 4,
          "lastRowidSystem": "SYSO          ",
          "hubStateInd": 1,
          "label": "PhoneNumbers",
          "phoneNum": "111-1111     ",
          "phoneCountryCd": "1"
        },
        {
          "rowidObject": "101724        ",
          "creator": "admin",
          "createDate": "2016-10-20T09:50:54.768-04:00",
          "updatedBy": "admin",
          "lastUpdateDate": "2016-10-20T09:50:54.769-04:00",
          "consolidationInd": 4,
          "lastRowidSystem": "SYSO          ",
          "hubStateInd": 1,
          "label": "PhoneNumbers",
          "phoneNum": "222-2222     ",
          "phoneCountryCd": "1"
        }
      ]
    },
    "Addresses": {
      "link": [],
      "firstRecord": 1,
      "recordCount": 2,
      "pageSize": 2,
      "item": [
        {
          "rowidObject": "2123          ",
          "creator": "admin",
```

```
            "createDate": "2016-10-20T09:50:37.956-04:00",
            "updatedBy": "admin",
            "lastUpdateDate": "2016-10-20T09:50:37.956-04:00",
            "consolidationInd": 4,
            "lastRowidSystem": "SYSO          ",
            "hubStateInd": 1,
            "label": "Addresses",
            "Address": {
              "rowidObject": "2121          ",
              "creator": "admin",
              "createDate": "2016-10-20T09:50:36.922-04:00",
              "updatedBy": "admin",
              "lastUpdateDate": "2016-10-20T09:50:37.923-04:00",
              "consolidationInd": 4,
              "lastRowidSystem": "SYSO          ",
              "hubStateInd": 1,
              "label": "Address",
              "cityName": "Toronto"
            }
          },
          {
            "rowidObject": "2124          ",
            "creator": "admin",
            "createDate": "2016-10-20T09:50:54.790-04:00",
            "updatedBy": "admin",
            "lastUpdateDate": "2016-10-20T09:50:54.790-04:00",
            "consolidationInd": 4,
            "lastRowidSystem": "SYSO          ",
            "hubStateInd": 1,
            "label": "Addresses",
            "Address": {
              "rowidObject": "2122          ",
              "creator": "admin",
              "createDate": "2016-10-20T09:50:54.777-04:00",
              "updatedBy": "admin",
              "lastUpdateDate": "2016-10-20T09:50:54.777-04:00",
              "consolidationInd": 4,
              "lastRowidSystem": "SYSO          ",
              "hubStateInd": 1,
              "label": "Address",
              "cityName": "Ottawa"
            }
          }
        ]
      }
    }
  }
```

3. マージオーバーライドを使用して PreviewMerge API を実行します。

   オーバーライドは、電話番号のマージを防止する外部呼び出しをトリガします。

```
POST https://<host>:<port>/cmx/cs/<ors>/Person/161923?action=previewMerge&depth=3
{
    keys: [
        { rowid: "161923" }
    ],
    overrides: {
        Person: {
            Addresses: {
                item:[
                    {
                        rowidObject: "2123",
                        MERGE: {
                            item:[{key:{rowid: "2124"}}],
                            $original: {
                                item:[null]
                            }
                        }
                    }
                ]
```

```
            },
            TelephoneNumbers: {
                item:[
                    {
                        rowidObject: "101723",
                        MERGE: {
                            item:[{key:{rowid: "101724"}}],
                            $original: {
                                item:[null]
                            }
                        }
                    }
                ]
            }
        }
    }
}
```

応答には、1 つの住所と 2 つの電話番号が含まれます。

```
{
  "Person": {
    "rowidObject": "161923        ",
    "creator": "admin",
    "createDate": "2016-10-20T09:50:35.878-04:00",
    "updatedBy": "admin",
    "lastUpdateDate": "2016-10-20T09:50:35.879-04:00",
    "consolidationInd": 4,
    "lastRowidSystem": "SYS0          ",
    "hubStateInd": 1,
    "label": "Person: , Bill",
    "partyType": "Person",
    "displayName": "Bill",
    "firstName": "Bill",
   "TelephoneNumbers": {
      "link": [],
      "firstRecord": 1,
      "recordCount": 2,
      "pageSize": 2,
      "item": [
        {
          "rowidObject": "101723        ",
          "creator": "admin",
          "createDate": "2016-10-20T09:50:35.904-04:00",
          "updatedBy": "admin",
          "lastUpdateDate": "2016-10-20T09:50:35.905-04:00",
          "consolidationInd": 4,
          "lastRowidSystem": "SYS0          ",
          "hubStateInd": 1,
          "label": "PhoneNumbers",
          "phoneNum": "111-1111     ",
          "phoneCountryCd": "1"
        },
        {
          "rowidObject": "101724        ",
          "creator": "admin",
          "createDate": "2016-10-20T09:50:54.768-04:00",
          "updatedBy": "admin",
          "lastUpdateDate": "2016-10-20T09:50:54.769-04:00",
          "consolidationInd": 4,
          "lastRowidSystem": "SYS0          ",
          "hubStateInd": 1,
          "label": "PhoneNumbers",
          "phoneNum": "222-2222     ",
          "phoneCountryCd": "1"
        }
      ]
    },
      "Addresses": {
      "link": [],
      "firstRecord": 1,
```

```
      "recordCount": 1,
      "pageSize": 1,
      "item": [
        {
          "rowidObject": "2123          ",
          "creator": "admin",
          "createDate": "2016-10-20T09:50:37.956-04:00",
          "updatedBy": "admin",
          "lastUpdateDate": "2016-10-20T09:50:37.956-04:00",
          "consolidationInd": 4,
          "lastRowidSystem": "SYS0          ",
          "hubStateInd": 1,
          "label": "Addresses"
        }
      ]
    },
    "PersonDetails": {
      "link": [],
      "recordCount": 0,
      "pageSize": 0,
      "item": []
    }
  }
}
```


MergeCO.BeforeEverything 外部呼び出しを使用したソース行 ID の取得

マージアクション中、ソース行 ID はソースベースオブジェクトのトレースに役立ちます。MergeCO.BeforeEverything 外部呼び出しを使用して、inputSDO プロセスメソッドパラメータからソース行 ID を取得できます。

ソース行 ID を取得するには、次の手順を実行します。

1. MergeCO.BeforeEverything 外部呼び出しを実装します。

   MergePerson.java ファイルからの次のサンプルコードは、MergeCO.BeforeEveryting 外部呼び出しでソース行 ID を取得します。

```
package com.informatica.mdm.sample.cs;

import java.util.List;
import java.util.Map;

import com.informatica.mdm.sdo.co.base.BaseObject;
import com.informatica.mdm.sdo.co.base.CompositeObjectBase;
import com.informatica.mdm.sdo.co.base.KeysAndOverrides;
import com.informatica.mdm.sdo.co.base.MergedRecord;
import com.informatica.mdm.sdo.co.base.MergedRecordPager;
import com.informatica.mdm.sdo.co.base.Pager;
import com.informatica.mdm.sdo.cs.base.BusinessEntityKey;
import com.informatica.mdm.sdo.cs.base.Key;
import com.informatica.mdm.spi.cs.StepException;
import com.informatica.mdm.spi.externalcall.CustomLogic;
import commonj.sdo.ChangeSummary;
import commonj.sdo.DataObject;
import commonj.sdo.Property;
import commonj.sdo.helper.HelperContext;
import org.eclipse.persistence.sdo.SDOChangeSummary;

/**
 * Call this piece of code before the Person business entity record merge or the previewMerge API call.
```

```
 */
public class MergePerson implements CustomLogic {

    public static final String KEYS_AND_OVERRIDES = "keysAndOverrides";
    public static final String OVERRIDES = "overrides";
    public static final String MERGE = "MERGE";

    @Override
    public DataObject process(HelperContext helperContext, DataObject inputSdo,
            Map<String, Object> inParams,
            Map<String, Object> outParams) throws StepException {

        DataObject keysAndOverrides = (DataObject) inParams.get(KEYS_AND_OVERRIDES);

        if (keysAndOverrides != null) {

            explainMerge(inParams);

            DataObject overrides = keysAndOverrides.getDataObject(OVERRIDES);
            if (overrides != null) {
                // if "overrides" are provided, then user is not simply merging several records, but
                // tryes to merge some child records or provides some winner override
                List<DataObject> list = overrides.getList("Person/TelephoneNumbers/item");
                if ((list != null) && !list.isEmpty()) {
                    SDOChangeSummary changeSummary = (SDOChangeSummary) overrides.getChangeSummary();
                    changeSummary.resumeLogging();
                    // remove "MERGE" element from all "Telephone" items, this will prevent mering of
"Telephone" child records
                    for (DataObject dataObject : list) {
                        if (dataObject.isSet(MERGE)) {
                            dataObject.unset(MERGE);
                        }
                    }
                    // send updated "overrides" back to Hub
                    outParams.put(KEYS_AND_OVERRIDES, keysAndOverrides);
                }
            }
        }
        return inputSdo;
    }

    private void explainMerge(Map<String, Object> inParams) {
        BusinessEntityKey businessEntityKey = (BusinessEntityKey) inParams.get("businessEntityKey");
        System.out.println(String.format("%s:%s", businessEntityKey.getName(),
businessEntityKey.getKey().getRowid()));
        KeysAndOverrides keysAndOverrides = (KeysAndOverrides) inParams.get("keysAndOverrides");
        List<Key> keys = keysAndOverrides.getKeys();
        for (Key key : keys) {
            System.out.println(String.format(" | <- %s", key.getRowid()));
        }
        CompositeObjectBase overrides = keysAndOverrides.getOverrides();
        if(overrides != null) {
            DataObject co = (DataObject) overrides;
            ChangeSummary changeSummary = co.getChangeSummary();
            DataObject root = co.getDataObject(co.getType().getName());
            dumpOne("", co.getType().getName(), root, changeSummary);
        }
    }

    private void dumpOne(String prefix, String name, DataObject dataObject, ChangeSummary changeSummary) {
        String rowidObject = dataObject.getString("rowidObject");
        if(rowidObject != null) {
            System.out.println(String.format("%s%s:%s", prefix, name, rowidObject));
        }
        List<MergedRecord> mergeItems = dataObject.getList("MERGE/item");
        if(mergeItems != null) {
            for (MergedRecord mergeItem : mergeItems) {
                System.out.println(String.format("%s | <- %s", prefix, mergeItem.getKey().getRowid()));
            }
```

```
        }
        List<Property> properties = dataObject.getInstanceProperties();
        for (Property property : properties) {
            Object o = dataObject.get(property);
            if((o instanceof Pager) && !(o instanceof MergedRecordPager)) {
                DataObject pager = (DataObject) o;
                System.out.println(prefix + property.getName());
                List<DataObject> items = pager.getList("item");
                for (int i = 0; i < items.size(); i++) {
                    DataObject item = items.get(i);
                    dumpOne(prefix + "  " ,"[" + i + "]", item, changeSummary);
                }
            }
            if(o instanceof BaseObject) {
                dumpOne(prefix + "  ", property.getName(), (DataObject) o, changeSummary);
            }
        }
        if(changeSummary.isModified(dataObject)) {
            List<ChangeSummary.Setting> oldValues = changeSummary.getOldValues(dataObject);
            for (ChangeSummary.Setting setting : oldValues) {
                Property property = setting.getProperty();
                if(!"MERGE".equals(property.getName())) {
                    Object value = dataObject.get(property);
                    System.out.println(String.format("%s manual override: %s = %s", prefix,
property.getName(), value));
                }
            }
        }
    }
}
```

注: inputSDO プロセスメソッドパラメータはマージターゲットです。キーとオーバーライドはすべて、マージソースからのものです。

2. 外部呼び出しをトリガするには、Postman などの標準の REST クライアントを使用してマージ API またはマージのプレビュー API の POST 要求を実行します。

   プロセスは MergeCO.BeforeEverything イベントを呼び出します。

   次のサンプル POST 要求は、ソース行 ID を出力します。

   ```
   POST: http://localhost:8080/cmx/cs/localhost-orcl103-MDM_SAMPLE/Person/{{person1_id}}?action=previewMerge

   {
           "keys": [
               {
                   "rowid": "{{person2_id}}"
               }
           ],
           "overrides": {
               "Person": {
                   "Addresses": {
                       "item":[
                           {
                               "rowidObject": "{{person1_Addresses1_id}}",
                               "MERGE": {
                                   "item":[{"key":{"rowid": "{{person2_Addresses1_id}}"}}],
                                   "$original": {
                                       "item":[null]
                                   }
                               }
                           }
                       ]
                   },
                   "TelephoneNumbers": {
                       "item":[
                           {
                               "rowidObject": "{{person1_TelephoneNumbers1_id}}",
                               "phoneNum": "777-77-77",
                               "$original": {
                                   "phoneNum":null
   ```

