Informatica

Informatica® Multidomain MDM
10.3

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
10.3
2018 年 9 月

発行日: 2019-05-28

目次

序文

*Multidomain MDM ビジネスエンティティのサービスガイド*へようこそ。このガイドでは、Informatica(R) MDM Hub のビジネスエンティティを対象にビジネスエンティティサービス呼び出しを行う方法について説明します。

このガイドは、カスタムユーザーインタフェースを設定して MDM Hub に対してビジネスエンティティサービス呼び出しを行う必要がある技術者を対象にしています。

Informatica のリソース

Informatica Network

Informatica Network は、 Informatica グローバルカスタマサポート、Informatica ナレッジベースなどの製品リソースをホストします。Informatica Network には、https://network.informatica.com からアクセスしてください。

メンバーは以下の操作を行うことができます。

- 1 つの場所からすべての Informatica のリソースにアクセスできます。
- ドキュメント、FAQ、ベストプラクティスなどの製品リソースをナレッジベースで検索できます。
- 製品の提供情報を表示できます。
- 自分のサポート事例を確認できます。
- 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

ドキュメント、ハウツー記事、ベストプラクティス、PAM などの製品リソースを Informatica Network で検索するには、Informatica ナレッジベースを使用します。

ナレッジベースには、https://kb.informatica.com から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

使用している製品の最新のドキュメントを取得するには、https://kb.informatica.com/_layouts/ProductDocumentation/Page/ProductDocumentSearch.aspx にある Informatica ナレッジベースを参照してください。

このマニュアルに関する質問、コメント、ご意見の電子メールの送付先は、Informatica マニュアルチーム（infa_documentation@informatica.com）です。

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 Network メンバである場合は、PAM（https://network.informatica.com/community/informatica-network/product-availability-matrices）にアクセスできます。

Informatica Velocity

Informatica Velocity は、Informatica プロフェッショナルサービスによって開発されたヒントおよびベストプラクティスのコレクションです。数多くのデータ管理プロジェクトの経験から開発された Informatica Velocity には、世界中の組織と協力して優れたデータ管理ソリューションの計画、開発、展開、および維持を行ってきた弊社コンサルタントの知識が集約されています。

Informatica Network メンバである場合は、Informatica Velocity リソース（http://velocity.informatica.com）にアクセスできます。

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 製品を強化したり拡張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。Informatica の開発者およびパートナーの何百ものソリューションを利用して、プロジェクトで実装にかかる時間を短縮したり、生産性を向上させたりできます。Informatica Marketplace には、https://marketplace.informatica.com からアクセスできます。

Informatica グローバルカスタマサポート

Informatica Network の電話またはオンラインサポートから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（http://www.informatica.com/us/services-and-training/support-services/global-support-centers）を参照してください。

Informatica Network メンバである場合は、オンラインサポート（http://network.informatica.com）を使用できます。

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 Java Bean 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。

スマート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 Enterprise JavaBeans 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。

Web Application Description Language（WADL）ファイルには、REST Web サービス、すべての REST URL、およびすべての REST 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 WADL 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 WADL ファイルは次の場所からダウンロードできます。

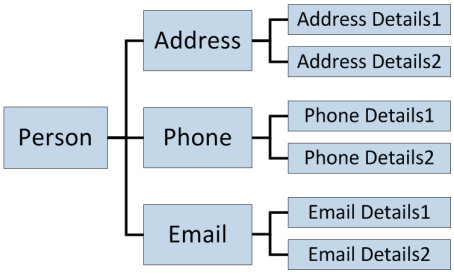
`http://<host: ホスト>:<port: ポート>/cmx/csfiles`

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。

第 2 章

Enterprise Java Bean 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。

Enterprise Java Bean ビジネスエンティティサービス呼び出しの概要

Enterprise Java Bean（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 Java Bean（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 Java Bean（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

Representational State Transfer 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

REST 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を 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 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更新または削除するのかを特定できます。

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

認証クッキーを使用して MDM Hub ユーザーを認証し、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。認証されたユーザーの資格情報に基づいてクッキーを取得できます。クッキ

ーを保存し、REST API の呼び出しに使用します。ユーザー名およびパスワードをハードコードする必要はありません。

次の POST 要求を実行し、ユーザー名とパスワードを使用してエンティティ 360 ビューにログインします。

```
POST http://<host>:<port>/e360/com.informatica.tools.mdm.web.auth/login
{
    user: 'admin',
    password: 'user password'
}
```

