Informatica® Multidomain MDM
10.4 HotFix 3

Business Entity Services Guide

Informatica Multidomain MDM Business Entity Services Guide
10.4 HotFix 3
October 2021

Publication Date: 2021-10-28

Table of Contents

Preface

See the Informatica® *Multidomain MDM Business Entity Services Guide* to learn about the business entity services available as web services. Use the guide to learn how to use Enterprise JavaBeans (EJB), Representational State Transfer (REST), Simple Object Access Protocol (SOAP), and external web service calls to perform operations on business entity data. You can also learn to configure custom user interfaces to make business entity service calls.

Informatica Resources

Informatica provides you with a range of product resources through the Informatica Network and other online portals. Use the resources to get the most from your Informatica products and solutions and to learn from other Informatica users and subject matter experts.

Informatica Network

The Informatica Network is the gateway to many resources, including the Informatica Knowledge Base and Informatica Global Customer Support. To enter the Informatica Network, visit https://network.informatica.com.

As an Informatica Network member, you have the following options:

- Search the Knowledge Base for product resources.
- View product availability information.
- Create and review your support cases.
- Find your local Informatica User Group Network and collaborate with your peers.

Informatica Knowledge Base

Use the Informatica Knowledge Base to find product resources such as how-to articles, best practices, video tutorials, and answers to frequently asked questions.

To search the Knowledge Base, visit https://search.informatica.com. If you have questions, comments, or ideas about the Knowledge Base, contact the Informatica Knowledge Base team at KB_Feedback@informatica.com.

Informatica Documentation

Use the Informatica Documentation Portal to explore an extensive library of documentation for current and recent product releases. To explore the Documentation Portal, visit https://docs.informatica.com.

If you have questions, comments, or ideas about the product documentation, contact the Informatica Documentation team at infa_documentation@informatica.com.

Informatica Product Availability Matrices

Product Availability Matrices (PAMs) indicate the versions of the operating systems, databases, and types of data sources and targets that a product release supports. You can browse the Informatica PAMs at https://network.informatica.com/community/informatica-network/product-availability-matrices.

Informatica Velocity

Informatica Velocity is a collection of tips and best practices developed by Informatica Professional Services and based on real-world experiences from hundreds of data management projects. Informatica Velocity represents the collective knowledge of Informatica consultants who work with organizations around the world to plan, develop, deploy, and maintain successful data management solutions.

You can find Informatica Velocity resources at http://velocity.informatica.com. If you have questions, comments, or ideas about Informatica Velocity, contact Informatica Professional Services at ips@informatica.com.

Informatica Marketplace

The Informatica Marketplace is a forum where you can find solutions that extend and enhance your Informatica implementations. Leverage any of the hundreds of solutions from Informatica developers and partners on the Marketplace to improve your productivity and speed up time to implementation on your projects. You can find the Informatica Marketplace at https://marketplace.informatica.com.

Informatica Global Customer Support

You can contact a Global Support Center by telephone or through the Informatica Network.

To find your local Informatica Global Customer Support telephone number, visit the Informatica website at the following link:
https://www.informatica.com/services-and-training/customer-success-services/contact-us.html.

To find online support resources on the Informatica Network, visit https://network.informatica.com and select the eSupport option.

CHAPTER 1

Introduction to Business Entity Services

This chapter includes the following topics:

Business Entity Services Overview

A business entity service is a set of operations that run MDM Hub code to create, update, delete, and search for base object records in a business entity. You can develop a custom user interface that can run Java code or JavaScript code to make business entity service calls.

For example, you can create a business entity service to augment a supplier record with Dun and Bradstreet data. Configure the business entity service to take the supplier record as an input, retrieve some information from Dun and Bradstreet, update the record, and then output the updated supplier record.

Base objects in a business entity have the following business entity services:

Read

Each business entity has a business entity service to perform a read operation.

Write

Each business entity has a business entity service to perform a write operation.

Search

Any business entity that has searchable fields has a business entity service to perform a search operation.

For example, a Person business entity has searchable fields. The MDM Hub generates a ReadPerson, WritePerson, and SearchPerson business entity service. The read, write, and search business entity service steps allow you to read, create, update, delete, and search for records in a business entity.

Business Entity Services

A business entity service performs an operation. You can use the ReadBE, WriteBE, and SearchBE business entity services.

A business entity service has service steps. An incoming request passes through each service step. The output of one step is an input for the next step. The output of a step can pass information to the input of the following step. All business entity service steps run as one Enterprise JavaBeans (EJB) call in a single transaction. The MDM Hub handles exceptions.

Note: Before you use the business entity services, validate the Operational Reference Store.

ReadBE Business Entity Service

The ReadBE business entity service reads data from a base object record in a business entity.

You can specify pagination parameters with the ReadBE step to set the number of records to return and the page of results to view.

The results of the ReadBE service do not include soft-deleted records.

If you do not pass the EffectiveDate parameter in the business entity service request, the MDM Hub assumes a NULL effective date and the business entity service reads data from the base object. If you pass the EffectiveDate parameter, the MDM Hub calculates the best version of the truth from cross-reference records and the read business entity service returns the up-to-date best version of the truth.

WriteBE Business Entity Service

The WriteBE business entity service can update the data in a business entity element, create a child business entity element, or delete child business entity elements.

Note: The WriteBE business entity service uses existing trust settings to calculate trust on base objects. You cannot perform a trust override with this service.

Optional Parameters

The following table describes optional parameters that you can use with the WriteBE business entity service:

Parameter	Description
recordState	Set the record state to ACTIVE, PENDING, or DELETED. **Note:** When you set `recordState=ACTIVE` and you run the service on a soft-deleted record, the service restores the record to the active state.
EffectivePeriod	Specify an effective period. If you do not pass the `EffectivePeriod` parameter, the MDM Hub assumes an unbounded period. The MDM Hub does not check that there is alignment for effective periods between the root objects and child objects. When you create or update records, ensure the effective periods of the parent record and child records align.

SearchBE Business Entity Service

Use the SearchBE business entity service to search for a root record in a business entity.

For information on configuring business entities for search, see the *Multidomain MDM Configuration Guide*.

Business Entity Service Endpoints

You can access business entity services through an Enterprise JavaBeans (EJB) endpoint, a Representational State Transfer (REST) endpoint, or a Simple Object Access Protocol (SOAP) endpoint.

REST endpoints are built upon an EJB standpoint. The REST business entity service configuration defines how the REST URLs are mapped to the EJB business entity service calls.

EJB Endpoint for Business Entity Services

The Enterprise JavaBeans (EJB) endpoint is the underlying endpoint for all types of business entity service calls. All other endpoints are mapped to the EJB endpoint

Business entity services are exposed as a stateless EJB. A stateless EJB container can pool instances, allocate instances, and apply load balancing strategies to distribute the load across different servers within the domain.

An EJB endpoint accepts a user name and password for authentication.

REST Endpoint for Business Entity Services

Representational State Transfer Endpoint (REST) calls make business entity services available as web services.

The MDM Hub uses Swagger to list all the REST web services with URLs and parameters.

You can access Swagger with the following URL:

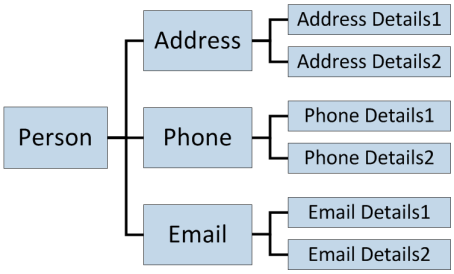
```
http://<host>:<port>/cmx/swagger-ui.html
```

REST and EJB Business Entity Service Calls

When you make a business entity service call, you might specify certain child branches instead of requesting the whole business entity.

For example, you want to perform a read operation on a business entity that has a Person root node and multiple child branches. The Person base object has Address, Phone, and Email child base objects. Each child base object has two grandchild base objects.

The following image shows the structure of a business entity with multiple branches:



You can read from multiple child branches at various depths in a single request. For example, you can read Person, Phone, Phone Details1, Phone Details 2, Email, and Email Details 2 in a single request.

The following URL sample shows how to make a REST read request to get the Person record with row ID 1242, in addition to the Address Details 1 and Email child records:

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

SOAP Endpoint for Business Entity Services

Simple Object Access Protocol (SOAP) endpoint calls make business entity services available as web services.

Web Services Description Language (WSDL) files contain the XML descriptions of the web services, formats of the SOAP requests and responses, and all parameters. The MDM Hub generates a WSDL file for each Operational Reference Store.

Identifying a Root Record

You can use one of the following approaches to identify a root record:

- rowid. The value in the ROWID_OBJECT column of the record.
- systemName and sourceKey. The systemName is the name of the system to which the record belongs. The sourceKey is the value in the PKEY_SRC_OBJECT column of the record.
- Global Business Identifier (GBID) of an object. A GBID can be a compound value, in which case you must pass all the values.

  Note: The GBID approach works only with the ReadBE service.

The following sample code uses the systemName and sourceKey to identify a record:

```
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);
```

Security and Data Filters

When base objects and resources have user role privileges, the business entity inherits those privileges. To access business entity records, the user role must have the appropriate privileges to the root base object for the business entity as well as other resources.

The business entity services also inherit any data filters that you set on business entity fields.

For more information about security and data filters, see the *Multidomain MDM Provisioning Tool Guide*.

Certificate Based Authentication

The MDM Hub uses a certificate-based authentication mechanism for securing the authentication between the MDM Hub components and trusted applications. The authentication mechanism is also supported for the Business Entity Services APIs.

External client applications can make business entity service requests to the MDM Hub. You must register the external client application users as trusted application users in the MDM Hub. You must also register the public certificate for the users associated with the external client application.

After registration, the external client users can send encrypted authentication requests to the MDM Hub using the private key. The MDM Hub decrypts the authentication request using the public key and responds with requested data in plain text.

User authentication requests for Business Entity Services payload are encrypted using the private key and are decrypted using the public key.

Note: To register the public key and the certificate with the MDM Hub, contact Informatica Global Customer Support.

For more information about how to configure trusted applications and certificate based authentication, see the Certificate Based Authentication chapter in the Informatica Multidomain MDM 10.4 Security Guide.

CHAPTER 2

EJB Business Entity Service Calls

This chapter includes the following topics:

EJB Business Entity Service Calls Overview

You can make Enterprise JavaBeans (EJB) business entity service calls to create, update, delete, and search for base object records in a business entity. You can create Java code to run EJB business entity service calls.

You can create Java code based on standard Service Data Objects (SDO) classes or you can create Java code based on java classes that the MDM Hub generates based on the business entity and business entity services configuration.

Java Code Example with Standard SDO Classes

The example shows Java code to run Enterprise JavaBeans (EJB) calls based on standard Service Data Objects (SDO) classes.

The example is in the following file in the resource kit: `C:\<MDM Hub installation directory>\hub\resourcekit\samples\COS\source\java\com\informatica\mdm\sample\cs\DynamicSDO.java`

The following Java code is based on standard SDO classes and runs EJB business entity service calls to create a Person base object record, add multiple child records, delete one child record, and then delete the Person record and all child records:

```
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();
    }

}
```

Java Code Example with Generated SDO Classes

The example shows Java code to run Enterprise JavaBeans (EJB) calls based on Java classes that the MDM Hub generates based on the business entity and business entity services configuration.

The example is in the following file in the resource kit: `C:\<MDM Hub installation directory>\hub\resourcekit\samples\COS\source\java\com\informatica\mdm\sample\cs\GeneratedSDO.java`

The following Java code is based on generated classes and runs EJB business entity service calls to create a person base object record, add multiple child records, delete one child record, and then delete the Person record and all child records:

```
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();
    }
}
```

CHAPTER 3

REST Business Entity Service Calls

This chapter includes the following topics:

REST APIs for Business Entity Services Overview

Representational State Transfer (REST) endpoint calls make all business entity services available as web services.

You can make REST calls to create, update, delete, and search for base object records and related child records in a business entity. You can perform operations, such as merge, unmerge, and match records. You can make REST calls to create, update, and search for tasks and perform tasks. You can also make REST calls to create, update, and delete files, such as attachments for tasks or records. Before you use the REST APIS to call the business entity services, validate the Operational Reference Store.