```
                        },
                        "MERGE": {
                            "item":[{"key":{"rowid": "{{person2_TelephoneNumbers1_id}}"}}],
                            "$original": {
                                "item":[null]
                            }
                        }
                    }
                ]
            }
        }
    }
}
```

次のサンプル出力は、ソース行 ID を表示します。

```
2020-03-20 11:59:49,631 INFO  [stdout] (default task-55) Person:401924
2020-03-20 11:59:49,631 INFO  [stdout] (default task-55)  | <- 401925
2020-03-20 11:59:49,632 INFO  [stdout] (default task-55)  | <- 401926
2020-03-20 11:59:49,632 INFO  [stdout] (default task-55) TelephoneNumbers
2020-03-20 11:59:49,632 INFO  [stdout] (default task-55)   [0]:1724
2020-03-20 11:59:49,632 INFO  [stdout] (default task-55)    | <- 1725
2020-03-20 11:59:49,633 INFO  [stdout] (default task-55)    | <- 1726
2020-03-20 11:59:49,633 INFO  [stdout] (default task-55)    manual override: phoneNum = 777-77-77
2020-03-20 11:59:49,633 INFO  [stdout] (default task-55) Addresses
2020-03-20 11:59:49,633 INFO  [stdout] (default task-55)   [0]:2124
2020-03-20 11:59:49,633 INFO  [stdout] (default task-55)    | <- 2125
2020-03-20 11:59:49,634 INFO  [stdout] (default task-55)    | <- 2126
```

例: カスタム検証およびビジネスエンティティサービスのロジック

Person レコードを追加およびマージするとき、カスタム検証とロジックをテストできます。カスタム検証では、Person レコードに住所があるかどうかを確認します。カスタムロジックでは、2 つの電話番号のマージは許可されません。REST API を使用して、Person レコードを作成およびマージします。

1. Person レコードの作成時に検証をチェックするには、次の手順を実行します。
   a. ［作成］API を使用して、住所なしの Person レコードを作成します。検証エラーが表示されます。
   b. ［作成］API を使用して、住所を持つ Person レコードを作成します。操作は成功します。
2. レコードのマージ時にカスタムロジックをチェックするには、次の手順を実行します。
   a. ［作成］API を使用して、住所と電話番号を持つ 2 つの Person レコードを作成します。
   b. ［マージのプレビュー］API を使用して、2 つの Person レコードをマージします。マージのプレビュー要求に overrides を追加して、住所と電話番号をマージします。応答では、住所は 1 つですが電話番号が 2 つ表示されます。カスタムロジックによって、電話番号のマージは阻止されます。

前提条件

カスタムロジックと検証をチェックするには、プロビジョニングツールで WSDL ファイルをアップロードする必要があります。SOAP サービスと操作を登録する必要があります。サービスを、カスタムロジックと検証を

使用するビジネスエンティティとイベントにバインドします。指定したビジネスエンティティとイベントについて、ロジックと検証をテストできます。

手順 1。カスタム検証のテスト

［作成］API を使用して、次の Person レコードを住所なしで作成します:

```
POST http://localhost:8080/cmx/cs/localhost-orcl-mdm_Sample/Person?systemName=Admin
{
        firstName: "John"
    }
```

検証エラーが表示されます。

［作成］API を使用して、次の Person レコードを住所ありで作成します:

```
POST http://localhost:8080/cmx/cs/localhost-orcl-mdm_Sample/Person?systemName=Admin
{
        firstName: "John",
        Addresses: {
            item: [
                {
                    cityName: "Toronto"
                }
            ]
        }
    }
```

要求で Person レコードが作成されます。

手順 2。カスタムロジックのテスト

次の手順を実行して、カスタムロジックをテストします。

1. ［作成］API を使用して、住所と電話番号を持つ 2 つの Person レコードを作成します。

```
POST http://localhost:8080/cmx/cs/localhost-orcl-mdm_sample/Person?systemName=Admin
    {
        firstName: "John",
        Addresses: {
            item: [
                {
                    cityName: "Toronto"
                }
            ]
        },
        TelephoneNumbers: {
            item:[
                {
                    phoneNum: "111-11-11"
                }
            ]
        }
    }
POST http://localhost:8080/cmx/cs/localhost-orcl-mdm_sample/Person?systemName=Admin
    {
        firstName: "John",
        Addresses: {
            item: [
                {
                    cityName: "Ottawa"
                }
            ]
        },
        TelephoneNumbers: {
            item:[
```

```
                {
                    phoneNum: "222-22-22"
                }
            ]
        }
    }
```

応答には、次の行 ID が含まれます。

- Person: 161923、161924
- 住所: 2123、2124
- 電話番号: 101723、101724

2. ［マージのプレビュー］API を実行して、両方の Person レコードをマージします。

```
POST http://localhost:8080/cmx/cs/localhost-orcl-mdm_sample/Person/161923?action=previewMerge&depth=2
{
        keys: [
            {
                rowid: "161924"
            }
        ]
    }
```

応答は、2 つの住所と 2 つの電話番号を持つ Person レコードです。


```
{
  "Person": {
    "rowidObject": "161923        ",
    "creator": "admin",
    "createDate": "2016-10-20T09:50:35.878-04:00",
    "updatedBy": "admin",
    "lastUpdateDate": "2016-10-20T09:50:35.879-04:00",
    "consolidationInd": 4,
    "lastRowidSystem": "SYSO          ",
    "hubStateInd": 1,
    "label": "Person: , Bill",
    "partyType": "Person",
    "displayName": "Bill",
    "firstName": "Bill",
   "TelephoneNumbers": {
      "link": [],
      "firstRecord": 1,
      "recordCount": 2,
      "pageSize": 2,
      "item": [
        {
          "rowidObject": "101723        ",
          "creator": "admin",
          "createDate": "2016-10-20T09:50:35.904-04:00",
          "updatedBy": "admin",
          "lastUpdateDate": "2016-10-20T09:50:35.905-04:00",
          "consolidationInd": 4,
          "lastRowidSystem": "SYSO          ",
          "hubStateInd": 1,
          "label": "PhoneNumbers",
          "phoneNum": "111-1111     ",
          "phoneCountryCd": "1"
        },
        {
          "rowidObject": "101724        ",
          "creator": "admin",
          "createDate": "2016-10-20T09:50:54.768-04:00",
          "updatedBy": "admin",
          "lastUpdateDate": "2016-10-20T09:50:54.769-04:00",
          "consolidationInd": 4,
          "lastRowidSystem": "SYSO          ",
          "hubStateInd": 1,
          "label": "PhoneNumbers",
          "phoneNum": "222-2222     ",
```

```
            "phoneCountryCd": "1"
          }
        ]
      },
      "Addresses": {
        "link": [],
        "firstRecord": 1,
        "recordCount": 2,
        "pageSize": 2,
        "item": [
          {
            "rowidObject": "2123          ",
            "creator": "admin",
            "createDate": "2016-10-20T09:50:37.956-04:00",
            "updatedBy": "admin",
            "lastUpdateDate": "2016-10-20T09:50:37.956-04:00",
            "consolidationInd": 4,
            "lastRowidSystem": "SYS0          ",
            "hubStateInd": 1,
            "label": "Addresses",
            "Address": {
              "rowidObject": "2121          ",
              "creator": "admin",
              "createDate": "2016-10-20T09:50:36.922-04:00",
              "updatedBy": "admin",
              "lastUpdateDate": "2016-10-20T09:50:37.923-04:00",
              "consolidationInd": 4,
              "lastRowidSystem": "SYS0          ",
              "hubStateInd": 1,
              "label": "Address",
              "cityName": "Toronto"
            }
          },
          {
            "rowidObject": "2124          ",
            "creator": "admin",
            "createDate": "2016-10-20T09:50:54.790-04:00",
            "updatedBy": "admin",
            "lastUpdateDate": "2016-10-20T09:50:54.790-04:00",
            "consolidationInd": 4,
            "lastRowidSystem": "SYS0          ",
            "hubStateInd": 1,
            "label": "Addresses",
            "Address": {
              "rowidObject": "2122          ",
              "creator": "admin",
              "createDate": "2016-10-20T09:50:54.777-04:00",
              "updatedBy": "admin",
              "lastUpdateDate": "2016-10-20T09:50:54.777-04:00",
              "consolidationInd": 4,
              "lastRowidSystem": "SYS0          ",
              "hubStateInd": 1,
              "label": "Address",
              "cityName": "Ottawa"
            }
          }
        ]
      }
    }
}
```


3. 住所と電話番号のマージのオーバーライドを指定して、［マージのプレビュー］API を実行して両方の Person レコードをマージします。

```
POST http://localhost:8080/cmx/cs/localhost-orcl-MDM_SAMPLE/Person/161923?action=previewMerge&depth=3
{
    keys: [
        { rowid: "161923" }
    ],
    overrides: {
        Person: {
```

```
            Addresses: {
                item:[
                    {
                        rowidObject: "2123",
                        MERGE: {
                            item:[{key:{rowid: "2124"}}],
                            $original: {
                                item:[null]
                            }
                        }
                    }
                ]
            },
            TelephoneNumbers: {
                item:[
                    {
                        rowidObject: "101723",
                        MERGE: {
                            item:[{key:{rowid: "101724"}}],
                            $original: {
                                item:[null]
                            }
                        }
                    }
                ]
            }
        }
    }
}
```

応答では、住所は 1 つですが電話番号が 2 つ表示されます。

```
{
  "Person": {
    "rowidObject": "161923        ",
    "creator": "admin",
    "createDate": "2016-10-20T09:50:35.878-04:00",
    "updatedBy": "admin",
    "lastUpdateDate": "2016-10-20T09:50:35.879-04:00",
    "consolidationInd": 4,
    "lastRowidSystem": "SYSO           ",
    "hubStateInd": 1,
    "label": "Person: , Bill",
    "partyType": "Person",
    "displayName": "Bill",
    "firstName": "Bill",
   "TelephoneNumbers": {
      "link": [],
      "firstRecord": 1,
      "recordCount": 2,
      "pageSize": 2,
      "item": [
        {
          "rowidObject": "101723        ",
          "creator": "admin",
          "createDate": "2016-10-20T09:50:35.904-04:00",
          "updatedBy": "admin",
          "lastUpdateDate": "2016-10-20T09:50:35.905-04:00",
          "consolidationInd": 4,
          "lastRowidSystem": "SYSO           ",
          "hubStateInd": 1,
          "label": "PhoneNumbers",
          "phoneNum": "111-1111      ",
          "phoneCountryCd": "1"
        },
        {
          "rowidObject": "101724        ",
          "creator": "admin",
          "createDate": "2016-10-20T09:50:54.768-04:00",
          "updatedBy": "admin",
          "lastUpdateDate": "2016-10-20T09:50:54.769-04:00",
```

```
            "consolidationInd": 4,
            "lastRowidSystem": "SYS0          ",
            "hubStateInd": 1,
            "label": "PhoneNumbers",
            "phoneNum": "222-2222     ",
            "phoneCountryCd": "1"
          }
        ]
      },
        "Addresses": {
        "link": [],
        "firstRecord": 1,
        "recordCount": 1,
        "pageSize": 1,
        "item": [
          {
            "rowidObject": "2123          ",
            "creator": "admin",
            "createDate": "2016-10-20T09:50:37.956-04:00",
            "updatedBy": "admin",
            "lastUpdateDate": "2016-10-20T09:50:37.956-04:00",
            "consolidationInd": 4,
            "lastRowidSystem": "SYS0          ",
            "hubStateInd": 1,
            "label": "Addresses"
          }
        ]
      },
      "PersonDetails": {
        "link": [],
        "recordCount": 0,
        "pageSize": 0,
        "item": []
      }
    }
  }
```


例: 外部呼び出しからサービス統合フレームワーク呼び出しを行う

セキュリティ上の理由から、MDM はパスワードとペイロードの情報を外部呼び出しに渡しません。BESExternalCallWithSIF の例は、外部呼び出しから Services Integration Framework（SIF）呼び出しを行う方法を示しています。

この例の外部呼び出しは、次のカスタム検証を実行します。

1. Person レコードの電話番号の内線番号に特殊文字が含まれているかどうかを確認します。
2. 電話番号に特殊文字が含まれている場合は、電話番号を修正します。
3. タイプ PRI の電話番号が 1 つだけかどうかを確認します。

外部呼び出しがタイプ PRI の電話番号が 1 つだけ存在すること検証した後に、外部呼び出しは SIF Put 呼び出しを使用して更新を実行します。

要件

外部サービスから MDM Hub への呼び出し用にアプリケーションユーザーを設定します。この例では、アプリケーションユーザーとして e360 を使用しています。このアプリケーションユーザーを、設定したユーザーに置き換えます。詳細については、『*Multidomain MDM のセキュリティガイド*』を参照してください。

外部呼び出しからの SIF 呼び出しの実行

1. bes-external-call.ear ファイルを作成してデプロイします。
2. プロビジョニングツールで、Web サービスを SOAP サービスとして登録します。
   次の表に、SOAP サービスに対して設定できるプロパティを示します。

プロパティ	値の例
名前	besexternal
URL	https://<host>:<port>/bes-external-call/CustomLogicService?wsdl

3. プロビジョニングツールで、外部呼び出しを登録します。
   次の表に、外部呼び出しに対して設定できるプロパティを示します。

プロパティ	値の例
SOAP 操作	validate
ビジネスエンティティ	Person
サービスフェーズ	WriteCO.BeforeValidate

   - 操作: validate
   - ビジネスエンティティ: Person
   - サービスフェーズ: WriteCO.BeforeValidate

4. 内線番号に特殊文字が含まれる電話番号を使用して Person レコードを作成します。
   次のサンプル POST 要求には、内線電話番号に特殊文字@が含まれています。