ログイン操作に成功すると、サーバーは set-cookie ヘッダフィールドで認証クッキーを返します。次のサンプルコードは、応答ヘッダの set-cookie を示しています。

```
Set-Cookie: auth_hash_cookie="admin===QTc1RkNGQkNCMzc1RjIyOQ==";
Version=1; Path=/
```

ハッシュを保存し、API 呼び出しの要求ヘッダで使用します。API 呼び出しではユーザー名とパスワードを指定する必要はありません。

次の例は、API 要求ヘッダで認証クッキーを使用する方法を示しています。

```
GET http://<IP of host>:8080/cmx/cs/localhost-orcl-DS_UI1/Person?action=meta
Cookie: auth_hash_cookie="admin===QTc1RkNGQkNCMzc1RjIyOQ=="
```

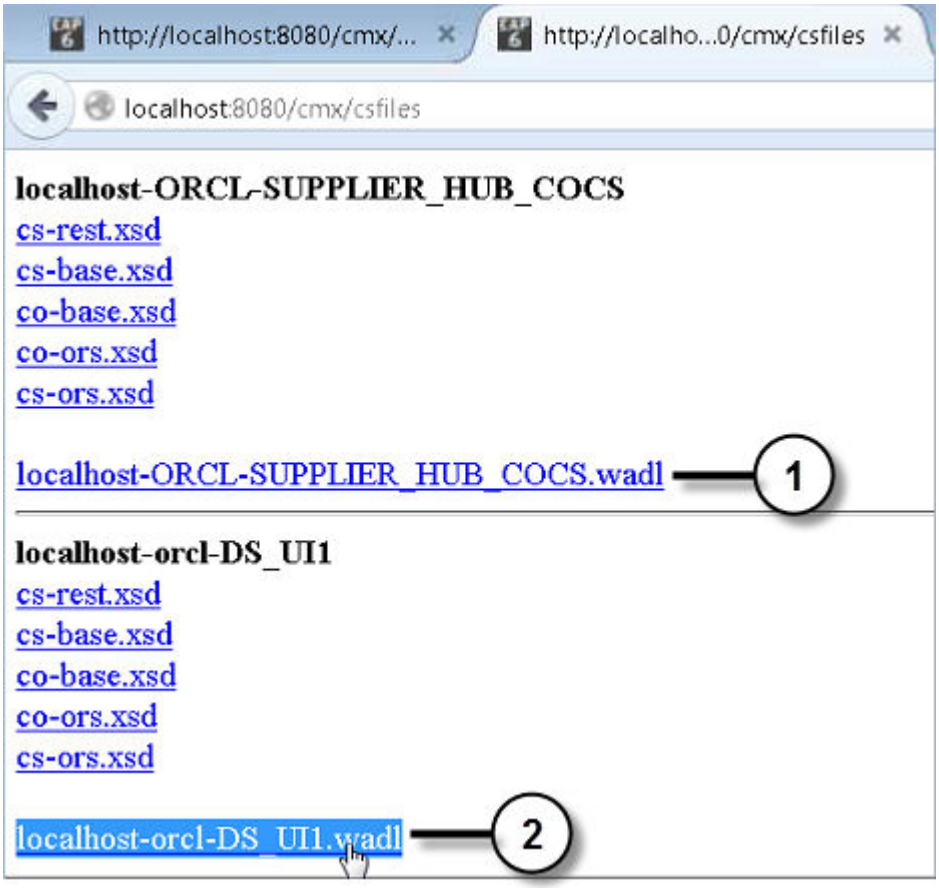
Web Application Description Language ファイル

Web Application Description Language（WADL）ファイルには、REST Web サービス、すべての REST URL、およびすべての REST 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 WADL 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 WADL 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 WADL ファイルをダウンロードできる場所を示しています。



1. SUPPLIER_HUB_COCS オペレーショナル参照ストアの WADL ファイルのダウンロード用リンク
2. DS_UI1 オペレーショナル参照ストアの WADL ファイルのダウンロード用リンク

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 API およびタスク API のコンテキストは `cmx/cs` です。

ファイル API のコンテキストは `cmx/file` 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。

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。
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。