A REST business entity service call is a web service request in the form of a Uniform Resource Locator (URL). The MDM Hub assigns a unique URL for each base object in a business entity. You can use the unique URL to identify which base object to create, update or delete.

You can specify either the business entity or business entity view in the URL of most REST API calls. For example, the Create Record REST API call can use the following formats:

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

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

Some REST APIs do not support business entity views. For example, REST APIs related to merge tasks, such as Preview Merge, Pending Merge, or Promote Merge, do not support business entity views.

Note: Before you use the REST APIs to call the business entity services, validate the Operational Reference Store.

Supported REST Methods

The REST APIs for business entity services use the standard HTTP methods to perform operations on the resources, such as records, tasks, and files.

The REST APIs for business entity services support the following HTTP request methods:

Method	Description
GET	Retrieves information about a record, a task, or a file.
POST	Creates a task, a root record, a child record, or a file. **Note:** The Operational Reference Store (ORS) name in a POST request is case sensitive. If the case of the ORS name does not match the name in the MDM Hub, an error occurs.
PUT	Updates a root record, a child record, a task, or a file.
PATCH	Updates a task partially.
DELETE	Deletes a root record, a child record, or a file.

Authentication Method

The REST endpoints for business entity services use the basic HTTP authentication method to authenticate users. When you first connect to a business entity service with your browser, you must provide your MDM Hub user name and password. After successful authentication, you can use the business entity services REST APIs to perform operations.

The browser caches the user credentials and uses them with every subsequent request to a business entity service.

Authentication Cookies for Login from Third-Party Applications

Use authentication cookies to authenticate MDM Hub users and call business entity services from third-party applications, such as Postman. With the use of authentication cookies, you need not hard-code the user name and password.

The process for using authentication cookies includes the following steps:

1. Use the credentials of an authenticated user to get an authentication cookie.
2. Save the authentication cookie.
3. Use the authentication cookie to call REST APIs.

All MDM sessions use an Informatica CSRF Token (ICT). An ICT appears in the HTTP header.

When a user requests a session ID, the response returns an ICT as part of the HTTP header. For external applications, there is a URL for authentication that the external applications can use. This URL returns an authentication cookie and an ICT that is in the header. The authentication cookie is an MDM session ID.

For any subsequent business entity services calls, the call must include the authentication cookie and the ICT. Subsequent requests can use the ICT and MDM session ID until the session expires.

When you send a request to get the authentication cookie and ICT, the `userinfo` section is optional.

To manage sessions, use REST calls.

Get the Authentication Cookie and Informatica CSRF Token

The following example shows how to get the authentication cookie and the Informatica CSRF Token (ICT):

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

Authorization:   No Auth

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

The following example shows how the ICT appears in the header:

```
ICT:  0ffa3b95c67e111b9c14c140a70273a15db266993d0c763d555135de4c4d6110
```

The following example shows how the authentication cookie appears in the response:

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

Create a Person Record

The following example shows how to create a Person record with the authentication cookie and Informatica CSRF Token (ICT):

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

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

Headers:
Cookie:  mdmsessionid: vppOTOhJLKMjYJf7Diil
ICT:  0ffa3b95c67e111b9c14c140a70273a15db266993d0c763d555135de4c4d6110

Body:

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

The ICT in this example is from a preceding sample.

The following sample shows the error that appears in the response when you do not pass the authentication cookie and ICT in the request:

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

Set the Session Timeout

When you send a request to get the authentication cookie and Informatica CSRF Token (ICT), the session `userinfo` section is optional. To set the session timeout, include a `userinfo` section in the request and specify the session timeout parameter. The value of the parameter is in seconds.

The following sample code shows the user information section and the session timeout parameter:

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

Get User Information in the Session Login Request

You can get the details of the user who requested the authentication cookie and Informatica CSRF Token (ICT).

The following example request shows how to get user information from the session login request:

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

Headers:

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

The following sample response output displays role and session timeout information:

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

Refresh a Session

You can extend the validity of a session by 30 minutes with every refresh request. You can specify the session timeout interval in the session login request.

The following example request shows how to extend or refresh a session:

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

Headers:

Cookie: mdmsessionid=zjl4PDN8MEqiQWqAXwxX
```

Validate a Session

You can validate whether a session is active and valid.

The following example request shows how to validate a session:

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

Headers:

Cookie: mdmsessionid=zjl4PDN8MEqiQWqAXwxX
```

The HTTP status of OK indicates that the session is active and valid.

Log Out of a Session

After you log out of a session, the authentication cookie and Informatica CSRF Token (ICT) related to the session expire. If you use them in a request, an error appears in the response.

The following example request shows how to log out of a session:

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

Headers:

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

The HTTP status of OK indicates that the log out of the session was successful.

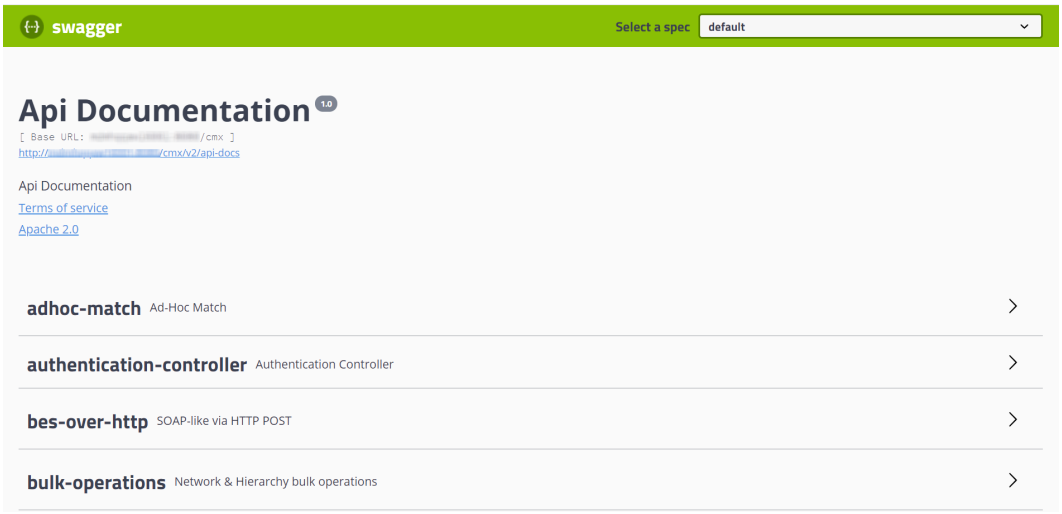
Swagger

The MDM Hub uses Swagger to list all the REST web services with URLs and parameters.

You can access Swagger with the following URL:

```
http:://<host>:<port>/cmx/swagger-ui.html
```

The following image displays the user interface of Swagger:



REST Uniform Resource Locator

You use a REST URL to make REST calls for business entity services.

The REST URL has the following syntax:

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

The URL has the following fields:

host

The host that is running the database.

port

Port number that the database listener uses.

context

The context for the business entity, search, query, match, task, and hierarchy APIs is `cmx/cs`.

The context for the match columns API is `cmx`.

The context for file APIs is `cmx/file`.

The context for task APIs is `cmx/task`.

The context for bulk task administration APIs is `cmx/task/operations`.

The context for bulk relationship APIs is `cmx/bulk`.

The context for chart report APIs is `cmx/report`.

Note: In a Hosted MDM environment, include the tenant name in the context. For example, the context can be `<tenant name>/cmx/cs` or `<tenant name>/cmx/file`.

database ID

ID of the ORS as registered in the Databases tool in the Hub Console.

path

The object that you want to use the API on, such as records, tasks, or files.

If the URL is for a root record, the path is the root object name followed by a unique identifier.

An example of a path for a Person root record is `Person/798243.json`.

If the URL is for a record that is a direct child of the root object, the path also includes the child record name and a unique identifier.

An example of a path for a billing address record that is a child of a Person root record is:

`Person/798243/BillAddresses/121522.json`.

If the URL is for a child record that is at a depth of two or greater, the path also includes the depth.

The following URL is an example of a REST URL for a child record with a depth of 2:

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

Note: Parameters are case sensitive. Ensure the case of the parameter names in the REST URL matches the case of the parameter names in the REST configuration.

Header and Body Configuration

A REST operation combines an HTTP method with the full URL to the resource. For a complete request, combine the REST operation with the appropriate HTTP headers and any required data. A REST request has a header and a body component. You can use the JSON or the XML format to define a request.

Request Header

Use a request header to define the operating parameters or the metadata of the REST operation. The header consists of a series of field-value pairs. The API request line contains the method and the URL. Specify the header fields after the request line.

To construct the REST API request header, add the header fields after the `<METHOD> <<host>:<port>/<context>/<database ID>/<Path>` request line, as shown in the following example:

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

The following table describes some of the commonly used request header fields:

Request component	Description
Content-Type	Media type of the data in the request. When you include a body in the REST request, you must specify the media type of the body in the Content-Type header field. Include the Content-Type header field in PUT and POST requests.
Accept	Media type of the data in the response. To specify the request format, use `application/<json/xml>` in the header or add `.json` or `.xml` to the URL. Default is XML.

Request Body

Use the REST API request body to send data in the request. A request body is used with a method that can have a body attached to it, such as the POST and the PUT methods. The body of data is written after the header lines. If the request message has a body, use the Content-Type header field to specify the format of the body in the request header.

The XML Schema Definition (XSD) files describe what elements and attributes you can use. The content of the request body depends on the element types that you define in the XSD files.

The XSD files are in the following location:

```
http:://<host>:<port>/cmx/csfiles
```

XML Format

When you use the XML request format, define the request object as an enclosing set of tags.

Use the following XML format to define a request object:

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

The following example shows the XML representation of a request object:

```
<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 Format

When you use the JSON request format, define the request object with the type attribute.

Use the following JSON format to specify a request object:

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

The following example shows the JSON representation of a request object:

```
{
     "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: ''
         }
       }
     }]
}
```

Standard Query Parameters

The business entity services REST APIs use standard query parameters to filter, paginate, and expand the results.

Use a question mark (?) to separate the query parameters from the other parameters. Query parameters are key-value pairs separated by the equal sign. Use an ampersand (&) to separate a sequence of query parameters.

The following REST request URL shows how to use query parameters:

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

Use the following standard query parameters:

Parameter	Description
recordsToReturn	Specifies the number of rows to return. Default is 10.
firstRecord	Specifies the first row in the result. Default is 1. Used in subsequent calls to read more pages.
searchToken	Specifies the search token returned with previous request. You can use the search token to fetch subsequent pages of search results. For example, the following query lists the first page: `/Person/123/Phone` The following query returns the second page: `/Person/123/Phone?searchToken=SVR1.AZAM5&firstRecord=10`

Parameter	Description
returnTotal	If set to true, returns the number of records in the result. Default is false.
depth	Specifies the number of child levels to include in the result.

Formats for Dates and Time in UTC

In the request and response, all dates and times are specified in UTC (Coordinated Universal Time), with or without an offset for a specific time zone.

When you specify a date and time in a request body, use one of the formats defined in Date and Time Formats (NOTE-datetime) for ISO specification 8601.

The following guidelines are taken from the NOTE-datetime document:

Type	Syntax	Example
Date: Year	YYYY	1997
Date: Year and month	YYYY-MM	1997-07
Date: Year, month, and day	YYYY-MM-DD	1997-07-16
Date plus hours and minutes	YYYY-MM-DDThh:mmTZD	1997-07-16T19:20+01:00
Date plus hours, minutes, and seconds	YYYY-MM-DDThh:mm:ssTZD	1997-07-16T19:20:30+01:00
Date plus hours, minutes, seconds, and fractional seconds	YYYY-MM-DDThh:mm:ss.sTZD	1997-07-16T19:20:30.45+01:00

where:

- YYYY = a four-digit year
- MM = a two-digit month, from 01 to 12
- DD = a two-digit day of the month, from 01 to 31
- T = a literal value that follows the date and introduces the time
- hh = two digits for the hour, from 00 to 23
- mm = two digits for the minutes, from 00 to 59
- ss = two digits for the seconds, from 00 to 59
- s = one or more digits representing a decimal fraction of a second
- TZD = time zone designator (Z or +hh:mm or -hh:mm)
  - Z for UTC time
  - +hh:mm for a local time zone that is ahead of UTC
  - -hh:mm for a local time zone that is behind UTC

Configuring WebLogic to Run Business Entity Service REST Calls

Because business entity services REST calls use basic HTTP authentication, you must disable the WebLogic Server authentication for the REST calls. To configure WebLogic to run business entity service REST calls, edit the WebLogic `config.xml` file.

1. Navigate to the following WebLogic directory:

   On UNIX.

   ```
   <WebLogic installation directory>/user_projects/domains/base_domain/config
   ```

   On Windows.

   ```
   <WebLogic installation directory>\user_projects\domains\base_domain\config
   ```

2. Open the following file in a text editor:

   ```
   config.xml
   ```

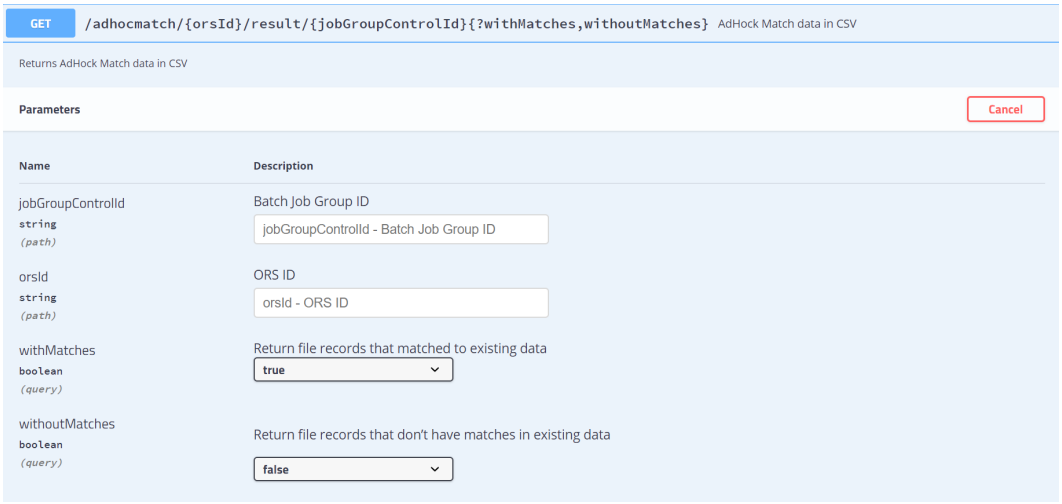
3. Before the closing </security-configuration> tag, add the following XML code:

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

Viewing Input and Output Parameters

You can use Swagger to view the input and output parameters of the REST APIs.

The following image shows the input parameters in Swagger for the adhoc-match REST API:



JavaScript Template

The following code sample shows a basic template that you can modify to create JavaScript code for REST business entity service calls. You need the jQuery java script library.

```
(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 Example

The resource kit has sample Java source code that shows how to make REST business entity service calls.

The sample code is in the following file:

`<MDM Hub installation directory>\hub\resourcekit\samples\COS\source\resources\webapp\rest-api.html`

The following code shows REST API calls to create a person root record, add multiple child records, delete one child record, and then delete the person record and all child records:

```
<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>
```

AJAX Call with HTTP Basic Authentication Example

A call to business entity services requires HTTP basic authentication.

The following code shows an example of an AJAX call with HTTP basic authentication that uses jQuery:

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

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

In this example, `btoa` is a standard function that encodes a string in the Base64 format.

The developer who implements this call can decide how to pass the user name and password.

Note: For the AJAX calls to be successful, host the JavaScript application on the same domain as the MDM Hub.

REST API Reference for Business Entity Services

The REST API reference for business entity services lists the REST APIs and provides a description for each API. The API reference also contains information on the URLs, the query parameters, the sample requests, and the sample responses.

Get Metadata

The Get Metadata REST API returns the data structure of a business entity or a business entity relationship.

The API uses the GET method to return the following metadata of a business entity:

- Structure of the business entity
- List of fields
- Field types such as the data type and size
- Security settings for operations, such as create, update, and merge
- Localized labels for nodes or fields
- Name of the code and display fields for lookup fields
- Default values for fields and lookup fields

  Note: Multiple default values are not supported with dependent lookup fields.

The API returns the following details of a business entity relationship:

- Name of the relationship and its label.
- From and To business entities.
- Direction of the relationship.

Request URL

The Get Metadata URL has the following format for a business entity:

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

The Get Metadata URL has the following format for a relationship:

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

Use the query parameter "action=meta" to retrieve the metadata information. Ensure that you specify the name of the business entity correctly.

Make an HTTP GET request to the Get Metadata URL:

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

You can add HTTP headers to the request.

Sample API Request

The following sample request retrieves metadata information for the Person business entity and root node:

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

The following sample request includes a header to retrieve the metadata information for the Person business entity in the JSON format:

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

The following sample request retrieves metadata information for the HouseholdContainsMemberPerson relationship:

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

Sample API Response

The sample responses for an entity and a relationship contain security permissions. The first `operations` section defines the possible permissions. The `object` section lists your permissions for the entire business entity or relationship. The `fields` section defines your permissions at the field level.

The following example shows a partial data structure for the Person business entity in the JSON format.

```
{
  "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
  }
}
```

List Metadata

The List Metadata REST API returns a list of business entities or relationships that you have defined. The response includes timeline information and security information, if the business entities contain that information. You can use the API to retrieve the list of relationships that start from an entity, end at an entity, or start from and end at specified entities.

The API uses the GET method.

List Metadata URL

The List Metadata URL has the following format for entity metadata:

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

The List Metadata URL has the following format for relationship metadata:

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

Make the following HTTP GET request to the List Metadata URL:

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

Query Parameters

You can append the query parameters to the request URL to filter the search results for the business entity relationships. You can specify the direction and search for relationships to and from a business entity.

The following table lists the query parameters:

Parameter	Description
start	Optional. Specifies the entity from which a relationship originates. For example, the `meta/relationship?start=Organization` query returns all relationships that start from the `Organization` business entity.
finish	Optional. Specifies the entity at which a relationship ends. For example, the `meta/relationship?finish=Person` query returns all relationships that end at the `Person` business entity.

You can specify both the parameters to retrieve all the relationships between two business entities. For example, the `meta/relationship?start=Organization&finish=Person` query returns all relationships that start from the `Organization` business entity and end at the `Person` business entity.

Sample API Request

The following sample request retrieves the list of business entities configured:

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

The following sample request retrieves the list of relationships configured:

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

The following sample request retrieves the list of relationships that start from the `Organization` business entity and end at the `Person` business entity:

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

Sample API Response

The following example shows an excerpt of the list of configured relationships:

```
{
  "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
      }
    },
    ...
  ]
}
```

List Match Columns

The List Match Columns REST API can return either a list of match rule sets for a specified business entity or a list of match columns for a specified match rule set. You can generate a list of match columns and use the match columns with the SearchMatch REST API.

For information on configuring match columns and match rule sets, see the *Multidomain MDM Configuration Guide*.

The API uses the GET method.

Request URL

The context for the List Match Columns URL is `cmx`. In a Hosted MDM environment, include the tenant name in the context, such as `<tenant name>/cmx`.

The List Match Columns URL has the following formats:

URL to return all the match rule sets

Use the following URL to list all the match rule sets for a specific business entity:

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

Make the following HTTP GET request to the List Match Columns URL:

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

URL to return the match columns used in a specified match rule set

Use the following URL to list all the match columns in a specified match rule set:

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

Sample API Request

Request for match rule sets

The following sample request lists the match rule sets for the Person business entity:

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

Request for match columns

The following sample request lists the match columns that are used in the match rule set IDL2:

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

Sample API Response

The following sample response contains the match columns that are included in the IDL2 match rule set:

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

Read Record

The Read Record REST API returns the details of a root record in the business entity. You can use the API to return the details of the child records of a root record. You can use the API to view the content metadata of a record.

The API uses the GET method.

You can sort the result set to view the information in ascending or descending order. Use the POST method when you need more and complex parameters. For example, when you want to retrieve the data where the child elements are ordered by a set of fields.

Request URL

Use the row ID or the name of the source system and the source key to specify the record in the request URL.

The Read Record URL can have the following formats:

URL with rowId

Use the following URL format when you specify the row ID:

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

Make the following HTTP GET request to the URL:

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

URL with source system name and source key

Use the following URL format when you specify the source system name and the source key:

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

URL with system name and global business identifier (GBID) of an object

Use the following URL format when you specify the source system name and the GBID:

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

URL with only the GBID

Use the following URL format when you specify only the GBID:

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

URL with more than one GBID

Use the following URL format when you specify more than one GBID:

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

URL to return the details of the child nodes

Use the following URL format to return the details of the child nodes:

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

URL to return the details of a child nodes

Use the following URL format to return the details of child nodes:

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

For example, `children= BillAddresses/Address,Email`

URL to return the details of a particular node

Use the following URL format to return the details of particular node:

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

URL to return the details of the children of a particular node

Use the following URL format to return he details of the children of a particular node:

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

URL to return the content metadata of a record

Use the following URL format to return the content metadata of a record:

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

For example, you can retrieve matches for the child records with the following GET request:

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

URL to sort the child elements by fields

Use the following URL format to sort the child elements by fields:

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

Use the character the character – as suffix to specify descending order.

Query Parameters

You can append the query parameters to the request URL to filter the details of the record.

The following table lists the query parameters:

Parameter	Description
depth	Number of child levels to return. Specify 2 to return the root node and its direct children, and 3 to return the root node, direct children, and grandchildren. Default is 1.
effectiveDate	Date for which you want to retrieve the data.
readSystemFields	Indicates whether to return the system fields in the result. Default is false.
recordStates	State of the record. Provide a comma-separated list of states. The supported record states are ACTIVE, PENDING, and DELETED. Default is ACTIVE.
contentMetadata	Metadata of the record. Provide a comma-separated list. For example, XREF, PENDING_XREF, DELETED_XREF, HISTORY, XREF_HISTORY, and MATCH. When you select MATCH, the response contains a list of matched records that are retrieved from the _MTCH table.
historyDate	Date for which you want to retrieve history data. The response contains record data for the specified date retrieved from the _HIST table. You can use historyDate together with the contentMetadata parameter to retrieve historical metadata. Set contentMetadata to XREF, BVT, or TRUST. - XREF. The response contains historical cross-reference data from the _HXRF table. - BVT. The response contains historical best version of the truth from the _HCTL table. - TRUST. The response contains historical trust settings from the _HCTL and _HVXR tables.
children	Comma-separated list of child node names or paths.
suppressLinks	Indicates whether the parent-child links are visible in the API response. Set the parameter to true to suppress all parent-child links in the response. Default is false. For example, the `Person/1242?depth=10&suppressLinks=true` query will display the record details up to 10 child levels, with no parent-child links visible in the response.

Parameter	Description
order	Comma-separated list of field names with an optional prefix of + or –. The prefix + indicates to sort the results in ascending order, and the prefix – indicates to sort the results in descending order. Default is +. When you specify more than one parameter, the result set is ordered by the parameter that is first in the list, followed by the next. For example, the `Person/1242/Names?order=-name` query will display the result with the names in descending order. The `Person/1242/BillAddresses?order=rowidObject,-effStartDate` query will display the billing addresses with the rowid in ascending order and then, the effective start date in descending order.

resolveLookup	Retrieves the entire lookup field for a specified business entity. Set the parameter to true to load the lookup field and include it in the response. Default is false.
	For example, the addressType field is a lookup field at the child level of the Person business entity.
	When the resolveLookup parameter is set to false, you might receive the following REST API response:
	`{ "label": "LU Address Type", "addressType": "BILL" }`
	When the resolveLookups parameter is set to true, the REST API response includes additional details and you might receive the following REST API response:
	`{ "label": "LU Address Type", "addressType": "BILL", "addressTypeDisp": "BILLING" }`

The following example shows how to filter the details of a record:

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

Related Topics:

- "Formats for Dates and Time in UTC" on page 34

POST Request to Specify Sort Order of Child Elements

Use a POST request to order the result set by multiple fields. Include the parameters or fields in the POST body.