```
POST /cmx/cs/<ors>/Person?systemName=Admin
    {
        "firstName": "John",
        "TelephoneNumbers": {
            "item": [
                {
                    "phoneExtNum": "e@1",
                    "phoneNum": "1234567",
                    "phoneTypeCd":{
                       "phoneType":"PRI"
                     }
                }
            ]
        }
    }
```

   次の応答は、検証エラーを表示します。

```
<?xml version="1.0" encoding="UTF-8"?>
   <ns1:CsFault xmlns:ns1="urn:cs-base.informatica.mdm" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
       <ns1:errorMessage>SIP-50022: Validation failed.</ns1:errorMessage>
       <ns1:errorCode>SIP-50022</ns1:errorCode>
       <ns1:details xsi:type="ns1:ValidationErrors">
           <ns1:error>
               <ns1:code>CUSTOM-00001</ns1:code>
               <ns1:message>The Phone Ext Number has special symbols.</ns1:message>
               <ns1:field>Person.TelephoneNumbers.phoneExtNum</ns1:field>
           </ns1:error>
```

```
      </ns1:details>
   </ns1:CsFault>
```

次のサンプルは、検証エラーを生成しない有効な POST 要求です。

```
POST /cmx/cs/<ors>/Person?systemName=Admin
    {
        "firstName": "John",
        "TelephoneNumbers": {
            "item": [
                {
                    "phoneExtNum": "e1",
                    "phoneNum": "1234567",
                    "phoneTypeCd":{
                       "phoneType":"PRI"
                     }
                }
            ]
        }
    }
```

5. o などの文字を含む電話番号を使用して Person レコードを作成します。
   POST 要求が実行されると、文字 o が数字の 0 に置換されます。

   次のサンプル POST 要求には、文字 o を使用した電話番号が含まれています。

```
POST /cmx/cs/<ors>/Person?systemName=Admin
    {
        "firstName": "John",
        "TelephoneNumbers": {
            "item": [
                {
                    "phoneExtNum": "e1",
                    "phoneNum": "123o4567",
                    "phoneTypeCd":{
                       "phoneType":"PRI"
                     }
                }
            ]
        }
    }
```

6. POST 要求に 1 つまたは複数の電話レコードが含まれているかどうかに応じて、電話タイプの値がどのように設定されるかを示す次のシナリオを考えてみましょう。

   - シナリオ 1。このシナリオには、それぞれ 1 つの電話レコードを作成する 2 つの POST 要求が含まれます。次の POST 要求は、電話タイプが PRI の最初の電話レコードを作成します。

     ```
     POST /cmx/cs/<ors>/Person?systemName=Admin
         {
             "firstName": "John",
             "TelephoneNumbers": {
                 "item": [
                     {
                         "phoneExtNum": "e1",
                         "phoneNum": "12304567",
                         "phoneTypeCd":{
                            "phoneType":"PRI"
                          }
                     }
                 ]
             }
         }
     ```

     次の POST 要求を実行して 2 番目の電話レコードを作成すると、2 番目の電話レコードの電話タイプが PRI に設定され、最初の POST 要求の電話タイプが OTH に変更されます。

     ```
     POST /cmx/cs/<ors>/Person/rowidObject?systemName=Admin
         {
             "TelephoneNumbers": {
                 "item": [
     ```

```
                {
                    "phoneExtNum": "e2",
                    "phoneNum": "23045678",
                    "phoneTypeCd":{
                       "phoneType":"PRI"
                     }
                }
            ]
        }
    }
```

- シナリオ 2。このシナリオには、2 つの電話レコードを作成する 1 つの POST 要求が含まれます。次のサンプル POST 要求には、電話タイプが PRI の 2 つの電話レコードが含まれています。

```
POST /cmx/cs/<ors>/Person?systemName=Admin
    {
        "firstName": "John",
        "TelephoneNumbers": {
            "item": [
                {
                    "phoneExtNum": "e1",
                    "phoneNum": "12304567",
                    "phoneTypeCd":{
                       "phoneType":"PRI"
                     }
                },
                {
                    "phoneExtNum": "e2",
                    "phoneNum": "23045678",
                    "phoneTypeCd":{
                       "phoneType":"PRI"
                     }
                }
            ]
        }
    }
```

この POST 要求が実行されると、最初の電話レコードの電話タイプが PRI に設定されます。2 番目の電話レコードでは、電話タイプは OTH に設定されます。

付録 A

REST API を使用したレコードの追加

この付録では、以下の項目について説明します。

REST API を使用したレコードの追加の概要

ビジネスエンティティモデルを作成して、ビジネスエンティティ構造を設定したら、REST API を使用してレコードを追加できます。

次のセクションでは、Person ビジネスエンティティの例を使用して、REST API によるレコードの追加方法を説明します。Person ビジネスエンティティには、会社の従業員のデータが含まれます。

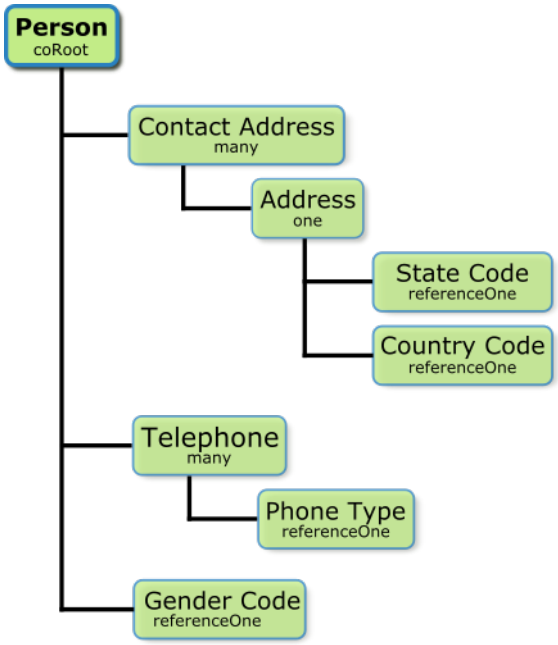
従業員の詳細を追加するには、次の API を使用します。

1. スキーマに関する情報を取得する。ビジネスエンティティのデータ構造に関する情報（構造、フィールドのリスト、フィールドタイプ、ルックアップフィールドの詳細など）を取得するには、［メタデータの取得］REST API を使用します。または、使用可能な要素および属性が記述された XML スキーマ定義（XSD）ファイルにアクセスすることもできます。XSD ファイルは、`http://<host>:<port>/cmx/csfiles` の場所にあります。
2. レコードを作成します。レコードを作成するには、［レコードの作成］REST API を使用します。
3. 追加したレコードからデータを読み取ります。レコードからデータを取得するには、［レコードの読み取り］REST API を使用します。

Person ビジネスエンティティの構造

REST API を使用して、Person レコードを追加します。Person ルートノードは、Person ビジネスエンティティ構造の最上位ノードです。Person ルートノードの下に、性別、住所、電話番号などの従業員の詳細に対応するノードがあります。

次の図は、Person ビジネスエンティティの構造を示しています。



Person は、Person ビジネスエンティティのルートノードです。ノード名の下に記載されているノードタイプは、親ノードと子ノード間のリレーションを示しています。Contact Address と Address 間には 1 対 1 のリレーションがあります。これは、各連絡先住所には、1 つの住所のみを関連付けられることを示しています。Person と Telephone 間には 1 対多のリレーションがあります。これは、Person レコードには、複数の電話番号レコードを関連付けられることを示しています。Person と Gender 間には 1 対 1 のリレーションがあります。これは、Person レコードには、1 つの性別の値のみを関連付けられることを示しています。性別の値は、ルックアップテーブルに存在します。同様に、州コードおよび国コードの値もルックアップテーブルに存在します。

手順 1. スキーマに関する情報の取得

スキーマに関する情報を取得するには、［メタデータの取得］REST API を使用します。［メタデータの取得］API は、ビジネスエンティティのデータ構造を返します。メタデータには、ビジネスエンティティフィールド、フィールドタイプ、およびルックアップフィールドの詳細が一覧表示されます。

［メタデータの取得］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/<business entity>?action=meta
```

次のサンプル要求は、Person ビジネスエンティティのメタデータ情報を取得します。

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person?action=meta
```

メタデータ応答の取得

次のサンプルは、Person ビジネスエンティティのデータ構造の抜粋を示しています。


```
{
    "object": {
        "field": [
            {
                "allowedValues": [
                    "Person"
                ],
                "name": "partyType",
                "label": "Party Type",
                "dataType": "String",
                "length": 255,
                "totalDigits": 0,
                "fractionDigits": 0,
                "readOnly": false,
                "required": false,
                "system": false
            },
            {
                "name": "imageUrl",
                "label": "Image URL",
                "dataType": "ImageURL",
                "length": 255,
                "totalDigits": 0,
                "fractionDigits": 0,
                "readOnly": false,
                "required": false,
                "system": false
            },
            {
                "name": "statusCd",
                "label": "Status Cd",
                "dataType": "String",
                "length": 2,
                "totalDigits": 0,
                "fractionDigits": 0,
                "readOnly": false,
                "required": false,
                "system": false
            },
            {
                "name": "displayName",
                "label": "Display Name",
                "dataType": "String",
                "length": 200,
                "totalDigits": 0,
                "fractionDigits": 0,
                "readOnly": false,
                "required": false,
                "system": false
            },
            {
                "name": "birthdate",
                "label": "Birthdate",
                "dataType": "Date",
                "length": 0,
                "totalDigits": 0,
                "fractionDigits": 0,
                "readOnly": false,
                "required": false,
                "system": false
            },
            {
                "name": "firstName",
                "label": "First Name",
                "dataType": "String",
                "length": 50,
                "totalDigits": 0,
```

```
        "fractionDigits": 0,
        "readOnly": false,
        "required": false,
        "system": false
    },
    {
        "name": "lastName",
        "label": "Last Name",
        "dataType": "String",
        "length": 50,
        "totalDigits": 0,
        "fractionDigits": 0,
        "readOnly": false,
        "required": false,
        "system": false
    },
    {
        "name": "middleName",
        "label": "Middle Name",
        "dataType": "String",
        "length": 50,
        "totalDigits": 0,
        "fractionDigits": 0,
        "readOnly": false,
        "required": false,
        "system": false
    },
    {
        "name": "dirtyIndicator",
        "label": "Dirty Indicator",
        "dataType": "Integer",
        "length": 38,
        "totalDigits": 0,
        "fractionDigits": 0,
        "readOnly": true,
        "required": false,
        "system": true
    },
    {
        "name": "hubStateInd",
        "label": "Hub State Ind",
        "dataType": "Integer",
        "length": 38,
        "totalDigits": 0,
        "fractionDigits": 0,
        "readOnly": true,
        "required": false,
        "system": true
    },
    {
        "name": "cmDirtyInd",
        "label": "Content metadata dirty Ind",
        "dataType": "Integer",
        "length": 38,
        "totalDigits": 0,
        "fractionDigits": 0,
        "readOnly": true,
        "required": false,
        "system": true
    },
    {
        "name": "lastRowidSystem",
        "label": "Last Rowid System",
        "dataType": "String",
        "length": 14,
        "totalDigits": 0,
        "fractionDigits": 0,
        "readOnly": true,
        "required": false,
        "system": true
    },
```