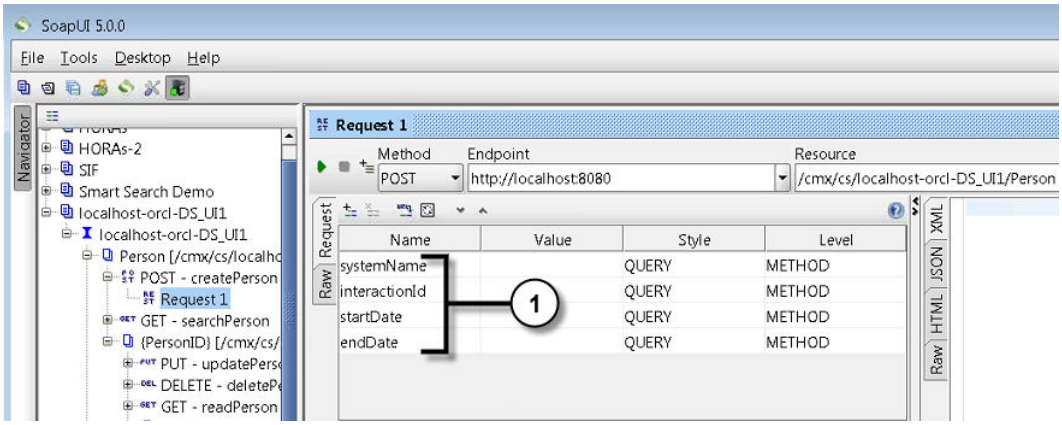
   ```
   <enforce-valid-basic-auth-credentials>
      false
   </enforce-valid-basic-auth-credentials>
   ```

入力パラメータと出力パラメータの表示

SoapUI などの有用なテストツールを使用し、REST API の入力パラメータと出力パラメータを確認できます。

WADL ファイルをダウンロードし、そのファイルを有用なテストツールにインポートして、REST プロジェクトを作成します。

次の図は、SoapUI に表示された、createPerson REST API の入力パラメータを示しています。



1. createPerson REST API の入力パラメータ

注: WebSphere 環境で生成される WADL ファイルは、SoapUI に正しくインポートされない場合があります。SoapUI に入力パラメータが表示されない場合は、WADL ファイルを編集して各 `param` 要素から `xmlns` 属性を削除し、その後で WADL ファイルをもう一度インポートしてください。

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>
```

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

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
      }
    },
    ...
  ]
}
```

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。

パラメータ	説明
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。
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。

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 29)

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"
        }
    ],
    "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。
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。

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。

関連項目：

- 「UTC での日付と時刻の形式」 (ページ 29)

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。

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_SVR1.29OT8,TEMP_SVR1.29OT9
Content-Type: application/json

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

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。

注: 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 バージョンも使用できます。

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 PUT 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。
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 29)

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
          ]
        }
      }
    ]
  }
}
```

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。

```
PUT http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/233?systemName=Admin&taskComment=Read my
comment&taskAttachments=TEMP_SVR1.29OT8,TEMP_SVR1.29OT9
```

```
{
    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。

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`

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。
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。
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=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、検索結果が誕生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。

パラメータ	説明
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。

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 29)

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>
```

注: Solr 検索エンジンを使用する場合は、プロセスサーバーが再起動するたびに提案元インデックスを再構築する必要があります。 プロセスサーバーが再起動するたびにインデックスが再構築されるようにするには、buildIndex パラメータを true に設定します。

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

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"

}
```

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。
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。

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?recordsToReturn=100&owner=sergey&status=OPEN
```

関連項目：

- 「UTC での日付と時刻の形式」 (ページ 29)

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。

パラメータ	説明
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 29)

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。
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 29)

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。
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 29)

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。
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 29)

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": {}
}
```

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_SVR1.0JU1 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_SVR1.0JU1
```

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。

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 29)

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 29)

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、マージされ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、要求本文で名を指定できます。マージ候補を削除するか、別のマージ候補を指定することで、子レコードを統合する方法をオーバーライド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。マージに追加する、同類の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対象子レコードで、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	子レコードをマージする方法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

［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
```

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、マージされ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、要求本文で名を指定できます。マージ候補を削除するか、別のマージ候補を指定することで、子レコードを統合する方法をオーバーライド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。マージに追加する、同類の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対象子レコードで、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	子レコードをマージする方法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       "
    }
}
```

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"
    }
  ]
}
```

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。

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

パラメータ	説明
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。
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 29)

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。
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"
    }
  }
}
```

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。
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 29)

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 29)

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

Simple Object Access Protocol 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の Simple Object Access Protocol 呼び出し

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

認証クッキーを使用して MDM Hub ユーザーを認証し、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。認証されたユーザーの資格情報に基づいてクッキーを取得できます。クッキーを保存し、SOAP API の呼び出しに使用します。ユーザー名およびパスワードをハードコードする必要はありません。

次の POST 要求を実行し、ユーザー名とパスワードを使用してエンティティ 360 ビューにログインします。