The following sample request shows how to use the POST request for a read operation and sort the data by multiple fields:

```
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"}]
      }
   ]}
 }
}
}
```

Note: At each level of business entity, for each type of children only one type of sort order is allowed.

Sort Order Considerations

The Read Record API supports sorting order by one or more fields for each business entity child node. The following section describes certain considerations that you need to be aware of when you specify the sort order.

- If you specify the sort order for the grandchild and not for the child, the grandchild element is sorted as per the order specified. The child element is not sorted by the sort order specified for the grandchild. The following is a sample request:

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

  In the sample request, while a descending sort order is specified for grandchild PhonePermissions, no order is specified for the child element Phone. Phone is not sorted by the PhonePermissions sort order.

- If you specify the sort order for the child and not for the grandchild, the child is sorted by the sort order specified. The grandchild is not sorted by the sort order specified for the child. The following is a sample request:

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

  In the sample request, the sort order is specified for child Phone and not for grandchild PhonePermissions. The child Phone is sorted by the order specified.

- If you specify the sort order for the child and the grandchild, both are sorted as per the sort order. The following sample request specifies the sort order for the Phone (child) and the PhonePermissions (grandchild):

  ```
  {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"}]},
      ]}
  }}}
  ```

- A child can be sorted only by the columns in the child itself, while the grandchild can be sorted by the columns in the grandchild. In the following sample requests, the Phone is sorted by PhoneType and the PhonePermissions is sorted by column 1. PhoneType is a column in Phone (child) and column 1 is a column in PhonePermissions (grandchild).

  ```
  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
  ```

- At each level of business entity, for each type of children only one type of sort order is allowed. In the following request, you have specified different sort orders for the PhonePermissions children of different parents. However, because first the sort order is specified as descending, the PhonePermissions children of both parents (rowid 861 and rowid 862) are sorted in descending order.

  ```
  {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"}]}
      ]}
  }}}
  ```

Sample API Requests

The following sample request returns the details of the root record in the Person business entity.

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

The following sample request returns the details of a child record with rowId 2. The child base object is genderCd and the child record is at a depth of 2.

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

The following sample request uses the system name and source key to returns the details of the root record:

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

The following sample request returns the details of the root record and its XREF records:

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

The following sample request returns the details of the root record, with the names in descending order:

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

The folllowing sample request returns the billing addresses with the rowid in ascending order and then, the effective start date in descending order:

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

Sample API Response

The following example shows the details of the root record in the Person business entity.

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

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

The following example shows the details of a child record in the genderCd base object.

```
{
    "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"
}
```

Create Record

The Create Record REST API creates a record in the specified business entity. Send the record data in the request body. Use the Promote API to promote and add the record in the business entity.

The API uses the POST method to create a record.

Request URL

The Create Record URL has the following format:

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

Note: The name of the source system is a required parameter in the URL.

Make the following HTTP POST request to the Create Record URL:

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

Add the Content-Type header to specify the media type of the data you want to send with the request:

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

URL Parameters

The name of the source system is a required parameter in the request URL.

The following table lists the parameters that you can use in the URL:

Parameter	Description
systemName	Name of the source system.
interactionId	ID of the interaction. You can group multiple requests into a single interaction. All changes are done with the interaction ID.
ignoreWarnings	Indicates whether the warning-level validation messages are ignored. Default is false. Set to true to ignore the warning messages.
startDate and endDate	Specifies the period of time for which the record is effective. Provide these parameters for a timeline-enabled base object.
validateOnly	Indicates whether the write business entity service validates incoming data. Default is false.

Parameter	Description
recordState	State of the record. Use the parameter to specify the initial state of the record. Use ACTIVE or PENDING. Default is ACTIVE.
taskComment	Add a comment to the workflow task triggered by the API.
taskAttachments	If task attachments are enabled, attach a file to the workflow task triggered by the API.

Related Topics:

- "Formats for Dates and Time in UTC" on page 34

Request Body

Send data for the record in the REST request body. Use the JSON format or the XML format to send data. Use the Get Metadata API to get the structure of the business entity and provide the required parameter values in the request body.

After you submit a REST API request to create a business entity record, Informatica MDM Hub calculates the trust score of the field. You can submit a REST API request for empty fields with calculated trust scores. You must specify double quotation marks in the request body.

The following JSON code sample requests the empty Gender Code field with a calculated trust score:

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

If you use null instead of double quotation marks, Informatica MDM Hub does not calculate the field trust score.

Response Header and Body

When the response is successful, the API returns the interactionId and the processId in the response header and the record details in the response body.

If the process generates an interaction ID and uses it to create the record, the API returns the interaction ID. If the process starts a workflow instead of directly saving the record to the database, the API returns the process ID which is the ID of the workflow process.

The following example shows a response header with an interaction ID and a process ID:

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

The response body contains the record with the generated row IDs.

Sample API Requests

The following sample request creates a record in the Person business entity. The request adds a comment and attachment to the workflow task that the API triggers.

```
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"
         }
      ]
   }
}
```

When you use a POST REST API request to create a new record, you can set the `sourceKey` attribute in the POST request body. The `sourceKey` attribute is used as the `pkey_src_object` value in the XREF table. For example, yu might send the following POST request to create a new Party record and the Party XREF record with the source system SFA and the `pkey_src_object` 123:

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

{
    "firstName": "John",
    "middleName": "jr",
    "lastName": "Smith",
    "genderCd": {
        "genderCode": "M"
    },
    "key": {
        "sourceKey": "123"
    }
}
```

Sample API Response

The following sample response shows the response header and body after successfully creating a record:

```
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"
                }
            ]
        }
    }
}
```

Update Record

The Update Record REST API updates the specified root record and its child records. Send the ID of the record in the request URL. Send the summary of the changes in the body of the request.

After the change, if the record is in a pending state, use the Promote API to promote the changes. For example, if the update triggered a review workflow, the record is in a pending state until the review is complete.

The API uses the POST method.

For root records, you can choose to use the simplified PUT version, where the body of the request contains the changed field, not a change summary. To update root records and related child or grandchild records, use the POST version. You also have the option to use the PUT version with the PKEY or the specify the rowidObject of the child record.

Request URL

The Update Record URL has the following format:

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

Note: The name of the source system is a required parameter in the URL.

Make the following HTTP POST request to the Update Record URL:

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

Add the Content-Type header to specify the media type of the data you want to send with the request:

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

Query Parameters

The name of the source system is a required query parameter.

You can use the following query parameters in the request:

Parameter	Description
systemName	Name of the source system.
interactionId	ID of the interaction. You can group multiple requests into a single interaction. All changes are done with the interaction ID.
ignoreWarnings	Indicates whether the warning-level validation messages are ignored. Default is false. Set to true to ignore the warning messages.
startdate and enddate	Specifies the period of time for which the record is effective. Provide these parameters for a timeline-enabled base object.
validateOnly	Indicates whether the incoming data is validated. Default is false.
recordState	Sets the state of the record. Use ACTIVE, PENDING, or DELETED. For example, if you set`recordState=ACTIVE`, and request runs on a soft-deleted record, the request restores the record to the active state.

Parameter	Description
taskComment	Add a comment to the workflow task triggered by the API.
taskAttachments	If task attachments are enabled, attach a file to the workflow task triggered by the API.

Related Topics:

- "Formats for Dates and Time in UTC" on page 34

Request Body

Send the data to update in the REST request body. Use the JSON format or the XML format to send the data.

Provide the new parameter values. Use the $original parameter to indicate the old value of a parameter you want to update.

You can also use the following properties with child records:

Properties / Elements	Type	Description
MATCH	object	If you want to add or remove match candidates from the match table for the child record, add the MATCH object to the child record.
MERGE	object	If you want to merge child records or remove candidates from the merge, add the MERGE object to the child record.

The following JSON code sample changes the first name in the root record to Bob:

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

The following JSON code sample removes a match candidate for an Address child record, and defines the merge for two Telephone child records:

```
{
  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
            ]
          }
        }
      ]
    }
}
```

After you submit a REST API request to update a business entity record, Informatica MDM Hub calculates the trust score of the field. You can submit a REST API request for empty fields with calculated trust scores. You must specify double quotation marks in the request body.

The following JSON code sample requests the empty Gender Code field with a calculated trust score:

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

If you use null instead of double quotation marks, Informatica MDM Hub does not calculate the field trust score.

Response Header

When the response is successful, the API returns the interactionId and the processId in the response header and the record details in the response body.

If the process generates an interaction ID and uses it to create the record, the API returns the interaction ID. If the process starts a workflow instead of directly saving the record to the database, the API returns the process ID which is the ID of the workflow process

The following example shows a response header with an interaction ID and a process ID:

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

The response body contains the record with the generated rowIds.

Sample API Request

The following sample request updates a root record and its child record in a business entity. Person is the business entity and Phone is a child base object. The request adds a comment and attachment to the workflow task triggered by the API.

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

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

Sample API Response

The following sample response shows the response header and body after successfully updating a record:

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

Delete Record

The Delete Record REST API deletes a root record in a business entity. Use the API to delete the child records of a root record.

The API uses the DELETE method to delete a record.

Request URL

The Delete record URL has the following format:

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

Note: The name of the source system as a required parameter in the URL.

Make the following HTTP DELETE request to the Delete Record URL:

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

Use the following URL format to delete a child record of a root record:

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

Query Parameter

The name of the source system is a required URL parameter. Use the systemName parameter to specify the source system.

Sample API Request

The following sample request deletes a root record in the Person business entity:

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

Sample API Response

The following sample response shows the response after successfully deleting a root record in the Person business entity:

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

List Record

The List Record REST API returns the list of lookup values or foreign key values. Lookups provide reference data and give a list of possible values for a given column.

The API uses the GET method.

Use the API to also get the lookup code values and lookup code descriptions. You can specify the sort order for the lookups. Use the POST method when you need more and complex parameters.

Request URL

The List Record REST URL has the following format:

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

Make the following HTTP GET request to the URL:

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

Use the following URL format to list the codes of the lookup values:

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

Note: Use the Get Metadata API to get the exact URL to list the lookup values.

Query Parameters

You can append the query parameters to the request URL to filter the results.

The following table lists the query parameters:

Parameter	Description
suppressLinks	Indicates whether the parent-child links are visible in the API response. Set the parameter to true to suppress all parent-child links in the response. Default is false. For example, the `Person/1242?depth=10&suppressLinks=true` query will display the record details up to 10 child levels, with no parent-child links visible in the response.
order	Used to list the lookup values in ascending or descending order. Use the character + as prefix to specify ascending order and the character – as prefix to specify descending order. By default, when you do not specify the prefix, the result set is ordered in ascending order. For example, the `LUNamePrefix?action=list&order=-namePrefixDisp` query lists the prefixes for name, which are sorted by the display names of the prefixes in descending order.

resolveLookup	Retrieves the entire lookup field for a specified business entity. Set the parameter to true to load the lookup field and include it in the response. Default is false.
	For example, the addressType field is a lookup field at the child level of the Person business entity.
	When the resolveLookup parameter is set to false, you might receive the following REST API response:
	`{ "label": "LU Address Type", "addressType": "BILL" }`
	When the resolveLookups parameter is set to true, the REST API response includes additional details and you might receive the following REST API response:
	`{ "label": "LU Address Type", "addressType": "BILL", "addressTypeDisp": "BILLING" }`

POST Request to Specify Sort Order

Use a POST request to specify the sort order for the lookup values. Include the parameters or fields in the POST body.

The following example shows how to use the POST request for a list operation:

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

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

Sample API Request

The following sample request list the lookup values:

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

The following sample request lists the codes for the gender lookup values:

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

The following sample request lists the prefixes for names, with the display names of the prefixes in descending order:

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

Sample API Response

The following sample response shows the list of lookup values:

```
{
    "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"
        }
    ]
}
```

The following sample response shows the code for the lookup values:

```
{
    "item": [
        {
            "id": "F",
            "label": "FEMALE"
        },
        {
            "id": "M",
            "label": "MALE"
        },
        {
            "id": "N",
            "label": "UNKNOWN"
        }
    ],
```