```
    ------------------------------------------------------------------------------------------
            {
                "name": "genderCd",
                "label": "Gender Cd",
                "dataType": "lookup",
                "readOnly": false,
                "required": false,
                "system": false,
                "lookup": {
                    "link": [
                        {
                            "href": "http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/LUGender?
action=list&idlabel=genderCode%3AgenderDisp",
                            "rel": "lookup"
                        },
                        {
                            "href": "http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/LUGender?
action=list",
                            "rel": "list"
                        }
                    ],
                    "object": "LUGender",
                    "key": "genderCode",
                    "value": "genderDisp"
                }
            }
        ],

    ------------------------------------------------------------------------------------------
                "child": [
                    {
                        "field": [
                            {
                                "name": "cityName",
                                "label": "City Name",
                                "dataType": "String",
                                "length": 100,
                                "totalDigits": 0,
                                "fractionDigits": 0,
                                "readOnly": false,
                                "required": false,
                                "system": false
                            },
                            {
                                "name": "addressLine2",
                                "label": "Address Line2",
                                "dataType": "String",
                                "length": 100,
                                "totalDigits": 0,
                                "fractionDigits": 0,
                                "readOnly": false,
                                "required": false,
                                "system": false
                            },
                            {
                                "name": "addressLine1",
                                "label": "Address Line1",
                                "dataType": "String",
                                "length": 100,
                                "totalDigits": 0,
                                "fractionDigits": 0,
                                "readOnly": false,
                                "required": false,
                                "system": false
                            },
                            {
                                "name": "isValidInd",
                                "label": "Is Valid Ind",
                                "dataType": "String",
                                "length": 1,
                                "totalDigits": 0,
```

```
                            "fractionDigits": 0,
                            "readOnly": false,
                            "required": false,
                            "system": false
                        },
                        {
                            "name": "postalCd",
                            "label": "Postal Cd",
                            "dataType": "String",
                            "length": 10,
                            "totalDigits": 0,
                            "fractionDigits": 0,
                            "readOnly": false,
                            "required": false,
                            "system": false
                        },

-------------------------------------------------------------------------------

                        {
                            "name": "countryCode",
                            "label": "Country Code",
                            "dataType": "lookup",
                            "readOnly": false,
                            "required": false,
                            "system": false,
                            "dependents": [
                                "Person.Address.Address.stateCd"
                            ],
                            "lookup": {
                                "link": [
                                    {
                                        "href": "http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/
LUCountry?action=list",
                                        "rel": "list"
                                    },
                                    {
                                        "href": "http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/
LUCountry?action=list&idlabel=countryCode%3AcountryNameDisp",
                                        "rel": "lookup"
                                    }
                                ],
                                "object": "LUCountry",
                                "key": "countryCode",
                                "value": "countryNameDisp"
                            }
                        },
                        {
                            "name": "stateCd",
                            "label": "State Cd",
                            "dataType": "lookup",
                            "readOnly": false,
                            "required": false,
                            "system": false,
                            "parents": [
                                "Person.Address.Address.countryCode"
                            ],
                            "lookup": {
                                "link": [
                                    {
                                        "href": "http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/
LUCountry/{Person.Address.Address.countryCode}/LUState?action=list",
                                        "rel": "list"
                                    },
                                    {
                                        "href": "http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/
LUCountry/{Person.Address.Address.countryCode}/LUState?action=list&idlabel=stateAbbreviation%3AstateNameDisp",
                                        "rel": "lookup"
                                    }
                                ],
```

```
                                    "object": "LUCountry.LUState",
                                    "key": "stateAbbreviation",
                                    "value": "stateNameDisp"
                                }
                            }
                        ],
                        "name": "Address",
                        "label": "Address",
                        "many": false
                    }
                ],
                "name": "Address",
                "label": "Contact Address",
                "many": true
            },
            {
                "field": [
                    {
                        "name": "phoneNum",
                        "label": "Phone Number",
                        "dataType": "String",
                        "length": 13,
                        "totalDigits": 0,
                        "fractionDigits": 0,
                        "readOnly": false,
                        "required": false,
                        "system": false
                    },
```

```
                    {
                        "name": "phoneTypeCd",
                        "label": "Phone Type",
                        "dataType": "lookup",
                        "readOnly": false,
                        "required": false,
                        "system": false,
                        "lookup": {
                            "link": [
                                {
                                    "href": "http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/LUPhoneType?
action=list&idlabel=phoneType%3AphoneTypeDisp",
                                    "rel": "lookup"
                                },
                                {
                                    "href": "http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/LUPhoneType?
action=list",
                                    "rel": "list"
                                }
                            ],
                            "object": "LUPhoneType",
                            "key": "phoneType",
                            "value": "phoneTypeDisp"
                        }
                    }
                ],
                "name": "Telephone",
                "label": "Telephone",
                "many": true
            }
        ],
        "name": "Person",
        "label": "Person",
        "many": false
    }
}
```

手順 2。レコードの作成

レコードを作成するには、［レコードの作成］REST API を使用します。ビジネスエンティティの名前とソースシステムの名前は必須パラメータです。レコードのデータを要求本文で送信します。

［レコードの作成］URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/<business entity>?systemName=<name of the source system>
```

systemName パラメータは、必須パラメータで、ソースシステムの名前を指定します。

Person ビジネスエンティティには、Person ルートノード、および第 2 レベルの address、gender、phone のノードがあります。

次のサンプル要求は、Person レコードを作成します:

```
POST http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/Person?systemName=Admin
{
    "firstName": "Boris",
    "lastName": "Isaac",
    "genderCd": {
        "genderCode": "M"
    },
    "Address": {
        "item": [
            {
                "Address": {
                    "addressLine1": "B-203, 101 Avenue, New York",
                    "stateCd": {
                        "stateAbbreviation": "NY"
                    },
                    "countryCode": {
                        "countryCode": "US"
                    }
                }
            }
        ]
    },
    "Telephone": {
        "item": [
            {
                "phoneNum": "1234567",
                "phoneTypeCd": {
                    "phoneType": "HOM"
                }
            },
            {
                "phoneNum": "7654321",
                "phoneTypeCd": {
                    "phoneType": "MOB"
                }
            }
        ]
    }
}
```

要求本文では、Person レコードの次の詳細を指定します:

- First name。
- Last name。
- Gender。
- Address（State Code と Country Code を含む）。
- Phone Number と Phone Type（自宅電話や携帯電話など）。

レコード応答の作成

次のサンプル応答は、Person レコードが正常に作成された後の応答を示しています。


```
{
    "Person": {
        "key": {
            "rowid": "658248",
            "sourceKey": "66240000025000"
        },
        "rowidObject": "658248",
        "genderCd": {
            "key": {
                "rowid": "2             "
            },
            "rowidObject": "2              "
        },
        "Address": {
            "link": [],
            "item": [
                {
                    "key": {
                        "rowid": "101526",
                        "sourceKey": "66240000028000"
                    },
                    "rowidObject": "101526",
                    "Address": {
                        "key": {
                            "rowid": "121506",
                            "sourceKey": "66240000027000"
                        },
                        "rowidObject": "121506",
                        "countryCode": {
                            "key": {
                                "rowid": "233           "
                            },
                            "rowidObject": "233            "
                        },
                        "stateCd": {
                            "key": {
                                "rowid": "52            "
                            },
                            "rowidObject": "52             "
                        }
                    }
                }
            ]
        },
        "Telephone": {
            "link": [],
            "item": [
                {
                    "key": {
                        "rowid": "20967",
                        "sourceKey": "66240000029000"
                    },
                    "rowidObject": "20967",
                    "phoneTypeCd": {
                        "key": {
                            "rowid": "8             "
                        },
                        "rowidObject": "8              "
                    }
                },
                {
                    "key": {
                        "rowid": "20968",
                        "sourceKey": "66240000030000"
                    },
```

```
                "rowidObject": "20968",
                "phoneTypeCd": {
                    "key": {
                        "rowid": "6             "
                    },
                    "rowidObject": "6             "
                }
            }
        ]
    }
  }
}
```

注: 応答本文には、生成された行 ID を持つレコードが含まれています。

レコードを作成するときにワークフロープロセスが開始されるように設定すると、次のことが起こります。

- レコードが保留状態で作成されます。
- ワークフロープロセスが開始します。
- ワークフロープロセス ID が応答ヘッダに返されます。

ワークフロープロセスを設定しない場合、レコードはデフォルトでアクティブ状態で作成されます。

相互作用 ID を使用して応答を処理する場合、API により、相互作用 ID が応答ヘッダに返されます。

手順 3。レコードの読み取り

追加したルートレコードの詳細を取得するには、[レコードの読み取り] REST API を使用します。この API は、ルートレコードの子レコードの詳細を取得するために使用できます。

[レコードの読み取り] URL の形式は次のとおりです。

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<rowId of the root record>
```

返す子レベルの数を指定するには、depth パラメータを使用します。ルートノードとその直接の子を返すには 2 を指定し、ルートノード、直接の子、および孫を返すには 3 を指定します。子レコードの詳細を返すには、次の URL を使用します。

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<rowId of the record>?depth=n
```

次のサンプル要求は、ルートノード、直接の子、および孫の詳細を返します:

```
GET http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/Person/658248?depth=3
```

この要求は、アクティブなレコードの詳細を返します。

注: レコードの作成時にワークフローが開始されると、作成しているレコードが保留状態になります。デフォルトでは、アクティブなレコードが [レコードの読み取り] 要求で読み取られます。レコードの保留状態を指定するには、recordStates パラメータを使用します。

次のサンプル要求は、保留中のレコードの詳細を読み取ります。

```
GET http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/Person/658248?depth=3&recordStates=PENDING
```

レコード応答の読み取り

次のサンプル応答は、追加したレコードの詳細を示しています:

```
{
    "link": [
        {
            "href": "http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/Person/658248",
```

```
            "rel": "self"
        },
        {
            "href": "http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/Person/658248?depth=2",
            "rel": "children"
        }
    ],
    "rowidObject": "658248        ",
    "label": "Person",
    "partyType": "Person",
    "displayName": "BORIS ISAAC",
    "firstName": "BORIS",
    "lastName": "ISAAC",
    "genderCd": {
        "link": [
            {
                "href": "http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/Person/658248/genderCd/2",
                "rel": "self"
            },
            {
                "href": "http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/Person/658248",
                "rel": "parent"
            },
            {
                "href": "http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/Person/658248/genderCd/2?
depth=2",
                "rel": "children"
            }
        ],
        "rowidObject": "2             ",
        "label": "LU Gender",
        "genderCode": "M",
        "genderDisp": "MALE"
    },
    "Address": {
        "link": [
            {
                "href": "http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/Person/658248/Address",
                "rel": "self"
            },
            {
                "href": "http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/Person/658248",
                "rel": "parent"
            }
        ],
        "firstRecord": 1,
        "pageSize": 10,
        "searchToken": "SVR1.PCWJ",
        "item": [
            {
                "link": [
                    {
                        "href": "http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/Person/658248/Address",
                        "rel": "parent"
                    },
                    {
                        "href": "http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/Person/658248/Address/
101526?depth=2",
                        "rel": "children"
                    },
                    {
                        "href": "http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/Person/658248/Address/
101526",
                        "rel": "self"
                    }
                ],
                "rowidObject": "101526        ",
                "label": "Contact Address",
                "Address": {
                    "link": [
                        {
```

```
                            "href": "http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/Person/658248/
Address/101526/Address/121506?depth=2",
                            "rel": "children"
                        },
                        {
                            "href": "http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/Person/658248/
Address/101526",
                            "rel": "parent"
                        },
                        {
                            "href": "http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/Person/658248/
Address/101526/Address/121506",
                            "rel": "self"
                        }
                    ],
                    "rowidObject": "121506        ",
                    "label": "Address",
                    "addressLine1": "B-203, 101 Avenue, New York",
                    "countryCode": {
                        "link": [
                            {
                                "href": "http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/Person/658248/
Address/101526/Address/121506/countryCode",
                                "rel": "self"
                            },
                            {
                                "href": "http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/Person/658248/
Address/101526/Address/121506",
                                "rel": "parent"
                            },
                            {
                                "href": "http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/Person/658248/
Address/101526/Address/121506/countryCode?depth=2",
                                "rel": "children"
                            }
                        ],
                        "countryCode": "US"
                    },
                    "stateCd": {
                        "link": [
                            {
                                "href": "http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/Person/658248/
Address/101526/Address/121506",
                                "rel": "parent"
                            },
                            {
                                "href": "http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/Person/658248/
Address/101526/Address/121506/stateCd?depth=2",
                                "rel": "children"
                            },
                            {
                                "href": "http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/Person/658248/
Address/101526/Address/121506/stateCd",
                                "rel": "self"
                            }
                        ],
                        "stateAbbreviation": "NY"
                    }
                }
            }
        ]
    },
    "Telephone": {
        "link": [
            {
                "href": "http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/Person/658248",
                "rel": "parent"
            },
            {
                "href": "http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/Person/658248/Telephone",
                "rel": "self"
```

```
            }
        ],
        "firstRecord": 1,
        "pageSize": 10,
        "searchToken": "SVR1.PCWK",
        "item": [
            {
                "link": [
                    {
                        "href": "http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/Person/658248/Telephone",
                        "rel": "parent"
                    },
                    {
                        "href": "http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/Person/658248/
Telephone/20967",
                        "rel": "self"
                    },
                    {
                        "href": "http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/Person/658248/
Telephone/20967?depth=2",
                        "rel": "children"
                    }
                ],
                "rowidObject": "20967          ",
                "label": "Telephone",
                "phoneNum": "1234567      ",
                "phoneTypeCd": {
                    "link": [
                        {
                            "href": "http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/Person/658248/
Telephone/20967",
                            "rel": "parent"
                        },
                        {
                            "href": "http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/Person/658248/
Telephone/20967/phoneTypeCd/8",
                            "rel": "self"
                        },
                        {
                            "href": "http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/Person/658248/
Telephone/20967/phoneTypeCd/8?depth=2",
                            "rel": "children"
                        }
                    ],
                    "rowidObject": "8              ",
                    "label": "LU Phone Type",
                    "phoneTypeDisp": "HOME",
                    "phoneType": "HOM"
                }
            },
            {
                "link": [
                    {
                        "href": "http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/Person/658248/
Telephone/20968",
                        "rel": "self"
                    },
                    {
                        "href": "http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/Person/658248/
Telephone/20968?depth=2",
                        "rel": "children"
                    },
                    {
                        "href": "http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/Person/658248/Telephone",
                        "rel": "parent"
                    }
                ],
                "rowidObject": "20968          ",
                "label": "Telephone",
                "phoneNum": "7654321      ",
                "phoneTypeCd": {
```

```
                    "link": [
                        {
                            "href": "http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/Person/658248/
Telephone/20968/phoneTypeCd/6",
                            "rel": "self"
                        },
                        {
                            "href": "http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/Person/658248/
Telephone/20968",
                            "rel": "parent"
                        },
                        {
                            "href": "http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/Person/658248/
Telephone/20968/phoneTypeCd/6?depth=2",
                            "rel": "children"
                        }
                    ],
                    "rowidObject": "6             ",
                    "label": "LU Phone Type",
                    "phoneTypeDisp": "MOBILE",
                    "phoneType": "MOB"
                }
            }
        ]
    }
}
```