```
POST http://<host>:<port>/e360/com.informatica.tools.mdm.web.auth/login
{
    user: 'admin',
    password: 'user password'
}
```

ログイン操作に成功すると、サーバーは set-cookie ヘッダフィールドで認証クッキーを返します。次のサンプルコードは、応答ヘッダの set-cookie を示しています。

```
Set-Cookie: auth_hash_cookie="admin===QTc1RkNGQkNCMzc1RjIyOQ==";
Version=1; Path=/
```

ハッシュを保存し、API 呼び出しの要求ヘッダで使用します。API 呼び出しではユーザー名とパスワードを指定する必要はありません。

次の例は、API 要求ヘッダで認証クッキーを使用する方法を示しています。

```
GET http://<IP of host>/cmx/cs/localhost-orcl-DS_UI1
Cookie: auth_hash_cookie="admin===QTc1RkNGQkNCMzc1RjIyOQ=="
```

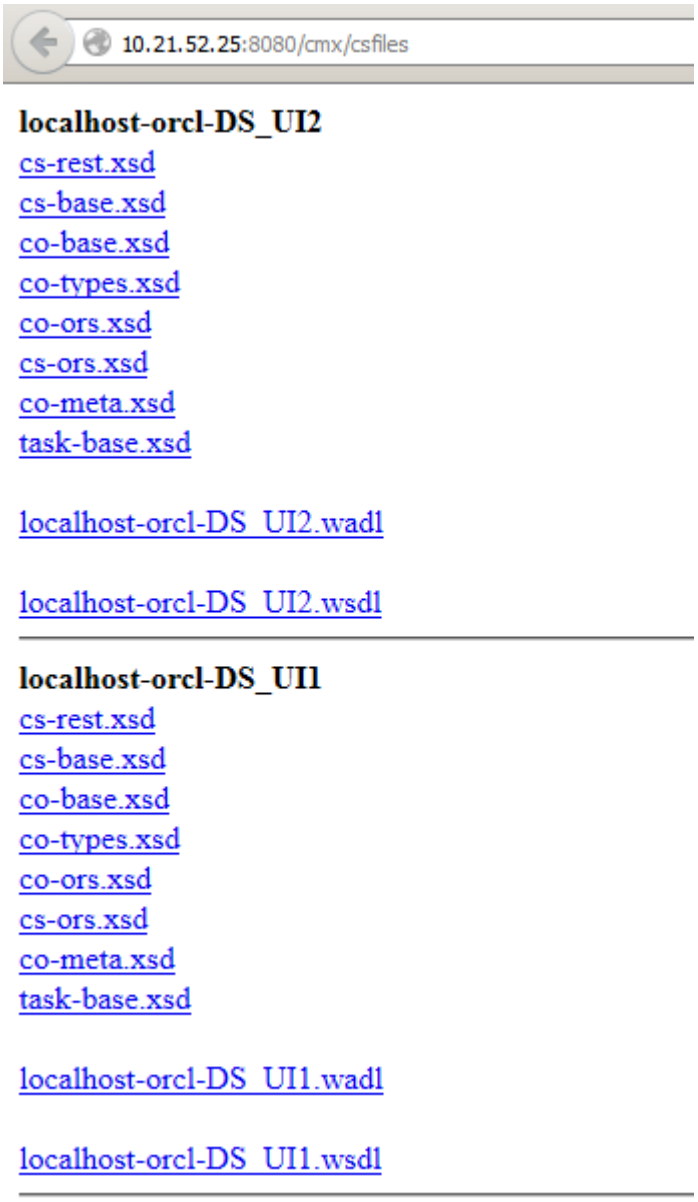
Web サービス記述言語ファイル

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。



DS_UI1 または DS_UI2 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。

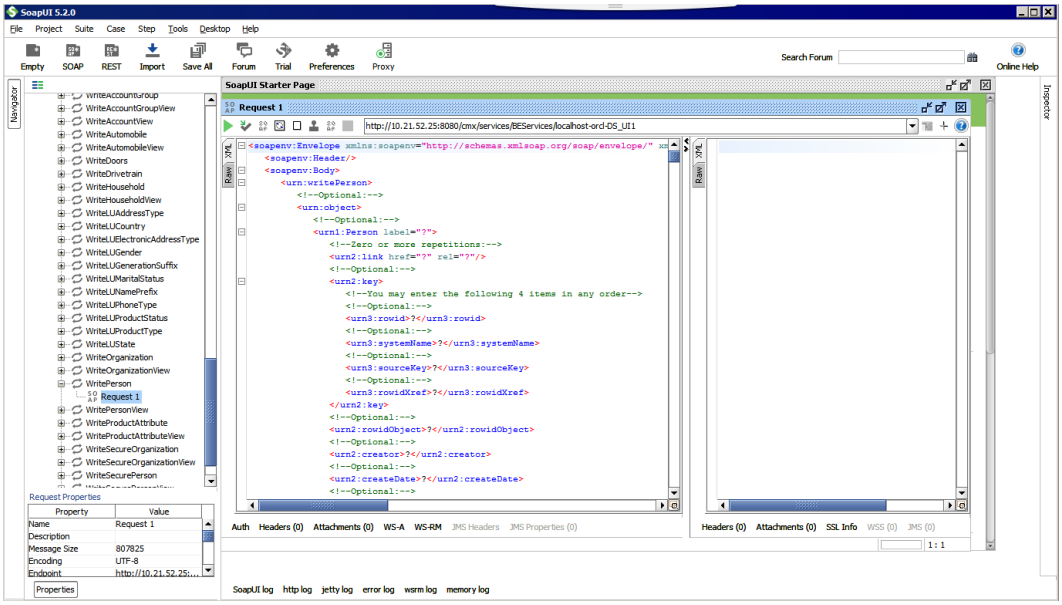
```
HTTP/1.0 200 OK
Content-Type: text/xml; charset=utf-8