```
    "firstRecord": 1,
    "recordCount": 0,
    "pageSize": 2147483647,
    "searchToken": "SVR1.AU5LD"
}
```

Search Record

The Search Record REST API searches for indexed values in a searchable root record and in all the child records. You can use filters and facets to view a subset of the search results. Facets group the search results into categories, while filters narrow down the search results. The API searches all fields that are configured as searchable and returns the records that match the search criteria.

The API uses the GET method to search the indexes of the searchable fields.

Request URL

The Search Record URL has the following formats:

URL for a simple search

Use the following URL for a simple search:

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

Make the following HTTP GET request to the Search Record URL:

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

URL for a fielded search

Use the following URL for a fielded search:

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

If you specify a negative numerical value for a business entity field, such as `-120`, the ranking of the records that are returned might be affected.

URL for a facets search

Use the following URL for a simple search with facets:

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

Use the following URL for a fielded search with facets:

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

URL for a filter search

Use the following URL for a simple search with filters:

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

Use the q or the fq parameter in the search.

URL encoding

Use URL encoding because the URL includes characters, such as spaces and single quotation marks.

The following example shows the URL encoded representation of the Search Record URL:

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

Query Parameters

Use the q or the fq query parameters to provide the string value for the search. The q and the fq query parameters are mutually exclusive. Use the fq parameter for a fielded search. Use the AND logical operator for multiple conditions.

The following table lists the parameters that you can use in the URL:

Parameter	Description

q	Specifies the string value or the search term. The query searches for occurrences of the search term anywhere in a record. Used in a simple search.
	For example, the `Person?q=STEVE` query searches for records with the term STEVE.
	To search for two or more terms together, include the terms in double quotation marks. Use the character + before each term if you want the search results to contain the term. If the field value contains a space, enclose the field value in single quotes.
	Use the following query to search for an exact match to WILLIAM JOHN LAWSON:
	`Person?q="WILLIAM JOHN LAWSON"`
	Use the following query to search for WILLIAM, JOHN or LAWSON:
	`Person?q=WILLIAM JOHN LAWSON`
	Use the following query to search for WILLIAM, JOHN, and LAWSON:
	`Person?q=WILLIAM JOHN LAWSON&queryOperator=AND`
	If the search term contains the ampersand (&) special character, encode the character as %26. Otherwise, the search request does not return the expected results.
	Use the following query to search for Johnson&Johnson:
	`Organization?q=Johnson%26Johnson&queryOperator=AND`

fq	Specifies the string value or search term in a particular field. The query searches for the term only in that part of a record. Used in a targeted search based on indexed fields. For example, the `Person?fq=displayname=STEVE` query searches for records with the display name STEVE. If the search term contains the ampersand (&) special character, encode the character as %26. Otherwise, the search request does not return the expected results. For example, use the following query to search for Mac&Cheese: `Product?fq=fname=Mac%26Cheese&queryOperator=AND`
facets	Specifies the fields that should be treated as facets or categories by which the search results are grouped. Specify only searchable fields. Used with the q and the fq parameters. Syntax is `&facets=FieldName1,FieldName2,FieldNameN` For example, the `Person?q=STEVE&facets=department` query searches for persons with the display name STEVE and groups the search results by the departments. Th search displays the records of the persons with the display name STEVE and these records are grouped by the departments.

Parameter	Description
filters	Specifies the fields by which you can narrow down the search results. Specify only filterable fields. Used with the q and the fq parameters. For example, the `Person?fq=displayname=STEVE&filters=birthdate='1980-11-27T08:00:00Z'` query searches for persons with the display name STEVE and filters the search results by the birth date. The search displays the records of the persons who have the display name STEVE and whose date of birth is 27 November, 1980. **Note:** Specify a date within single quotation marks.
depth	Specifies the number of child levels to return. Specify 2 to return the root node and its direct children, and 3 to return the root node, direct children, and grandchildren. Specify 1 to return the root node alone. By default, no depth is specified. If no depth is specified, then the search results return the root node and children where a match for the search term is found. For example, the `Person?q=STEVE&depth=2` query searches for records with the term STEVE and returns information about the root record and its direct children.
queryOperator	Specifies whether the search finds any of the strings in the search term or all of the strings in the search term. The parameter takes one of the following values: - `OR`. Searches for any of the strings listed in the `f` or `fq` parameter. - `AND`. Searches for all of the strings listed in the `f` or `fq` parameter. If you do not specify this parameter, the default is `OR`. For example, the `Person?q=WILLIAM JOHN LAWSON&queryOperator=AND` query searches for records that contains WILLIAM, JOHN, and LAWSON.
suppressLinks	Indicates whether the parent-child links are visible in the API response. Set the parameter to true to suppress all parent-child links in the response. Default is false. For example, the `Person?q=STEVE&suppressLinks=true` query searches for records with the term STEVE and returns the response where no parent-child links are visible.
readSystemFields	Indicates whether to return the system fields in the result. Default is false.
order	Comma-separated list of field names with an optional prefix of + or –. The prefix + indicates to sort the results in ascending order, and the prefix – indicates to sort the results in descending order. Default is +. If you want to use a child field to sort the results, use the full name of the field. For example, `BillAddresses.Address.cityName`. When you specify more than one parameter, the result set is ordered by the parameter that is first in the list, followed by the next. For example, the `Person?order=displayName,-BillAddresses.Address.cityName` query sorts the results by display name in ascending order and then by city name in descending order.
maxRecordsToSort	Maximum number of search results that you want to sort. Default is 1000.
resolveLookup	Retrieves the entire lookup field for a specified business entity. Set the parameter to true to load the lookup field and include it in the response. Default is false.

Specify a range with the `filters` parameter:
You can use the filters parameter to narrow down the search results within a specified range. You can specify the range for filterable fields of the numeric and the date data types.

Use the following format for the integer data type:

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

The range is from the fromValue to the toValue. Ensure that the fromValue is lower than the toValue. For example, the `filters=age=[35,45]` query narrows the search results and searches for records in the age group of 35 to 45.

Use the following format for the date data type:

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

The range is from the fromDate to the toDate. For example, the `filters=birthdate=[2000-06-12T12:30:00Z,2015-06-12T12:30:00Z]` query specifies the date of birth between 12 June 2000 and 12 June 2015.

Note: When you specify a exact match date filter, enclose it within single quotation marks. When you specify a date range, do not use quotation marks.

Related Topics:

- "Formats for Dates and Time in UTC" on page 34

Sample API Request

Request with the q parameter

The following sample request searches the Person business entity for records with the name STEVE.

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

Request with the fq parameter

The following sample request searches the Person business entity for records with the display name STEVE. The displayName field is an indexed field.

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

Request with the sort option

The following sample request searches the Person business entity for records with the display name STEVE and sorts the results by city in ascending order:

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

Request with the fq parameter and the AND logical operator

The following sample request searches the Person business entity for records with the display name STEVE and the tax ID DM106:

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

Request with a facet

The following sample request searches the Person business entity for records with the display name STEVE and narrows down the results by grouping them into departments:

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

Request with a filter (exact filter)

The following sample request searches the Person business entity for records with the display name STEVE and filters by the specified city and country:

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

Request with a filter range

The following sample request searches the Person business entity for records with display name STEVE and filters by the age group 35 to 45:

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

Sample API Response

The following sample response shows the result of a search by the name STEVE:


```
{
    "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"
            }
        }
    ]
}
```

Suggester

The Suggester REST API returns a list of related terms for a search string, based on the data present in your database. Use the API to accept the characters that you type in a user interface field and return suggestions to autocomplete what you type. You can find and select the string from the list of suggestions. Use the Suggester API for searchable fields.

The API uses the GET method.

Note: To use the API to provide a list of autocomplete suggestions for a searchable field, set the `suggester` property of the field to true and reindex the data.

Request URL

The Suggester REST URL has the following format:

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

Make the following HTTP GET request to the URL:

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

Query Parameters

The following table lists the query parameters:

Parameter	Description
suggest	Required. Specifies the string that you want suggestions for.
recordsToReturn	Specifies the number of rows to return.
buildIndex	Optional. Indicates whether to build the index. When you restart the system, set this parameter to true to explicitly build the index for the collection. This parameter might be deprecated in a future release.

Sample API Request

The following sample request returns a list of suggestions that you can use in a user interface:

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

Sample API Response

The following sample response shows the list of suggestions:

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

}
```

SearchQuery

The SearchQuery REST API searches for records that are an exact match of the field values specified in a query. You can use the SearchQuery API to retrieve all the records for a specific business entity or to retrieve

records based on specific field values. Unlike the Search Records API that searches for the specified values across all the searchable fields, the SearchQuery API searches for the specified values within specific fields.

You can filter the query results to display specific values in a root business entity record and in the child records. You can use the AND, IN, and RANGE operators in the query.

To include specific fields in the query results, specify the fields or a business entity view that includes the fields. You can sort the query results to view the records in ascending or descending order.

The API uses the GET method.

Request URL

The context for the SearchQuery URL is `cmx/cs`. In a Hosted MDM environment, include the tenant name in the context, such as `<tenant name>/cmx/cs`.

The SearchQuery URL has the following formats:

URL to return all the records of a specific business entity type

Use the following URL to search for all the records of the specified business entity type:

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

Make the following HTTP GET request to the SearchQuery URL:

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

URL to return all the details of records that match the specified field values

Use the following URL to search for records that match the field values that you specify:

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

URL to return specific details of records that match the specified field values

Use the following URL to search for records and display specific record fields in the search results:

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

Query Parameters

Define the query as a list of field-value pairs.

The following table describes the query parameters that you can use in the URL:

Parameter	Description
action	Required. Returns all the records of the specified business entity type in the query results. Set to `query`, and use the parameter with the `filter` parameter. When used without the `filter` parameter, the query searches for all the records of the specified business entity type. For example, use the following query to search for all the Person business entity records: `Person?action=query`
filter	Required. Specifies a list of field-value pairs separated by operators. Valid operators are AND, IN, and Range. For example, use the following query to search for the Person records with the first name STEVE and the last name SMITH: `Person?action=query&filter=firstName='STEVE' AND lastName='SMITH'`
depth	Specifies the number of child record levels to return. For example, you can specify the following levels: - 1. Return the root record. - 2. Returns the root record and its direct child records. - 3. Returns the root record, the direct child records, and grandchild records. For example, use the following query to search for records with the first name STEVE and return information about the root record and its direct child records: `Person?action=query&filter=firstName='STEVE' AND lastName='SMITH'&depth=2`
suppressLinks	Indicates whether the parent-child links are visible in the API response. Set the parameter to true to suppress all the parent-child links in the response. Default is false. For example, use the following query to search for records with the first name STEVE and return a response where no parent-child links are visible: `Person?action=query&filter=firstName='STEVE'&suppressLinks=true`
readSystemFields	Indicates whether to return the system fields in the result. Default is false.
fields	Specifies the fields to display in the query results.
outputView	Specifies the business entity view that you want to use to display the query results. When you configure the business entity view for the query results, include the fields that you want to display in the query results.

Parameter	Description
Order	Specifies the sort order of the query results. Use the plus (+) character as prefix to specify ascending order and the minus (-) character as prefix to specify descending order. By default, the query result is in ascending order. When you specify more than one parameter, the result set is ordered by the parameter that is first in the list, followed by the next parameter.

resolveLookup	Retrieves the entire lookup field for a specified business entity. Set the parameter to true to load the lookup field and include it in the response. Default is false.
	For example, the addressType field is a lookup field at the child level of the Person business entity.
	When the resolveLookup parameter is set to false, you might receive the following REST API response:
	`{ "label": "LU Address Type", "addressType": "BILL" }`
	When the resolveLookups parameter is set to true, the REST API response includes additional details and you might receive the following REST API response:
	`{ "label": "LU Address Type", "addressType": "BILL", "addressTypeDisp": "BILLING" }`

Operators

You can use the following operators within the filter parameter:

AND

Searches for records with all the field values listed in the filter parameter.

For example, use the following query to search for records with the first name STEVE and the last name SMITH:

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

IN

Searches for records with any of the values listed in the filter parameter.

For example, use the following query to search for records with the first name STEVE or JOHN:

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

Range

Searches for records within a specified range. You can specify a range for the numeric and the date data type fields.