付録 B

REST API を使用したファイルのアップロード

この付録では、以下の項目について説明します。

REST API を使用したファイルのアップロードの概要

REST API を使用すると、ファイルをストレージタイプにアップロードできます。ファイルをアップロードした後、そのファイルをレコードまたはタスクに添付したり、そのファイルを使用して Data Director ユーザーインタフェースをローカライズすることができます。

ファイルの使用方法に基づいて、使用する REST API、ファイルコンポーネント、およびストレージタイプの組み合わせが異なる場合があります。例えば、レコードまたはタスクにファイルを添付するには、ファイルのメタデータを作成し、そのファイルを一時ストレージにアップロードします。ファイルをアップロードした後、そのファイルをデータベース内のレコードに添付したり、そのファイルを BPM ストレージ内のタスクに添付することができます。Data Director ユーザーインタフェースをローカライズするには、ZIP ファイルをダウンロードし、圧縮されたファイルを変更してから、変更した ZIP バンドルをバンドルストレージにアップロードします。

REST API ファイルの

一連の汎用 REST API を使用して、添付またはローカライズ用のファイルをアップロードおよび管理できます。

次の表に、ファイルの REST API を示します。

REST API	**説明**	**サポートされているストレージタイプ**
ファイルのメタデータの一覧表示	ストレージ内のファイルのメタデータのリストを返します。	BPM または TEMP
ファイルのメタデータの作成	ストレージ内のファイルのメタデータを作成します。	DB または TEMP
ファイルのメタデータの取得	ファイルのメタデータを返します。	BPM、BUNDLE、DB、または TEMP
ファイルのメタデータの更新	ファイルのメタデータを更新します。	DB または TEMP
ファイルコンテンツのアップロード	ファイルのコンテンツをストレージにアップロードします。	BUNDLE、DB、または TEMP
ファイルコンテンツの取得	ファイルのコンテンツをダウンロードします。	BPM、BUNDLE、DB、または TEMP
ファイルの削除	ファイルのメタデータやコンテンツなど、ファイルに関連付けられているコンポーネントを含む、ストレージ内のファイルを削除します。	BUNDLE、DB、または TEMP

ファイルコンポーネント

ファイルをレコードまたはタスクに添付するには、ファイルのメタデータを作成してから、ファイルコンテンツをアップロードします。Data Director ユーザーインタフェースをローカライズするには、リソースバンドルファイルをダウンロードしてから、変更されたリソースバンドルファイルをアップロードします。

ファイルのメタデータ

ファイルに関する情報（ファイル名、ファイルタイプ、コンテンツタイプなど）。ストレージタイプに応じて、その他のパラメータ（作成者、作成時間、アップロード日など）を含める必要があります。

ファイルコンテンツ

ファイルのコンテンツ。例えば、テキスト、イメージ、ドキュメント、またはリソースバンドル。

ストレージタイプ

サポートされているストレージ実装にファイルをアップロードして保存します。使用するストレージタイプは、Data Director ユーザーインタフェースをローカライズするか、ファイルをレコードまたはタスクに添付するかによって異なります。

次のリストにサポートされているストレージタイプを示します。

BPM

タスクに添付されたファイルをタスクデータとともに保存します。ファイルをタスクに添付すると、プロセスによりファイルが TEMP ストレージから BPM ストレージに保存されます。

BPM ストレージに保存されているファイルは、次のファイル ID 形式を使用します: `taskId::filename`。

注: ファイルをトリガされたタスクまたは既存のタスクに添付するには、プロビジョニングツールで、タスクトリガ、タスクタイプ、およびタスクアクションの添付ファイルを有効にします。詳細については、『*Multidomain MDM のプロビジョニングツールガイド*』を参照してください。

BUNDLE

Data Director ユーザーインタフェースをローカライズするリソースバンドルファイルを保存します。

BUNDLE ストレージに保存されているファイルは、次のファイル ID 形式を使用します: `besMetadata`。

DB

レコードの添付ファイルを C_REPOS_ATTACHMENTS テーブルに保存します。ファイルをレコードに添付すると、プロセスによりファイルが TEMP ストレージから DB ストレージに保存されます。

DB ストレージに保存されているファイルは、次のファイル ID 形式を使用します: `DB_<RowID>`。

注: ファイルをレコードに添付するには、プロビジョニングツールで、データ型に `FileAttachment` を指定してフィールドを設定します。データ型の設定の詳細については、『*Multidomain MDM のプロビジョニングツールガイド*』を参照してください。

TEMP

C_REPOS_ATTACHMENTS テーブルにファイルを一時的に保存し、ファイルを TEMP としてマークします。ファイルは、BPM または DB ストレージに正常にアップロードされた後、または事前設定された有効期限が過ぎた後、TEMP ストレージから削除されます。

TEMP ストレージに保存されているファイルは、次のファイル ID 形式を使用します: `TEMP_<ROWID_ATTACHMENT>`。

有効期限の設定の詳細については、『*Multidomain MDM の設定ガイド*』を参照してください。

レコードへのファイルの添付

ファイルをレコードに添付する前に、そのファイルのメタデータを作成してから、そのファイルを一時ストレージにアップロードしてください。

1. ファイルのメタデータを作成するには、ストレージタイプに TEMP を指定した［ファイルのメタデータの作成］REST API を使用します。
   例えば、次の要求は `Document_3.pdf` ファイルのメタデータを作成します:

```
POST http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/TEMP
Content-Type: application/json
{
    "fileName": "Document_3.pdf",
```

```
    "fileType": "pdf",
    "fileContentType": "application/pdf"
}
```

注: 常に TEMP ストレージにファイルのメタデータを作成してください。

［ファイルのメタデータの作成］REST API はファイルの ID を返します。ファイル ID の形式は次のとおりです: <Storage Type>_<RowID>。RowID は、ストレージにアップロードするファイルの行 ID を表します。

この例では、API 呼び出しによって Document_3.pdf ファイルの次の ID が返されます: TEMP_SVR1.OJU3

ファイル ID を使用して、ファイルのアップロード、添付、更新、ダウンロード、および削除を行うことができます。

2. ファイルをアップロードするには、ストレージタイプに TEMP を指定した［ファイルコンテンツのアップロード］REST API を使用します。
例えば、次の要求はファイルを TEMP ストレージにアップロードします。

```
PUT http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/TEMP/TEMP_SVR1.OJU3/content
Content-Type: application/octet-stream
<file object (upload using REST client)>
```

注: ファイルをアップロードした後、TEMP ストレージには事前設定された期間である 60 分間、ファイルが保存されます。事前設定された期間が終了する前に、ファイルをレコードに添付する必要があります。

3. レコードを作成して、ファイルを新しいレコードに添付するには、［レコードの作成］REST API を使用します。

例えば、次の要求はレコードを作成し、ファイル ID が TEMP_SVR1.OJU3 のファイルを添付します。

```
POST http://localhost:8080/cmx/cs/localhost-orcl-MDM_SAMPLE/Person?systemName=Admin
Content-Type: application/json
{
   "frstNm":"John",
   "lstNm":"Smith",
   "addrLn1":"2100 Breverly Road",
   "addrTyp":{
      "addrTyp":"Billing",
      "addrTypDesc":"Billing"
   },
   "cntryCd":{
      "cntryCd":"AX",
      "cntryDesc":"Aland"
   },
   "attachments":{
      "item":[
         {
            "fileId":"TEMP_SVR1.OJU3"
         }
      ]
   }
}
```

注: ファイルをレコードに添付すると、プロセスによりファイルがデータベースに保存されます。ファイルの ID は DB_<RowID>に変更されます。ここで、DB はファイルがデータベースに保存されていることを示します。

4. レコードに添付されたファイルを置き換えるには、ストレージタイプに DB を指定した［ファイルコンテンツのアップロード］REST API を使用します。
例えば、次の要求は、データベース内でファイル ID が DB_SVR1.OJU3 の添付ファイルを置き換えます。

```
PUT http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/DB/DB_SVR1.OJU3/content
Content-Type: application/octet-stream
<file object (upload using REST client)>
```

注: 要求 URL のストレージタイプは DB です。

5. ファイルをレコードに添付した後、ファイルのメタデータを編集するには、ストレージタイプに DB を指定した［ファイルのメタデータの更新］REST API を使用します。

例えば、次の要求は、DB ストレージ内でファイル ID DB_SVR1.0JU3 に関連付けられているファイルのメタデータを更新します。

```
PUT http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/DB/DB_SVR1.0JU3
Content-Type: application/json
{
    "fileName": "Document_4.pdf",
    "fileType": "pdf",
    "fileContentType": "application/pdf"
}
```

6. レコードに添付されたファイルをダウンロードするには、ストレージタイプに DB を指定した［ファイルコンテンツの取得］REST API を使用します。
   例えば、次の要求は、ファイル ID DB_SVR1.0JU3 に関連付けられているファイルを DB ストレージからダウンロードします:

   ```
   GET http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/DB/DB_SVR1.0JU3/content
   ```

7. レコードに添付されたファイルを削除するには、ストレージタイプに DB を指定した［ファイルの削除］REST API を使用します。
   例えば、次の要求は、ファイル ID DB_SVR1.0JU3 に関連付けられているファイルを DB ストレージから削除します。

   ```
   DELETE http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/DB/DB_SVR1.0JU3
   ```

タスクへのファイルの添付

ファイルのメタデータを作成してから、ファイルコンテンツを一時ストレージにアップロードします。ファイルをアップロードした後、そのファイルをトリガされたタスクまたは既存のタスクに添付します。

注: ファイルをトリガされたタスクまたは既存のタスクに添付するには、プロビジョニングツールで、タスクトリガ、タスクタイプ、およびタスクアクションの添付ファイルを有効にします。詳細については、『*Multidomain MDM のプロビジョニングツールガイド*』を参照してください。

1. ファイルのメタデータを作成するには、ストレージタイプに TEMP を指定した［ファイルのメタデータの作成］REST API を使用します。
   例えば、次の要求は file1.txt ファイルのメタデータを作成します。

   ```
   POST http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/TEMP

   {
       “fileName”: “file1.txt”,
       “fileType”: “text”,
       “fileContentType”: “text/plain”
   }
   ```