<?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

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 応答では、割り当て可能なユーザーが示されています。

```
<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>
```

第 5 章

相互参照レコードおよび BVT 計算用のサービス

この章では、以下の項目について説明します。

相互参照レコードおよび BVT 計算用のサービスの概要

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 6 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 123)
- 「DaaS 更新」 (ページ 126)

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 7 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。

サポートされるイベント

ビジネスエンティティサービスはサービス手順で構成されます。任意の手順にカスタムロジックや検証を適用できます。

次のイベントについて、外部呼び出しを行うことができます。

- WriteCO.BeforeEverything
- WriteCO.BeforeValidate
- WriteCO.AfterValidate
- WriteCO.AfterEverything
- WriteView.BeforeEverything
- WriteView.BeforeValidate
- WriteView.AfterValidate
- WriteView.AfterEverything
- MergeCO.BeforeEverything
- MergeCO.AfterEverything
- PreviewMergeCO.BeforeEverything
- PreviewMergeCO.AfterEverything
- ReadCO.BeforeEverything
- ReadCO.AfterEverything
- ReadView.BeforeEverything
- ReadView.AfterEverythingEvents

外部呼び出しの構成方法

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、外部サービスへの呼び出しを構成できます。

次の手順を実行して、外部呼び出しを構成します。

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、次のタスクを実行します。
   a. 外部サービスの WSDL ファイルをアップロードします。
   b. Web サービスを SOAP サービスとして登録します。
   c. 外部呼び出しを構成します。

WSDL ファイルのアップロード、SOAP サービスの登録、および外部呼び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。

EAR ファイルの作成および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。

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": "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": 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": "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. 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": "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": []
```

```
        }
      }
    }
```

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。

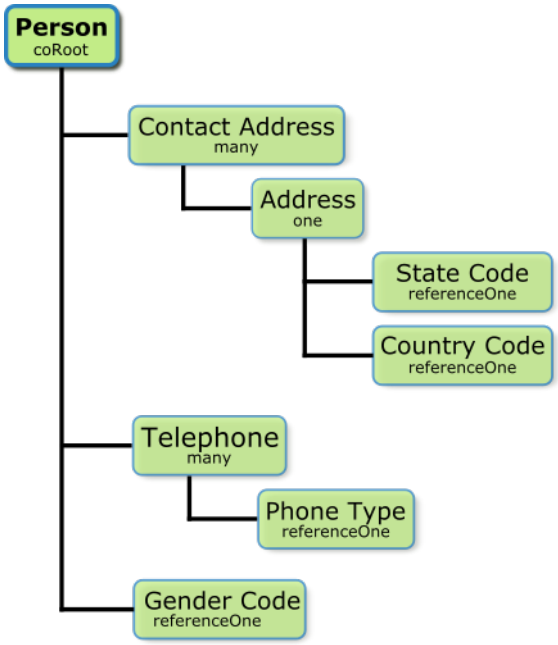
従業員の詳細を追加するには、次の API を使用します。

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.OJU3 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.OJU3
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.OJU3 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.OJU3/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.OJU3 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.OJU3
   ```

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
   ```

索引