Use the following format for the integer data type:

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

The range is from the fromValue to the toValue. Ensure that the fromValue is lower than the toValue.

For example, use the following query to search for records in the age group 35 to 45:

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

Use the following format for the date data type:

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

The range is from the fromDate to the toDate.

For example, use the following query to search for records with the date of birth between 12 June 2000 and 12 June 2015:

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

!=

Searches for records that do not match a specified field value or range.

For example, use the following query to search for the records with the first name other than ADAM:

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

For example, use the following query to search for the records with the first name other than ADAM and with the date of birth prior to 16 November 2017 and after November 16 2020.

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

Wildcard

You can use text and an asterisk (*) wildcard operator to specify a text pattern instead of a complete search string. You can use the asterisk wildcard operator to increase your chances of finding the records that you want. The asterisk wildcard operator is useful when you do not know the exact text or want to search for similar text.

The following table lists example search strings and explains how they function:

Example Query String	Query Behavior
John*	Queries for records that contain a value that starts with `John`. For example, `Johnson` or `Johnny`.
Jo*n	Queries for records that start with `Jo` and end with `n`. For example, `Johansson` or `Jordan`.
*	Returns all the records.

Sample API Requests

This topic provides sample request queries.

The following sample request queries the Person business entity for records with the first name STEVE and last name SMITH:

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

The following sample request queries the Person business entity for records with corresponding original ROWIDs. In this example, only one record is returned, based on the value '1' used as a unique identifier ROWID_OBJECT for this Person.

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

The following sample request queries the Person business entity for surviving records with corresponding ROWIDs. In this example, only one record is returned, based on the value '1' used as a unique identifier ROWID_OBJECT for this Person.

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

Sample API Response

The following sample response shows the results of a query for the Person records with the first name STEVE and last name SMITH:

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

Exporting the SearchQuery Results to a CSV File

To export the results of a SearchQuery request to a CSV file, in the request URL path, specify the name of the business entity as a `.CSV` file. You can use all the query parameters in the request URL.

For example, use the following request URL to export the results of a search for records that match the field values that you specify:

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

Sample API Request

The following sample request queries for records with the first name STEVE and last name SMITH and returns the query results in the CSV format:

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

Sample API Response

The following sample shows the query results in the CSV format for a query with the first name STEVE and last name SMITH:

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

SearchMatch

The SearchMatch REST API searches for records that are fuzzy matches. The API identifies the matching records based on specific business entity fields that are configured as match columns in the MDM Hub. Before you use the SearchMatch API, use the List Match Columns API to identify the match columns for a business entity.

Optionally, you can specify a match rule set instead of match columns. To use a match rule set for the query, ensure that the Enable Search by Rules option is enabled for the match rule set. For information on configuring match columns and match rule sets, see the *Multidomain MDM Configuration Guide*.

You can use the AND, IN, and RANGE operators in the query.

To include specific fields in the query results, specify the fields or a business entity view that includes the fields. You can specify the sort order to view the records in the query results in ascending or descending order.

The API uses the GET method to query the business entity fields and returns the records that are fuzzy matches along with their match scores and associated match rules.

Request URL

The context for the SearchMatch URL is `cmx/cs`. In a Hosted MDM environment, include the tenant name in the context, such as `<tenant name>/cmx/cs`.

The SearchMatch URL has the following formats:

URL to return matching records based on the values of specific fields that are configured as match columns

Use the following URL to search for records that match the field values that you specify:

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

Make the following HTTP GET request to the SearchMatch URL:

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

URL to return matching records based on a match rule set

Use the following URL to search for matching records based on a match rule set that you specify:

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

Query Parameters

Use the `fuzzyFilter` parameter to specify the field values that you want to query. Use the `fuzzyFilter` parameter with the `action` parameter.

The following table describes the query parameters that you can use in the URL:

Parameter	Description
action	Required. Returns the matching records for the specified business entity. Set to `match`, and use the parameter with the `fuzzyFilter` parameter. For example, use the following query to search for a person with the first name STEVE: `Person?action=match&fuzzyFilter=STEVE`
fuzzyFilter	Required. Specifies a comma-separated list of field name and field value pairs that you want to use to query for records of a specific business entity type. For example, use the following query to search for records with the first name STEVE, who have a Toronto address: `Person?action=match&fuzzyFilter=firstName=STEVE,Address.Address.City=TORONTO`
matchRuleSet	Specifies a match rule set based on which you want to identify the matching records. If you do not have a specific match rule set, specify NONE. The automatic and the manual merge match rules are used.

filter	Specifies the field values to use to filter the results of a fuzzy search.
	For example, use the following query to search for records with the first name STEVE, who live in Toronto:
	`Person?`
	`action=match&fuzzyFilter=firstName='STEVE',lastName="SMITH'&filter=city=Toronto`

depth	Specifies the number of child record levels to return. For example, you can specify the following levels: - 1. Return the root record. - 2. Returns the root record and its direct child records. - 3. Returns the root record, the direct child records, and grandchild records. For example, use the following query to search for records with the first name STEVE and return information about the root record and its direct child records: `Person?action=match&fuzzyFilter=firstName='STEVE'&filter=city=Toronto`
suppressLinks	Indicates whether the parent-child links are visible in the API response. Set the parameter to true to suppress all the parent-child links in the response. Default is false. For example, use the following query to search for records with the first name STEVE and return a response where no parent-child links are visible: `Person?action=match&fuzzyFilter=firstName='STEVE'&suppressLinks=true`
readSystemFields	Indicates whether to return the system fields in the result. Default is false.
fields	Specifies the fields to display in the query results.

Parameter	Description
outputView	Specifies the business entity view that you want to use to display the query results. When you configure the business entity view for the query results, include the fields that you want to display in the query results.

resolveLookup	Retrieves the entire lookup field for a specified business entity. Set the parameter to true to load the lookup field and include it in the response. Default is false.
	For example, the addressType field is a lookup field at the child level of the Person business entity.
	When the resolveLookup parameter is set to false, you might receive the following REST API response:
	`{ "label": "LU Address Type", "addressType": "BILL" }`
	When the resolveLookups parameter is set to true, the REST API response includes additional details and you might receive the following REST API response:
	`{ "label": "LU Address Type", "addressType": "BILL", "addressTypeDisp": "BILLING" }`

You can use the following operators within the filter parameter:

AND

Searches for records with all the field values listed in the filter parameter.

For example, use the following query to search for records with the first name STEVE and the last name SMITH:

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

IN

Searches for records with any of the values listed in the filter parameter.

For example, use the following query to search for records with the first name STEVE or last name JOHN, who live in the city Toronto or Ottawa:

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

Range

Searches for records within a specified range. You can specify a range for the numeric and the date data type fields.

Use the following format for the integer data type:

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

The range is from the fromValue to the toValue. Ensure that the fromValue is lower than the toValue.

For example, use the following query to search for records in the age group 35 to 45:

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

Use the following format for the date data type:

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

The range is from the fromDate to the toDate.

For example, use the following query to search for records with the date of birth between 12 June 2000 and 12 June 2015:

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

Sample API Request

The following sample request queries the Person business entity for records with the first name STEVE by using the match rule set IDL2:

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

Sample API Response

The following sample response shows the results of a query for records with the first name STEVE based on the match rule set IDL2:

```
{
  "link": [],
  "firstRecord": 1,
  "recordCount": 3,
  "pageSize": 10,
  "searchToken": "SVR1.17LJ2",
  "matchedEntity": [
    {
      "businessEntity": {
        "Person": {
          "rowidObject": "145           ",
          "label": "SAMUEL,STEVE",
          "partyType": "Person",
          "lastName": "SAMUEL",
          "displayName": "MR STEVE SAMUEL                                        ",
          "statusCd": "A ",
          "firstName": "STEVE",
          "genderCd": {
            "genderCode": "M"
          },
          "namePrefixCd": {
            "namePrefixCode": "MR"
          }
        }
      },
      "matchRule": "IDL2|1",
     "matchScore": "93",
      "link": [
        {
          "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI2/meta/matchRule/
Person/IDL2|1.json",
          "rel": "matchRule"
        }
      ]
    },

    {
      "businessEntity": {
        "Person": {
          "rowidObject": "268           ",
          "label": "SMITH,STEVE",
          "partyType": "Person",
          "lastName": "SMITH",
```

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

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

Exporting the SearchMatch Results to a CSV File

To export the results of a SearchMatch request as a CSV file, in the request URL path, specify the name of the business entity as a `.CSV` file. You can use all the query parameters in the request URL.

For example, use the following request URL to export the results of a search for matching records based on a match rule set:

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

Sample API Request

The following sample request searches for records that match the first name STEVE and last name SMITH and returns the query results in the CSV format:

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

Sample API Response

The following sample shows the query results for records that match the first name STEVE and last name SMITH in the CSV format:

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

```
00035,Steven,Smith
00048,Steve,Smith
00079,Steve,Smithson
```

Get BPM Metadata

The Get BPM Metadata REST API returns the task types and two indicators that specify whether the BPM workflow tool can perform the Get Task Lineage service and the administration services.

The API uses the GET method.

Request URL

The Get BPM Metadata URL has the following format:

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

Make the following HTTP GET request to the Get BPM Metadata URL:

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

Sample API Request

The following sample request returns information about the task types and the BPM workflow tool:

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

Sample API Response

The following sample response shows the task types and the value of the two indicators for the BPM workflow tool:

```
{
    "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"
                }
            ]
        }
```

```
        }
    }
```

List Tasks

The List Tasks API returns a list of workflow tasks. A workflow defines the activities in a business process and the paths of execution through the activities. Each activity is called a task.

The API uses the GET method to return a sorted and paged list of tasks.

Request URL

The List Tasks URL has the following format.

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

Make the following HTTP GET request to the List Tasks URL:

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

You can add HTTP headers to the request.

Query Parameters

Use the task data fields as query parameters to filter the list of tasks.

You can use the following query parameters:

Parameter	Description
taskType	A set of actions that you can perform on a record. Use the name attribute to specify the task type. For more information about task types, see the *Multidomain MDM Data Director Implementation Guide*.
taskId	ID of the task.
processId	ID of the workflow process that contains the task.
owner	User who performs the task.
title	Short description for the task.
status	Status of the task in the workflow. Use one of the following two values: - Open: Task has not started or is in progress. - Closed: Task is completed or is cancelled.
priority	Level of importance of the task. Use one of the following values: high, normal, and low.
creator	User who creates the task.
createDateBefore and createDateAfter	Date range. You can filter the tasks by the createDate field.
dueDateBefore and dueDateAfter	Date range. You can filter the tasks by the dueDate field.

Parameter	Description
scope	Shows available tasks based your user role. Use the `admin` value to filter the list of tasks based on your role. **Note:** Users assigned the Task Administrator role can manage all tasks, while users assigned other user roles can only manage tasks available to their role.
q	Specifies the search term. The query searches for occurrences of the search term in tasks related to business entity records. For example, the q=KAREN query searches tasks with the search term KAREN.

Use the query parameters as name-value pairs in the request URL.

The following example shows how to use the query parameters to filter tasks:

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

Related Topics:

- "Formats for Dates and Time in UTC" on page 34

Sort Parameters

To determine the sort order in the REST API response, use the generic sort parameter and provide a comma-separated list of task fields. You can specify the sort order for each field. Use the dash sign (-) to specify descending order. The default sort order is ascending.

The following example shows how to sort the results:

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

Sample API Request

The following sample request retrieves the list of tasks:

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

The request does not contain a body.

Sample API Response

The following sample response shows the list of tasks in the JSON format:

```
{
    "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"
        }
    ]
}
```


Read Task

The Read Task REST API returns the details of a task, such as task type, priority, and status.

The API uses the GET method.

Request URL

The Read Task URL has the following format:

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

Note: Use the List Tasks API to get the ID of the task.

Make the following HTTP GET request to the Read Task URL:

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

Sample API Request

The following sample request returns the details of a task:

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

Sample API Response

The following sample response shows the details of the task:


```
{
    "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"
}
```

Create Task

The Create Task REST API creates a task and starts a workflow.

The API uses the POST method to create a task and returns the ID of the workflow process that contains the task.