   注: 常に TEMP ストレージにファイルのメタデータを作成してください。

   ［ファイルのメタデータの作成］REST API はファイルの ID を返します。ファイル ID の形式は次のとおりです: <Storage Type>_<RowID>。RowID は、ストレージにアップロードするファイルの行 ID を表します。この例では、API 呼び出しによって file1.txt ファイルの次の ID が返されます: TEMP_SVR1.1VDVS

   ファイル ID を使用して、ファイルのアップロード、添付、更新、および削除を行うことができます。

2. ファイルをアップロードするには、ストレージタイプに TEMP を指定した［ファイルコンテンツのアップロード］REST API を使用します。
   例えば、次の要求はファイルを TEMP ストレージにアップロードします:

   ```
   PUT http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/TEMP/TEMP_SVR1.1VDVS/content

   Test attachment content: file 1
   ```

注: ファイルをアップロードした後、TEMP ストレージには事前設定された期間である 60 分間、ファイルが保存されます。事前設定された期間が終了する前に、ファイルをタスクに添付する必要があります。

3. レコードを管理する際にトリガされるタスクにファイルを添付します。

- レコードを作成する際にトリガされるタスクにファイルを添付するには、taskattachments パラメータを指定した［ビジネスエンティティの作成］REST API を使用します。
  例えば、次の要求はレコードを作成し、ファイル ID が TEMP_SVR1.1VDVS のファイルを添付します:

```
POST http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person?
systemName=Admin&taskAttachments=TEMP_SVR1.1VDVS
Content-Type: application/json

{
     firstName: "John",
     lastName: "Smith",
     Phone: {
          item: [
               {
                    phoneNumber: "111-11-11"
               }
          ]
     }
}
```

- レコードを更新する際にトリガされるタスクにファイルを添付するには、taskattachments パラメータを指定した［ビジネスエンティティの更新］REST API を使用します。
  例えば、次の要求はレコードを更新し、ファイル ID が TEMP_SVR1.1VDVS のファイルを添付します:

```
PUT http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/233?
systemName=Admin&taskAttachments=TEMP_SVR1.1VDVS
{
    rowidObject: "233",
    firstName: "BOB",
    lastName: "LLOYD",
    Phone: {
        item: [
            {
                rowidObject: "164",
                phoneNumber: "777-77-77",
                $original: {
                    phoneNumber: "(336)366-4936"
                }
            }
        ]
    },
    $original: {
        firstName: "DUNN"
    }
}
```

- レコードをマージする際にトリガされるタスクにファイルを添付するには、taskattachments パラメータを指定した［ビジネスエンティティのマージ］REST API を使用します。
  例えば、次の要求はレコードをマージし、ファイル ID が TEMP_SVR1.1VDVS のファイルを添付します:

```
POST http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/2478245?
action=merge&taskAttachments=TEMP_SVR1.1VDVS
Content-Type: application/<json/xml>
{
  keys: [
    {
        rowid: "2478246"
    }
  ],
  overrides: {
    Person: {
       firstName: "Charlie"
    }
```

```
  }
}
```

- レコードをマージ解除する際にトリガされるタスクにファイルを添付するには、taskattachments パラメータを指定した［ビジネスエンティティのマージ解除］REST API を使用します。
  例えば、次の要求はレコードをマージ解除し、ファイル ID が TEMP_SVR1.1VDVS のファイルを添付します:

```
POST http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/2478248?
action=unmerge&taskAttachments=TEMP_SVR1.1VDVS
{
        rowid: "4880369"
}
```

4. 既存のタスクにファイルを添付します。

- タスクを更新する際にファイルを添付するには、要求本文で attachments を指定した［タスクの更新］REST API を使用します。
  例えば、次の要求はタスクを更新し、ファイル ID が TEMP_SVR1.1VDVS のファイルを添付します:

```
PUT http://localhost:8080/cmx/cs/localhost-orcl-MDM_SAMPLE/task/urn:b4p2:15934
{
     taskType: {
          name: "UpdateWithApprovalWorkflow"
     },
     taskId: "urn:b4p2:15934",
     owner: "John",
     title: "Smoke test task - updated",
     comments: "Smoke testing - updated",
     "attachments": [
          {
               "id": "TEMP_SVR1.1VDVS"
          }
     ],
     ...
}
```

- タスクアクションを実行する際にファイルを添付するには、要求本文で attachments を指定した［タスクアクションの実行］REST API を使用します。
  例えば、次の要求はタスクアクションを実行し、ファイル ID が TEMP_SVR1.1VDVS のファイルを添付します:

```
POST http://localhost:8080/cmx/cs/localhost-orcl-MDM_SAMPLE/task/urn:b4p2:15934?taskAction=Cancel
{
     taskType: {
          name:"UpdateWithApprovalWorkflow",
          taskAction: [{name: "Cancel"}]
     },
     taskId: "urn:b4p2:15934",
     owner: "manager",
     title: "Smoke test task 222",
     comments: "Smoke testing",
     "attachments": [
          {
               "id": "TEMP_SVR1.1VDVS"
          }
     ],
     ...
}
```

ファイルをタスクに添付すると、プロセスによりファイルは TEMP ストレージから移動され、ファイルがタスクデータとともに BPM ストレージに保存されます。ファイルの ID は taskId::filename に変更されます。

リソースバンドルファイルのアップロード

Data Director ユーザーインタフェースをローカライズするには、リソースバンドル ZIP ファイルをダウンロードし、ZIP ファイル内のファイルを変更してから、変更した ZIP ファイルをバンドルストレージにアップロードします。

1. リソースバンドル ZIP ファイルをダウンロードするには、ストレージタイプに BUNDLE を指定した［ファイルコンテンツの取得］REST API を使用します。
   例えば、次の要求はリソースバンドル ZIP ファイルをダウンロードします:

   ```
   GET http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/BUNDLE/besMetadata/content
   ```

2. 言語固有のバンドルファイルを追加して、ZIP ファイルを変更します。
   例えば、フィールド名、ラベル、テーブル名をロシア語にローカライズするには、`besMetadata_ru.properties` ファイルを追加します。

3. 変更したリソースバンドル ZIP ファイルをアップロードするには、ストレージタイプに BUNDLE を指定した［ファイルコンテンツのアップロード］REST API を使用します。
   例えば、次の要求はリソースバンドル ZIP ファイルをアップロードします:

   ```
   PUT http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/BUNDLE/besMetadata/content
   Content-Type: application/octet-stream
   Body: binary stream – zip file with besMetadata bundle
   ```

付録 C

REST API を使用したレポートの管理

この付録では、以下の項目について説明します。

REST API を使用したレポートの管理の概要

REST API を使用して、マスタデータに関する情報を収集する追加設定なしで使用できるレポートとカスタムレポートを管理できます。後で、プロビジョニングツールで、チャートコンポーネントにレポートデータを入力できます。その後 Data Director で、ユーザーがチャートを表示し、マスタデータに関するデータを分析できます。

追加設定なしで使用できるレポートは、事前定義された条件に関する情報を収集します。マスタデータに関する他の情報を収集する場合は、カスタムレポートを使用します。

レポートの REST API

REST API のセットを使用して、レポートを管理できます。

次の表に、レポートの REST API を示します。

REST API	**説明**
レポートの一覧表示	登録済みレポートのリストとレポートの設定を返します。
レポート設定およびデータの取得	レポート設定およびデータを返します。
レポート設定およびドリルダウンレポートの取得	レポート設定と関連するドリルダウンレポートを返します。
登録レポート	カスタムレポートを登録します。他の API で使用できるレポートのレポート ID を返します。
レポート設定の更新	レポートの設定を更新します。
レポートデータの追加または更新	レポートのデータエントリを追加または更新します。
レポートの削除	レポートを削除します。
レポート更新ジョブの実行	追加設定なしで使用できるレポートに関連付けられているバッチジョブを開始します。
レポートの更新ジョブのステータスの取得	レポート更新ジョブのステータスを返します。

レポート設定

レポート設定は、レポート名、説明、次元、メトリック名などの複数のパラメータで構成されます。追加設定なしで使用できるレポートのレポート設定は事前設定されています。カスタムレポートを使用する場合は、カスタムレポートのレポート設定を指定する必要があります。

以下の表に、レポート設定内のパラメータを示します。

パラメータ	**説明**
ROWID_RPT_CONFIG	レポートの ID。
DIMENSION_NAME_1	レポート内のデータの次元の名前。次元は例えば、ユーザー名、レビュータスクタイプ、ビジネスエンティティタイプなどです。
DIMENSION_NAME_2	オプション。レポート内のデータの次元の名前。次元は例えば、ユーザー名、レビュータスクタイプ、ビジネスエンティティタイプなどです。
TIMEPERIOD_NAME	オプション。データが適用される期間の名前。例えば、「月」と指定できます。
RPT_NAME	レポートの名前。

パラメータ	説明
METRIC_ NAME	レポートによって収集されたデータのタイプのラベル。デフォルトでは、このパラメータに指定された値は、グラフの y 軸の名前として表示されます。例えば、ユーザーがクローズしたタスクに関する情報を収集するレポートがあるとします。メトリック名に「クローズされたタスクの数」を使用できます。
RPT_DESC	レポートの説明。
RPT_TYPE	レポートがドリルダウンレポートかどうかを示します。値が null の場合、レポートはルートレポートです。値が null 以外の場合、レポートはドリルダウンレポートです。

レポートデータ

グラフにレポートデータを入力できるように、データをレポートに追加します。追加設定なしで使用できるレポートでは、レポートに関連付けられているレポート更新ジョブを使用して、レポートデータを収集および追加します。カスタムレポートを使用する場合は、カスタムレポートに手動でデータエントリを追加する必要があります。

次の表は、レポートの各データエントリに対して設定する必要があるパラメータを示しています。

パラメータ	説明
DIMENSION_NAME_1	レポート内のデータの次元の名前。次元は例えば、ユーザー名、レビュータスクタイプ、ビジネスエンティティタイプなどです。
DIMENSION_NAME_2	オプション。レポート内のデータの次元の名前。次元は例えば、ユーザー名、レビュータスクタイプ、ビジネスエンティティタイプなどです。
TIMEPERIOD_NAME	オプション。データが適用される期間の名前。例えば、「月」と指定できます。
METRIC_VALUE	データの次元を表す数値。例えば、ユーザーがクローズしたタスクに関する情報を収集するレポートがあるとします。レポートの 1 つの次元はユーザー John Smith で、別の次元はレビュータスクタイプです。メトリック値は 5 となる場合があります。これは、John Smith がレビュータスクタイプの 5 つのタスクを閉じたことを意味します。
DRILLDOWN_RPT_ID	ドリルダウンレポートの ID。値が null の場合、設定されたドリルダウンレポートはありません。

次の例は、レポート内の 3 つのデータエントリを示しています。

```
[
  {
    "DIMENSION_VALUE_1": "High",
    "DIMENSION_VALUE_2": "AVOSBeMerge",
    "TIMEPERIOD_VALUE": "null",
    "METRIC_VALUE": "3",
    "DRILLDOWN_RPT_ID": "null"
 },
  {
    "DIMENSION_VALUE_1": "High",
    "DIMENSION_VALUE_2": "AVOSBeReviewNoApprove",
    "TIMEPERIOD_VALUE": "null",
```

```
    "METRIC_VALUE": "0",
    "DRILLDOWN_RPT_ID": "null"
 },
 {
    "DIMENSION_VALUE_1": "High",
    "DIMENSION_VALUE_2": "AVOSBeUpdate",
    "TIMEPERIOD_VALUE": "null",
    "METRIC_VALUE": "0",
    "DRILLDOWN_RPT_ID": "null"
 }
    ...
]
```

ドリルダウンレポート

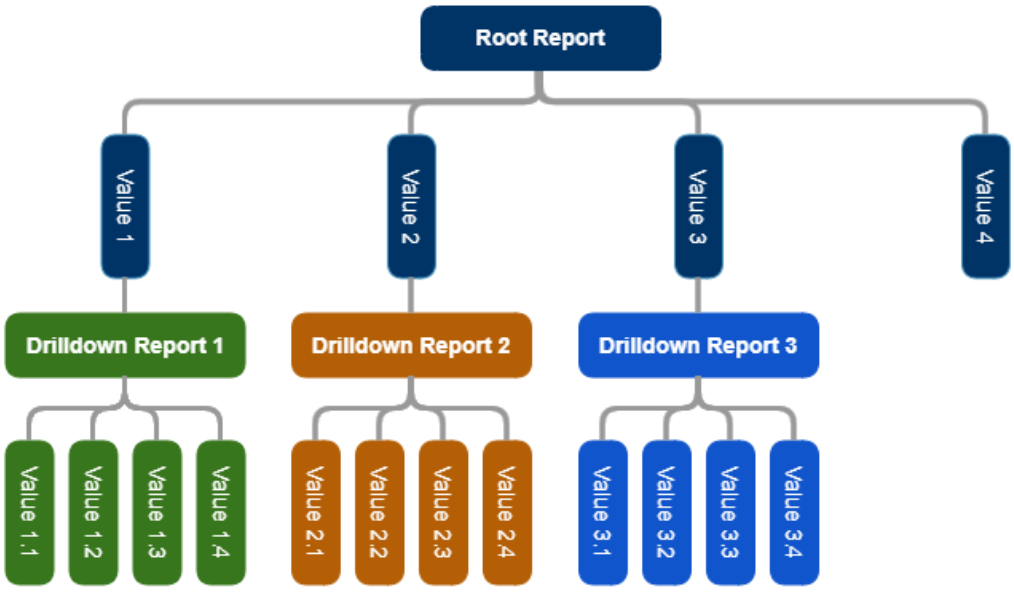
ドリルダウンレポートには、レポート内のデータエントリに関する別のデータのレイヤが含まれます。追加設定なしで使用できるレポートに関連付けられているドリルダウンレポートを使用するか、またはカスタムレポートにドリルダウンレポートを設定できます。

カスタムレポートを作成する場合、ルートレポートを登録します。ドリルダウンレポートを作成し、ルートレポートのデータエントリまたはその他のドリルダウンレポートのデータエントリに関連付けることができます。

例えば、ステータスごとのタスク数についての情報を収集するレポートがあるとします。終了したタスクの数にドリルダウンレポートを関連付けることができます。ドリルダウンレポートには、ユーザーごとの終了したタスクの数が含まれます。承認されたタスクの数にドリルダウンレポートを関連付けることもできます。ドリルダウンレポートには、ユーザーごとの承認されたタスクの数が含まれます。

Data Director では、ユーザーにステータスごとのタスクの数を示す親チャートが表示されます。さらに、終了したタスクの数を選択すると、ユーザーごとの終了したタスクの数を示すドリルダウンチャートへドリルダウンします。

次の図は、ドリルダウンレポートの構造を示しています。



追加設定なしで使用できるレポートに関連付けられているドリルダウンレポートの詳細については、「追加設定なしで使用できるレポート」 (ページ 341)を参照してください。

追加設定なしで使用できるレポート

追加設定なしで使用できるレポートはマスタデータに関連する情報を収集します。一部の追加設定なしで使用できるレポートにはドリルダウンレポートも含まれます。ドリルダウンレポートにはレポートデータに関する別のレイヤのデータが含まれます。

例えば、ステータスごとのタスクの概要レポートは、ステータスごとのタスクの数に関する情報を収集します。レポートに関連付けられたドリルダウンレポートには、ステータスごとにタスクが割り当てられたユーザーが示されます。

追加設定なしで使用できるレポートを使用するには、使用するレポートに関連付けられたレポートの更新ジョブを実行する必要があります。レポートの更新ジョブは、レポートを登録し、レポートの条件に基づきマスタデータに関する情報を収集します。そうすると、プロビジョニングツールで、レポートデータを含むチャートコンポーネントを取り込むことができます。追加設定なしで使用できるレポートに関連付けられたドリルダウンレポートがある場合、チャートをドリルダウンチャートにリンクすることができます。

次の表では、追加設定なしで使用できるレポートについて説明し、各レポートの ID をリストします:

レポート ID	**レポート**	**説明**
MDM.RPT.1	ステータス別のタスクの概要	タスクの総数を表示し、ステータス別にタスクを分類します。各タスクステータスに割り当てられたユーザーにドリルダウンできます。
MDM.RPT.2	優先度別のタスクの概要	タスクの総数を表示し、優先度別にタスクを分類します。各タスク優先度に割り当てられたユーザーにドリルダウンできます。
MDM.RPT.4	年別の追加されたビジネスエンティティ	ビジネスエンティティタイプごとに追加されたレコードの数を表示し、年別にレコードを分類します。
MDM.RPT.5	顧客のオンボーディング時間	レコードの修飾にかかった時間を表示します。
MDM.RPT.6	ソースシステム別のビジネスエンティティ	各ソースシステムからの各ビジネスエンティティタイプのレコードの総数を表示します。詳細レポートは、ビジネスエンティティタイプとソースシステムの各ペアのレコードを年別に分類します。
MDM.RPT.7	ユーザー別の割り当てられたタスク	各ユーザーに割り当てられたタスクの総数を表示します。
MDM.RPT.8	ユーザーロール別の開いているタスク	ユーザーロール別の開いているタスクの総数を表示します。
MDM.RPT.10	ユーザー別の終了したタスク	ユーザー別の終了したタスクの総数を表示します。

追加設定なしで使用できるレポートの管理

追加設定なしで使用できるレポートを使用してマスタデータに関するデータを収集するには、追加設定なしで使用できるレポートに関連付けられた更新ジョブを実行します。更新ジョブを実行すると、プロビジョニングツールで、レポートデータを含むチャートコンポーネントを取り込むことができます。

開始する前に、追加設定なしで使用できるレポートを確認し、使用するレポートを決定します。詳細については、「追加設定なしで使用できるレポート」 (ページ 341)を参照してください。

1. 追加設定なしで使用できるレポートに関連付けられた更新ジョブを実行するには、ブラウザのアドレスバーに、レポートの更新ジョブの実行 REST API から REST URL を入力します。
   例えば、次の REST URL は、ステータスごとのタスクの概要レポートに関連付けられた更新ジョブを実行します。

   ```
   http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/data/collect/MDM.RPT.1
   ```

   レポートを登録し、レポートの条件に基づくデータの収集を開始しました。次のサンプル応答では、レポートの更新ジョブのステータスが示されています。

   ```
   {
     "status": "PROCESSING"
   }
   ```

2. レポートの更新ジョブのステータスを取得するには、ブラウザのアドレスバーに、レポートの更新ジョブのステータスの取得 REST API から REST URL を入力します。
   例えば、次の REST URL は、更新ジョブのステータスを取得します。

   ```
   http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/data/collect/MDM.RPT.1/status
   ```

   次のサンプル応答では、レポートの更新ジョブのステータスが示されています。

   ```
   {
     "status": "COMPLETED_SUCCESSFULLY",
     "jobId": "SVR1.2X9N1 ",
     "lastUpdateDate": "2019-10-31T14:37:11.846-04:00",
     "startRunDate": "2019-10-31T14:37:09.120-04:00"
   }
   ```

3. レポート設定およびデータを取得するには、ブラウザのアドレスバーに、レポート設定およびデータの取得 REST API から REST URL を入力します。
   例えば、次の REST URL は、レポート設定およびデータを取得します。

   ```
   http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/data/MDM.RPT.1
   ```

   次のサンプル応答では、レポート設定およびデータが示されています。

   ```
   {
     "metadata":{
       "fieldsMetadata":[
         "DIMENSION_VALUE_1",
         "DIMENSION_VALUE_2",
         "TIMEPERIOD_VALUE",
         "METRIC_VALUE",
         "DRILLDOWN_RPT_ID"
       ],
       "ROWID_RPT_CONFIG":"MDM.RPT.1     ",
       "DIMENSION_NAME_1":"Task Status",
       "METRIC_NAME":"Number of tasks",
       "DIMENSION_NAME_2":"Task Type",
       "TIMEPERIOD_NAME":null,
       "RPT_NAME":"Task Status/Type Report",
       "RPT_DESC":"Metrics for task status/type",
       "RPT_TYPE":null
     },
     "data":[
       [
         "Overdue",
         "AVOSBeMerge",
         null,
         "36",
   ```

```
      "SVR1.4FCA2    "
    ],
    [
      "Overdue",
      "AVOSBeReviewNoApprove",
      null,
      "189",
      "SVR1.4FC8E    "
    ],
    [
      "Overdue",
      "AVOSBeUpdate",
      null,
      "1",
      "SVR1.4FCAQ    "
    ],
    [
      "Overdue",
      "AVOSBeFinalReview",
      null,
      "1",
      "SVR1.4FC9W    "
    ]
                ...
  ]
}
```

4. レポート設定およびドリルダウンレポートを取得するには、レポート設定およびドリルダウンレポートの取得 REST API を使用します。
   例えば、次の REST URL は、レポート設定およびドリルダウンレポートを取得します。

   ```
   http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/meta/MDM.RPT.1
   ```

   次のサンプル応答では、レポート設定およびドリルダウンレポートが示されています。

   ```
   {
     "ROWID_RPT_CONFIG":"MDM.RPT.1",
     "DIMENSION_NAME_1":"Task Status",
     "METRIC_NAME":"Number of tasks",
     "DIMENSION_NAME_2":"Task Type",
     "TIMEPERIOD_NAME":null,
     "RPT_NAME":"Task Status/Type Report",
     "RPT_DESC":"Metrics for task status/type",
     "RPT_TYPE":null,
     "DRILLDOWN":[
       {
         "RPT_NAME":"Task's Owner per Task Status and Task Type",
         "DETAILED_RPT_IDS":[
           "SVR1.4FCA3",
           "SVR1.4FCA4",
           "SVR1.4FCA5",
           "SVR1.4FCA6",
           "SVR1.4FCA7",
           "SVR1.4FC8F",
           "SVR1.4FC8G",
           "SVR1.4FC8H",
           "SVR1.4FC8I",
           "SVR1.4FC8J",
           "SVR1.4FCAR",
           "SVR1.4FCAS",
           "SVR1.4FCAT",
           "SVR1.4FCAU",
           "SVR1.4FCAV",
           "SVR1.4FC9X",
           "SVR1.4FC9Y",
           "SVR1.4FC9Z",
           "SVR1.4FCA0",
           "SVR1.4FCA1",
           "SVR1.4FC9L",
           "SVR1.4FC9M",
           "SVR1.4FC9N",
   ```

```
        "SVR1.4FC9O",
        "SVR1.4FC9P",
        "SVR1.4FC8L",
        "SVR1.4FC8M",
        "SVR1.4FC8N",
        "SVR1.4FC8O"
      ],
      "CONFIG_RPT_IDS":[
        "SVR1.4FCA2",
        "SVR1.4FC8E",
        "SVR1.4FCAQ",
        "SVR1.4FC9W",
        "SVR1.4FC9K",
        "SVR1.4FC8K",
        "SVR1.4FCAW",
        "SVR1.4FC9E",
        "SVR1.4FC88",
        "SVR1.4FC8W",
        "SVR1.4FC82",
        "SVR1.4FC92",
        "SVR1.4FC98",
        "SVR1.4FCAE",
        "SVR1.4FCA8",
        "SVR1.4FCB8",
        "SVR1.4FC9Q",
        "SVR1.4FCAK",
        "SVR1.4FC8Q",
        "SVR1.4FCB2"
      ]
    }
  ]
}
```

カスタムレポート

追加設定なしで使用できるレポートではビジネス要件が満たされない場合、カスタムレポートを設定して、マスタデータに関するその他の情報を収集できます。例えば、優先度別の終了したタスクの数についての情報を収集する場合があります。

カスタムレポートを使用するには、レポートを設定および登録してから、データエントリをそのレポートに追加する必要があります。ドリルダウンレポートをレポートデータと関連付けるには、ドリルダウンレポートを設定および登録します。その後、ドリルダウンレポートをカスタムレポートのデータエントリに関連付けることができます。

カスタムレポートの管理

マスタデータに関する情報を収集するのに使用するカスタムレポートを登録および設定します。そうすると、プロビジョニングツールで、レポートデータを含むチャートコンポーネントを取り込むことができます。

開始する前に、収集するデータを決定します。

1. 登録済みのレポートを確認するには、レポートのリスト REST API を使用します。
   例えば、次の要求は、登録済みのルートレポートをリストします:

   ```
   GET http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/list
   ```

2. カスタムレポートを登録するには、レポートの登録 REST API を使用します。
   例えば、次の要求は、カスタムレポートを登録します:

   ```
   POST http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/list
   {
   ```

```
  "DIMENSION_NAME_1": "Task Priority",
  "DIMENSION_NAME_2": "Task Type",
  "TIMEPERIOD_NAME": "null",
  "RPT_NAME": "Task Priority/Type Report",
  "RPT_DESC": "Metrics for task status/type",
  "METRIC_NAME": "Number of tasks"
 }
```

次のサンプル応答では、登録済みのレポートが示されています:

```
{
  "ROWID_RPT_CONFIG": "SVR1.2X9NO",
  "DIMENSION_NAME_1": "Task Priority",
  "DIMENSION_NAME_2": "Task Type",
  "TIMEPERIOD_NAME": "null",
  "RPT_NAME": "Task Priority/Type Report",
  "RPT_DESC": "Metrics for task status/type",
  "METRIC_NAME": "Number of tasks",
  "RPT_TYPE": "null"
}
```