Request URL

The Create Task URL has the following format:

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

Make the following HTTP POST request to the Create Task URL:

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

Add the Content-Type header to specify the media type of the data you want to send with the request:

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

Request Body

Specify the task attributes when you create the task. Use the JSON format or the XML format to send the task data in the request.

The following table describes the task parameters in the request body:

Parameter	Description
taskType	A set of actions that you can perform on a record. Use the name attribute to specify the task type. For more information about task types, see the *Multidomain MDM Data Director Implementation Guide*.
owner	User to whom the creator assigns the task.
title	Short description for the task.

Parameter	Description
comments	Comments for the task.
attachments	Attachments for the task.
dueDate	Date when the owner must complete the task.
status	Status of the task in the workflow. Use one of the following two values: - Open: Task has not started or is in progress. - Closed: Task is completed or is cancelled.
priority	Level of importance of the task. Use one of the following values: high, normal, and low. Default is normal.
creator	User who creates the task.
createDate	Date when you create the task.
orsId	ID of the Operational Reference Store (ORS) as registered in the Databases tool in the Hub Console.
processId	ActiveVOS® task type ID. For more information, see the *Multidomain MDM Data Director Implementation Guide*.
taskRecord	The business object root record or the cross-reference record associated with the task. Use the row ID or the source system and source key to specify the record.
businessEntity	Name of the business entity to which the taskRecord belongs.
interactionId	ID of the interaction. Use the interaction ID to keep a task context relationship between a task and records.
groups	Assign a task to all users in the specified user groups. You define user groups in the MDM Hub Console. Specify the groups as an array.

The following sample code uses the rowId to specify the taskRecord:

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

The request body has the following format:

```
{
     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.
         }
       }
     }]
}
```

Related Topics:

- "Formats for Dates and Time in UTC" on page 34

Sample API Request

The following sample request creates a task for a root record:

```
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"
         }
       }
     }]
}
```

Sample API Response

The following example shows the response on successfully creating a task. The API returns the ID of the workflow process that contains the task.

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

Update Task

The Update Task REST API updates a single task.

The API uses the PATCH method to update some task fields and the PUT method to update the complete task. Provide the ID of the task as the URL parameter.

Request URL

The Update Task URL has the following format:

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

Note: Use the List Tasks API to get the ID of the task.

Make the following HTTP PUT request to the Update Task URL to update the complete task:

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

Make the following HTTP PATCH request to the Update Task URL to update some task fields:

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

Add the Content-Type header to specify the media type of the data you want to send with the request:

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

Request Body

Use the Read Task API to get the details of the task. Specify the task attributes when you update the task. Use the JSON format or the XML format to send the data to update in the request.

The following table describes the task parameters in the request body:

Parameter	Description
taskType	A set of actions that you can perform on a record. Use the name attribute to specify the task type. For more information about task types, see the *Multidomain MDM Data Director Implementation Guide*.
taskId	ID of the task.
owner	User who performs the task.
title	Short description for the task.
comments	Comments for the task.
attachments	Attachments for the task.
dueDate	Date when the owner must complete the task.
status	Status of the task in the workflow. Use one of the following two values: - Open: Task has not started or is in progress. - Closed: Task is completed or is cancelled.
priority	Level of importance of the task. Use one of the following values: high, normal, and low. Default is normal.

Parameter	Description
creator	User who creates the task.
createDate	Date when the task was created.
updatedBy	User who updates the task.
lastUpdateDate	Date when the task was last updated.
orsId	ID of the ORS as registered in the Databases tool in the Hub Console.
processId	ID of the workflow process that contains the task.
taskRecord	The root record or the cross-reference record associated with the task. Use the row ID or the source system and source key to specify the record.
businessEntity name	Name of the business entity to which the taskRecord belongs.

The following sample code uses the rowId to specify the taskRecord:

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

For a PATCH request, the request body contains those task fields that you want to change. You can change the task title, the priority, the due date, and the owner.

For a PUT request, the request body contains all task fields. Use the following request body for a PUT request:

```
{
     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: ''
         }
       }
     }]
}
```

Related Topics:

- "Formats for Dates and Time in UTC" on page 34

Sample API Request

The following sample PUT request updates a complete task:

```
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: ''
         }
       }
     }]
}
```

The following sample PATCH request updates some task fields:

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

{
     processId: "3719",
     priority: "HIGH",
     owner: "John"
}
```

Sample API Response

The API returns a 200 OK response code on successfully updating a task. The response body is empty.

Task Complete

The Task Complete REST API closes a task workflow after you complete all the tasks in the workflow. Use the API to close a workflow after you process all the task related records. For example, when you select merge candidates, you can create a task that initiates the merge workflow. The merge task is complete after you preview each candidate and either merge it or mark it as not a match. Use the API to close the merge workflow.

The API uses the PUT method.

Request URL

The Task Complete URL has the following format:

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

Make the following HTTP PUT request to the Task Complete URL:

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

Add the Content-Type header to specify the media type of the data you want to send with the request:

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

Request Body

Send the task details in the request body. Use the Read Task API to get the details of the task.

The following table describes the task parameters in the request body:

Parameter	Description
taskType	A set of actions that you can perform on a record. Use the name attribute to specify the task type. For more information about task types, see the *Multidomain MDM Data Director Implementation Guide*.
taskId	ID of the task.
owner	User who performs the task.
title	Short description for the task.
comments	Comments for the task.
dueDate	Date when the owner must complete the task.
status	Status of the task in the workflow. Use one of the following two values: - Open: Task has not started or is in progress. - Closed: Task is completed or is cancelled.
priority	Level of importance of the task.
creator	User who creates the task.
createDate	Date when the task was created.
updatedBy	User who updates the task.

Parameter	Description
lastUpdateDate	Date when the task was last updated.
orsId	ID of the ORS as registered in the Databases tool in the Hub Console.
processId	ID of the workflow process that contains the task.
taskRecord	The root record or the cross-reference record associated with the task. Use the row ID or the source system and source key to specify the record.
businessEntity name	Name of the business entity to which the taskRecord belongs.

The following sample code uses the rowId to specify the taskRecord:

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

Related Topics:

- "Formats for Dates and Time in UTC" on page 34

Sample API Request

The following sample request completes the merge workflow:

```
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": ""
         }
       }
     }]
}
```

Sample API Response

The API returns a 200 OK response on successfully completing a task workflow. The response body is empty.

Execute Task Action

The Execute Task Action REST API sets a task back to the workflow for further processing. Each task type has a set of task actions and a workflow that specifies the sequence of tasks. When you execute a task action, the task moves to the next step in the workflow. If a task action does not have a subsequent task, the workflow terminates when you execute the task action.

The API uses the POST method to perform actions, such as approve, escalate, or cancel a task.

Request URL

The following URL specifies the format of the Execute Task Action URL:

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

Note: Use the List Tasks API to get the ID of the task.

Make the following HTTP POST request to the Execute Task Action URL:

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

If you want to edit the task before executing the task action, add the Content-Type header to specify the media type of the request data:

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

Request Body

Provide the task data in the request body if you want to change the task details before you execute the task action.

The following table describes the parameters in the request body:

Parameter	Description
taskType	A set of actions that you can perform on a record. Use the name attribute to specify the task type. For more information about task types, see the *Multidomain MDM Data Director Implementation Guide*.
taskId	ID of the task.
owner	User who performs the task.
title	Short description for the task.
comments	Comments for the task.
attachments	Attachments for the task.
dueDate	Date when the owner must complete the task.
status	Status of the task in the workflow. Use one of the following two values: - Open: Task has not started or is in progress. - Closed: Task is completed or is cancelled.

Parameter	Description
priority	Level of importance of the task. Use one of the following values: high, normal, and low.
creator	User who creates the task.
createDate	Date when the task was created.
updatedBy	User who updates the task.
lastUpdateDate	Date when the task was last updated.
businessEntity	Name of the business entity.
orsId	ID of the ORS as registered in the Databases tool in the Hub Console.
processId	ID of the workflow process that contains the task.
taskRecord	The root record or the cross-reference record associated with the task. Use the row ID or the source system and source key to specify the record.
businessEntity name	Name of the business entity to which the taskRecord belongs.

The following sample code uses the rowId to specify the taskRecord:

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

Related Topics:

- "Formats for Dates and Time in UTC" on page 34

Sample API Request

The following sample request cancels a task and ends the workflow:

```
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: ''
            }
          }
        }]
}
```

Sample API Response

The API returns a 200 OK response code on successfully executing a task action. The response body is empty.

List Assignable Users

The List Assignable Users REST API returns a list of users to whom you can assign the tasks that belong to a task type. Use the API to get the target users for a task.

The API uses the GET method.

Request URL

The List Assignable Users URL has the following format:

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

Make the following HTTP GET request to the List Assignable Users URL:

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

Query Parameters

The following table lists the required parameters in the URL:

Parameter	Description
taskType	A set of actions that you can perform on a record. The task types include update with approval, update with optional approval, merge, unmerge, review no approval, final review, and update rejected record.
businessEntity	Name of the business entity.

Sample API Request

The following sample request retrieves a list of assignable users:

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

Sample API Response

The following sample response shows the list of assignable users for the task type ReviewNoApprove:

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

Bulk Claim Tasks

The Bulk Claim Task REST API claims multiple tasks. Use the API to claim tasks for which you are a potential owner.

The API uses the POST method.

Note: You can specify a maximum of 100 tasks in a request.

Request URL

The Bulk Claim Tasks URL has the following format:

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

Make the following HTTP POST request to the Bulk Claim Tasks URL:

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

Request Body

Specify the tasks that you want to claim in the request body.

The following table describes the parameters in the request body:

Parameter	Description
taskId	ID of the task.
opType	Operation that you want to perform on the tasks. Specify `Claim` to claim the tasks.

Sample API Request

The following sample request claims multiple tasks:

```
POST http://localhost:8080/cmx/task/operations/mdmdb3-MDM_SAMPLE
{
    "tasks":
    [
        {
```

```
            "taskId": "urn:b4p2:2315"
        },
        {
            "taskId": "urn:b4p2:2312"
        },
        {
            "taskId": "urn:b4p2:2309"
        },
        {
            "taskId": "urn:b4p2:2306"
        },
        {
            "taskId": "urn:b4p2:2303"
        }
    ],
    "opType": "Claim"
}
```

Sample API Response

The following sample response shows a list of tasks that were successfully claimed:

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

Bulk Release Tasks

The Bulk Release Tasks REST API releases multiple tasks back to the pool of unassigned tasks.

The API uses the POST method.

Note: You can specify a maximum of 100 tasks in a request.

Request URL

The Bulk Release Tasks URL has the following format:

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

Make the following HTTP POST request to the Bulk Release Tasks URL:

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

Request Body

Specify the tasks that you want to release back to the pool in the request body.

The following table describes the parameters in the request body:

Parameter	Description
taskId	ID of the task.
comments	Optional. Comments for the tasks.
opType	Operation that you want to perform on the tasks. Specify `Release` to release the tasks back to the pool of available tasks.

Sample API Request

The following sample request releases multiple tasks and adds the same comment to each task:

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

The following sample request releases multiple tasks and adds a different comment to each task:

```
POST http://localhost:8080/cmx/task/operations/mdmdb3-MDM_SAMPLE
{
    "tasks":
    [
        {
            "taskId": "urn:b4p2:2318",
            "comments": "Jane is on vacation."
        },
        {
            "taskId": "urn:b4p2:2209"
            "comments": "Joe is on vacation."
        }
    ],
    "opType": "Release"
}
```

Sample API Response

The following sample response shows a list of tasks that were successfully released:

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