この要求では、ROWID_RPT_CONFIG パラメータのレポート ID が返されます。

3. データエントリをレポートに追加するには、レポートデータの追加または更新 REST API を使用します。例えば、次の要求は、データエントリをレポートに追加します:

```
POST http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/data/SVR1.2X9NO
[
  {
    "DIMENSION_VALUE_1": "High",
    "DIMENSION_VALUE_2": "AVOSBeMerge",
    "TIMEPERIOD_VALUE": "null",
    "METRIC_VALUE": "3",
    "DRILLDOWN_RPT_ID": "null"
 },
  {
    "DIMENSION_VALUE_1": "High",
    "DIMENSION_VALUE_2": "AVOSBeReviewNoApprove",
    "TIMEPERIOD_VALUE": "null",
    "METRIC_VALUE": "0",
    "DRILLDOWN_RPT_ID": "null"
 },
 {
    "DIMENSION_VALUE_1": "High",
    "DIMENSION_VALUE_2": "AVOSBeUpdate",
    "TIMEPERIOD_VALUE": "null",
    "METRIC_VALUE": "0",
    "DRILLDOWN_RPT_ID": "null"
 }
    ...
]
```

4. レポート設定およびデータを取得するには、レポート設定およびデータの取得 REST API を使用します。例えば、次の要求は、レポート設定およびデータを取得します。

```
GET http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/data/SVR1.2X9NO
```

次のサンプル応答では、レポート設定およびデータが示されています:

```
{
  "metadata":{
    "fieldsMetadata":[
      "DIMENSION_VALUE_1",
      "DIMENSION_VALUE_2",
      "TIMEPERIOD_VALUE",
      "METRIC_VALUE",
      "DRILLDOWN_RPT_ID"
    ],
    "ROWID_RPT_CONFIG": "MDM.RPT.2",
    "DIMENSION_NAME_1": "Task Priority",
    "METRIC_NAME": "Number of tasks",
    "DIMENSION_NAME_2": "Task Type",
```

```
    "TIMEPERIOD_NAME": "null",
    "RPT_NAME": "Task Priority/Type Report",
    "RPT_DESC": "Metrics for task status/type",
    "RPT_TYPE": "null"
  },
  "data":[
    [
      "High",
      "AVOSBeMerge",
      "null",
      "3",
      "SVR1.48P5G"
    ],
    [
      "High",
      "AVOSBeReviewNoApprove",
      "null",
      "0",
      "null"
    ],
    [
      "High",
      "AVOSBeUpdate",
      null,
      "0",
      "null"
    ],
    [
      "High",
      "AVOSBeFinalReview",
      null,
      "0",
      "null"
    ]
    ...
  ]
}
```

5. レポートを削除するには、レポートの削除 REST API を使用します。
   例えば、次の要求は、レポートを削除します:

   ```
   DELETE http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/data/SVR1.2X9NO
   ```

ドリルダウンレポートを含むカスタムレポートの管理

使用するカスタムレポートを登録し、データエントリをレポートに追加します。次にドリルダウンレポートを登録し、データエントリをそのドリルダウンレポートに追加します。データエントリをルートレポートに追加または更新するときに、ドリルダウンレポートをデータエントリに関連付けることができます。そうすると、プロビジョニングツールで、レポートデータを含むチャートコンポーネントを取り込み、ルートチャートをドリルダウンチャートにリンクすることができます。

開始する前に、収集するデータを決定します。また、設定してカスタムレポートのデータエントリに関連付けるドリルダウンレポートも決定します。

1. 登録済みのレポートを確認するには、レポートのリスト REST API を使用します。
   例えば、次の要求は、登録済みのルートレポートおよびドリルダウンレポートをリストします。

   ```
   GET http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/list?show=all
   ```

2. カスタムレポートを登録するには、レポートの登録 REST API を使用します。
   例えば、次の要求は、カスタムレポートを登録します。

   ```
   POST http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/list
   {
     "DIMENSION_NAME_1": "Task Priority",
     "DIMENSION_NAME_2": "Task Type",
     "TIMEPERIOD_NAME": "null",
   ```

```
  "RPT_NAME": "Task Priority/Type Report",
  "RPT_DESC": "Metrics for task status/type",
  "METRIC_NAME": "Number of tasks"
 }
```

次のサンプル応答では、登録済みのレポートが示されています。

```
{
  "ROWID_RPT_CONFIG": "SVR1.2X9NO",
  "DIMENSION_NAME_1": "Task Priority",
  "DIMENSION_NAME_2": "Task Type",
  "TIMEPERIOD_NAME": "null",
  "RPT_NAME": "Task Priority/Type Report",
  "RPT_DESC": "Metrics for task status/type",
  "METRIC_NAME": "Number of tasks",
  "RPT_TYPE": "null"
}
```

この要求では、ROWID_RPT_CONFIG パラメータのレポート ID が返されます。

3. データエントリをルートレポートに追加するには、レポートデータの追加または更新 REST API を使用します。

   後で、ドリルダウンレポートをデータエントリに関連付けることができます。

   例えば、次の要求は、データエントリをレポートに追加します。

```
POST http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/data/SVR1.2X9NO
[
  {
    "DIMENSION_VALUE_1": "High",
    "DIMENSION_VALUE_2": "AVOSBeMerge",
    "TIMEPERIOD_VALUE": "null",
    "METRIC_VALUE": "3",
    "DRILLDOWN_RPT_ID": "null"
  },
  {
    "DIMENSION_VALUE_1": "High",
    "DIMENSION_VALUE_2": "AVOSBeReviewNoApprove",
    "TIMEPERIOD_VALUE": "null",
    "METRIC_VALUE": "0",
    "DRILLDOWN_RPT_ID": "null"
 },
 {
    "DIMENSION_VALUE_1": "High",
    "DIMENSION_VALUE_2": "AVOSBeUpdate",
    "TIMEPERIOD_VALUE": "null",
    "METRIC_VALUE": "0",
    "DRILLDOWN_RPT_ID": "null"
 }
 ...
]
```

4. ドリルダウンレポートを登録するには、レポートの登録 REST API を使用します。
   例えば、次の要求は、ドリルダウンレポートを登録します。

```
POST http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/list
{
  "DIMENSION_NAME_1": "Task's Owner",
  "DIMENSION_NAME_2": "null",
  "TIMEPERIOD_NAME": "null",
  "RPT_NAME": "Task's Owner per Task Priority and Task Type",
  "RPT_DESC": "Number of tasks for each users for Task Priority/Task Type",
  "METRIC_NAME": "Number of Tasks"
 }
```

次のサンプル応答では、登録済みのドリルダウンレポートが示されています。

```
{
  "ROWID_RPT_CONFIG": "SVR1.48P5G",
  "DIMENSION_NAME_1": "Task's Owner",
  "DIMENSION_NAME_2": "null",
```

```
  "TIMEPERIOD_NAME": "null",
  "RPT_NAME": "Task's Owner per Task Priority and Task Type",
  "RPT_DESC": "Number of tasks for each users for Task Priority/Task Type",
  "METRIC_NAME": "Number of Tasks",
  "RPT_TYPE": "High/AVOSBeMerge"
}
```

この要求では、ROWID_RPT_CONFIG パラメータのドリルダウンレポート ID が返されます。

5. データエントリをドリルダウンレポートに追加するには、レポートデータの追加または更新 REST API を使用します。
例えば、次の要求は、データエントリをドリルダウンレポートに追加します。

```
POST http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/data/SVR1.48P5G
[
  {
    "DIMENSION_VALUE_1": "admin",
    "DIMENSION_VALUE_2": "null",
    "TIMEPERIOD_VALUE": "null",
    "METRIC_VALUE": "0",
    "DRILLDOWN_RPT_ID": "null"
 },
  {
    "DIMENSION_VALUE_1": "srmgr2",
    "DIMENSION_VALUE_2": "null",
    "TIMEPERIOD_VALUE": "null",
    "METRIC_VALUE": "0",
    "DRILLDOWN_RPT_ID": "null"
 },
 {
    "DIMENSION_VALUE_1": "mgr2",
    "DIMENSION_VALUE_2": "null",
    "TIMEPERIOD_VALUE": "null",
    "METRIC_VALUE": "0",
    "DRILLDOWN_RPT_ID": "null"
 }
]
```

6. ドリルダウンレポートをレポート内のデータエントリに関連付けるには、レポートデータの追加または更新 REST API を使用します。DRILLDOWN_RPT_ID パラメータのドリルダウンレポート ID を指定します。
例えば、次の要求はドリルダウンレポートをデータエントリに関連付けます。

```
POST http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/data/SVR1.2X9NO
[
  {
    "DIMENSION_VALUE_1": "High",
    "DIMENSION_VALUE_2": "AVOSBeMerge",
    "TIMEPERIOD_VALUE": "null",
    "METRIC_VALUE": "5",
    "DRILLDOWN_RPT_ID": "SVR1.48P5G"
  }
]
```

7. レポート設定およびデータを取得するには、レポート設定およびデータの取得 REST API を使用します。
例えば、次の要求は、レポート設定およびデータを取得します。

```
GET http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/data/SVR1.2X9NO
```

次のサンプル応答では、レポート設定およびデータが示されています。

```
{
  "metadata":{
    "fieldsMetadata":[
      "DIMENSION_VALUE_1",
      "DIMENSION_VALUE_2",
      "TIMEPERIOD_VALUE",
      "METRIC_VALUE",
      "DRILLDOWN_RPT_ID"
    ],
    "ROWID_RPT_CONFIG": "SVR1.2X9NO",
    "DIMENSION_NAME_1": "Task Priority",
```

```
    "METRIC_NAME": "Number of tasks",
    "DIMENSION_NAME_2": "Task Type",
    "TIMEPERIOD_NAME": "null",
    "RPT_NAME": "Task Priority/Type Report",
    "RPT_DESC": "Metrics for task status/type",
    "RPT_TYPE": "null"
  },
  "data":[
    [
      "High",
      "AVOSBeMerge",
      "null",
      "3",
      "SVR1.48P5G"
    ],
    [
      "High",
      "AVOSBeReviewNoApprove",
      "null",
      "0",
      "null"
    ],
    [
      "High",
      "AVOSBeUpdate",
      "null",
      "0",
      "null"
    ],
    [
      "High",
      "AVOSBeFinalReview",
      "null",
      "0",
      "null"
    ]
    ...
  ]
}
```

8. カスタムレポートおよびドリルダウンレポートを含む登録済みのレポートを取得するには、レポートのリスト REST API を使用します。
   例えば、次の要求は、登録済みのルートレポートおよびドリルダウンレポートをリストします。

   ```
   GET http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/list?show=all
   ```

   次のサンプル応答では、登録済みのルートレポートおよびドリルダウンレポートが示されています。

   ```
   [
     {
       "ROWID_RPT_CONFIG": "SVR1.2X9N0",
       "DIMENSION_NAME_1": "Task Priority",
       "METRIC_NAME": "Number of tasks",
       "DIMENSION_NAME_2": "Task Type",
       "TIMEPERIOD_NAME": "null",
       "RPT_NAME": "Task Priority/Type Report",
       "RPT_DESC": "Metrics for task status/type",
       "RPT_TYPE": "null"
     },
     {
       "ROWID_RPT_CONFIG": "SVR1.48P5G",
       "DIMENSION_NAME_1": "Task's Owner",
       "DIMENSION_NAME_2": "null",
       "TIMEPERIOD_NAME": "null",
       "RPT_NAME": "Task's Owner per Task Priority and Task Type",
       "RPT_DESC": "Number of tasks for each users for Task Priority/Task Type",
       "METRIC_NAME": "Number of Tasks",
       "RPT_TYPE": "High/AVOSBeMerge"
     }
   ]
   ```

9. レポートを削除するには、レポートの削除 REST API を使用します。

例えば、次の要求は、レポートを削除します。

```
DELETE http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/data/SVR1.2X9NO
```

レポート API のトラブルシューティング

レポート API で問題が発生した場合は、次の情報を使用してトラブルシューティングを行います。

追加設定なしで使用できるレポートに関連付けられている更新ジョブの実行に失敗しました。

レポートサービスが ActiveVOS に接続されていない可能性があります。JNDI ルックアップ文字列をカスタマイズするために ActiveVOS EAR ファイルを編集した場合は、`cmxserver.properties` ファイルに `activevos.jndi` プロパティを追加し、カスタム JNDI ルックアップ文字列を指定する必要があります。詳細については、*『Multidomain MDM 設定ガイド』*を参照してください。

索引

R

S