```
            "status": "Completed"
        }
    ]
}
```

Bulk Assign Tasks

The Bulk Assign Tasks REST API allows you to assign or reassign tasks to users. Use the API to assign or reassign tasks to any user that is a potential owner of the task.

To determine the potential owners for tasks, use the List Potential Owners for Tasks AP. For more information, see "List Potential Owners for Tasks" on page 110.

The API uses the POST method.

Note: You can specify a maximum of 100 tasks in a request.

Request URL

The Bulk Assign Tasks URL has the following format:

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

Make the following HTTP POST request to the Bulk Assign Tasks URL:

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

Request Body

Specify the tasks and the users that you want to assign to the tasks in the request body.

The following table describes the parameters in the request body:

Parameter	Description
taskId	ID of the task.
comments	Optional. Comments for the tasks.
assignTo	User responsible for the task. **Note:** You can only assign tasks to users who are potential owners for the task. To determine the potential owners for tasks, use the List Potential Owners for Tasks AP. For more information, see "List Potential Owners for Tasks" on page 110.
opType	Operation that you want to perform on the tasks. Specify `Assign` to assign the tasks.

Sample API Request

The following sample request assigns a user to multiple tasks and adds the same comment to each task:

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

```
    "assignTo": "srmngr1",
    "comments": "Please take a look at these tasks instead.",
    "opType": "Assign"
}
```

The following sample request assigns the tasks to different users and adds a different comment to each task:

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

Sample API Response

The following sample response shows a list of tasks that were successfully assigned:

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

Bulk Edit Tasks

The Bulk Edit Tasks REST API allows to you edit the task details of multiple tasks.

The API uses the POST method to update some task fields.

Note: You can specify a maximum of 100 tasks in a request.

Request URL

The Bulk Edit Tasks URL has the following format:

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

Make the following HTTP POST request to the Bulk Edit Tasks URL:

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

Request Body

Specify the tasks that you want to edit and provide the task data in the request body.

The following table describes the parameters in the request body:

Parameter	Description
taskId	ID of the task.
dueDate	Optional. Date when the owner must complete the task.
comments	Optional. Comments for the task.
title	Optional. Short description for the task.
priority	Optional. Level of importance of the task. Use one of the following values: `high`, `normal`, and `low`.
assignTo	Optional. User responsible for the task.
fileId	Optional. The ID of the file that you want to attach to the task. To attach a file to a task, you must upload the file to a temporary storage. For more information, see Appendix B, "Using REST APIs to Upload Files" on page 330.
opType	Operation that you want to perform on the tasks. Specify `Edit` to edit the tasks.

Sample API Request

The following sample request edits multiple tasks by using the same task details:

```
POST http://localhost:8080/cmx/task/operations/mdmdb3-MDM_SAMPLE
{
    "tasks":
    [
        {
            "taskId": "urn:b4p2:4383"
        },
        {
            "taskId": "urn:b4p2:4384"
        }
    ],
    "comments": "Please work on this task ASAP. There's more information in the
attachment.",
    "priority": "High",
    "title": "Test Demo Review Person",
    "dueDate": "2019-05-01 00:00:00",
    "fileId": "TEMP_SVR1.1IOPK",
    "opType": "Edit"
}
```

The following sample request edits multiple tasks by using different task details:

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

```
            {
                "taskId": "urn:b4p2:4384",
                "priority": "High"
            }
        ],
        "opType": "Edit"
    }
```

Sample API Response

The following sample response shows a list of tasks that were successfully edited:

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

Bulk Task Action

The Bulk Task Action REST API executes task actions on multiple tasks. When you execute a task action, the task moves to the next step in the workflow. If a task action does not have a subsequent task, the workflow ends when you execute the task action.

The API uses the POST method to perform actions, such as approve, escalate, or reject a task.

To determine the task actions that you can perform on a task, use the Get Task Actions API. For more information, see "Get Task Actions" on page 108.

Note: You can specify a maximum of 100 tasks in a request.

Request URL

The Bulk Task Action URL has the following format:

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

Make the following HTTP POST request to the Bulk Task Action URL:

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

Request Body

Specify the tasks that you want to execute a task action on and the actions you want to execute.

The following table describes the parameters in the request body:

Parameter	Description
taskId	ID of the task.
actionName	Task action that you want to execute. Use the Get Task Actions API to determine the task actions that you can perform on the tasks.
opType	Operation that you want to perform on the tasks. Specify `Action` to execute a task action.

Sample API Request

The following sample request approves tasks:

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

The following sample request approves one task and escalates one task:

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

Sample API Response

The following sample response shows a list of tasks that were successfully edited:

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

```
            "status": "Completed"
        }
    ]
}
```

Get Task Actions

The Get Task Actions REST API retrieves a list of task actions that you can perform on tasks.

The API uses the POST method to retrieve a list of task actions.

Note: You can specify a maximum of 100 tasks in a request.

Request URL

The Get Task Actions URL has the following format:

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

Make the following HTTP POST request to the Get Task Actions URL:

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

Request Body

Specify the tasks for which you want to know the task actions that you can perform.

The following table describes the parameters in the request body:

Parameter	Description
taskId	ID of the task.
opType	Operation that you want to perform on the tasks. Specify `GetTaskActions` to retrieve a list of task actions that you can perform on the task.

Sample API Request

The following sample request retrieves the task actions you can perform on the tasks:

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

Sample API Response

The following sample response shows a list of tasks and the task actions that you can perform on each task:

```
{
    "tasks":
    [
        {
```

```
            "taskId": "urn:b4p2:5351",
            "status": "Completed",
            "taskActions":
            [
                {
                    "name": "Approve",
                    "displayName": "Approve",
                    "description": null,
                    "manualReassign": false,
                    "closeTaskView": true,
                    "cancelTask": false,
                    "nextTaskType": "AVOSBeNotification"
                },
                {
                    "name": "Reject",
                    "displayName": "Reject",
                    "description": null,
                    "manualReassign": false,
                    "closeTaskView": true,
                    "cancelTask": false,
                    "nextTaskType": "AVOSBeUpdate"
                },
                {
                    "name": "Disclaim",
                    "displayName": "Disclaim",
                    "description": null,
                    "manualReassign": true,
                    "closeTaskView": true,
                    "cancelTask": false,
                    "nextTaskType": "AVOSBeFinalReview"
                }
            ]
        },
        {
            "taskId": "urn:b4p2:5352",
            "status": "Completed",
            "taskActions":
            [
                {
                    "name": "Approve",
                    "displayName": "Approve",
                    "description": null,
                    "manualReassign": false,
                    "closeTaskView": true,
                    "cancelTask": false,
                    "nextTaskType": "AVOSBeNotification"
                },
                {
                    "name": "Reject",
                    "displayName": "Reject",
                    "description": null,
                    "manualReassign": false,
                    "closeTaskView": true,
                    "cancelTask": false,
                    "nextTaskType": "AVOSBeUpdate"
                },
                {
                    "name": "Disclaim",
                    "displayName": "Disclaim",
                    "description": null,
                    "manualReassign": true,
                    "closeTaskView": true,
                    "cancelTask": false,
                    "nextTaskType": "AVOSBeFinalReview"
                }
            ]
        }
    ]
}
```

List Potential Owners for Tasks

The List Potential Owners for Tasks REST API retrieves the potential owners for multiple tasks.

The API uses the GET method.

Note: You can specify a maximum of 100 tasks in a request.

Request URL

The List Potential Task Owners for Tasks URL has the following format:

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

Make the following HTTP GET request to the List Potential Task Owners for Tasks URL:

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

Request Body

Specify the tasks for which you want to retrieve the potential owners.

The following table describes the parameters in the request body:

Parameter	Description
taskId	ID of the task.
opType	Operation that you want to perform on the tasks. Specify `PotentialOwners` to get a list of potential owners for the tasks.

Sample API Request

The following sample response retrieves the potential owners for multiple tasks:

```
GET http://localhost:8080/cmx/task/operations/mdmdb3-MDM_SAMPLE
{
    "tasks":
    [
        {
            "taskId": "urn:b4p2:4383"
        },
        {
            "taskId": "urn:b4p2:4382"
        }
    ],
    "opType": "PotentialOwners"
}
```

Sample API Response

The following sample response shows the potential owners for each task:

```
{
    "tasks":
    [
        {
            "taskId": "urn:b4p2:4383",
            "status": "Completed",
            "users":
            [

                [
```

```
                    "srmngr2",
                    "srmngr 2"
                ],

                [
                    "srmngr1",
                    "senior manager"
                ]
            ]
        },
        {
            "taskId": "urn:b4p2:4382",
            "status": "Completed",
            "users":
            [

                [
                    "srmngr2",
                    "srmngr 2"
                ],

                [
                    "srmngr1",
                    "senior manager"
                ]
            ]
        }
    ]
}
```

List Potential Owners for a Task

The List Potential Owners for a Task REST API retrieves the potential owners for a task.

The API uses the GET method.

Request URL

The List Potential Task Owners for a Task URL has the following format:

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

Make the following HTTP GET request to the List Potential Task Owners for a Task URL:

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

Request Body

Specify the task for which you want to retrieve the potential owners.

The following table describes the parameters in the request body:

Parameter	Description
taskId	ID of the task.
opType	Operation that you want to perform on the task. Specify `PotentialOwners` to get a list of potential owners for the task.

Sample API Request

The following sample response retrieves the potential owners for a task:

```
GET http://localhost:8080/cmx/task/operations/list/owners/mdmdb3-MDM_SAMPLE
{
    "tasks":
    [
        {
            "taskId": "urn:b4p2:4383"
        },
        {
            "taskId": "urn:b4p2:4382"
        }
    ],
    "opType": "PotentialOwners"
}
```

Sample API Response

The following sample response shows the potential owners for a task:

```
{
    "tasks":
    [
        {
            "taskId": "urn:b4p2:4383",
            "status": "Completed",
            "users":
            [

                [
                    "srmngr2",
                    "srmngr 2"
                ],

                [
                    "srmngr1",
                    "senior manager"
                ]
            ]
        },
        {
            "taskId": "urn:b4p2:4382",
            "status": "Completed",
            "users":
            [

                [
                    "srmngr2",
                    "srmngr 2"
                ],

                [
                    "srmngr1",
                    "senior manager"
                ]
            ]
        }
    ]
}
```

List File Metadata

The List File Metadata REST API returns a list of file metadata in a storage.

Use the List File Metadata REST API with a BPM or TEMP storage.

The API uses the GET method.

Request URL

The List File Metadata URL has the following format:

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

Make the following HTTP GET request to the List File Metadata URL:

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

Sample API Request

The following sample request retrieves a list of file metadata in a TEMP storage:

```
GET http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/TEMP
```

Sample API Response

The following sample response shows a list of file metadata:

```
{
    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",
        },
        ...
    ]
}
```

Create File Metadata

The Create File Metadata REST API creates metadata for a file and returns a file ID for the file.

You can use the file ID to upload, attach, update, download, and delete the file.

Use the Create File Metadata REST API with a DB or TEMP storage.

The API uses the POST method.

Request URL

The Create File Metadata URL has the following format:

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

Make the following HTTP POST request to the Create File Metadata URL:

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

Request Body

Specify the metadata for the file.

The following table describes the parameters for the file metadata in the request body:

Parameter	Description
fileName	Name of the file. For example, `file.txt`.
fileType	Category of the file type. For example, `text` or `image`.
fileContentType	Content type of the file. The content type consists of a type and a subtype that are separated by a `/`. For example, `image/png`.

Note: The parameters required for the Create File Metadata REST API request are storage-specific.

Sample API Request

The following sample request creates metadata for a file in a TEMP storage:

```
POST http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/TEMP

{
    "fileName": "file1.txt",
    "fileType": "text",
    "fileContentType": "text/plain"
}
```

Sample API Response

The following example shows the response on successfully creating metadata for a file in a TEMP storage. The API returns the file ID for the file.

```
TEMP_SVR1.1VDVS
```

Note: The format of the file ID is <storage type>_<uniqueID>.

Get File Metadata

The Get File Metadata REST API returns the metadata for a file associated with a file ID.

Use the Get File Metadata REST API with a BPM, BUNDLE, DB, or TEMP storage.

The API uses the GET method.

Request URL

The Get File Metadata URL has the following format:

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

Make the following HTTP GET request to the Get File Metadata URL:

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

Sample API Request

The following sample request returns the metadata for a file with the file ID, `TEMP_SVR1.1VDVS`, in the TEMP storage:

```
GET http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/TEMP/TEMP_SVR1.1VDVS
```

The following sample request returns the metadata for the resource bundle file with the file ID, `besMetadata`, in the BUNDLE storage:

```
GET http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/BUNDLE/besMetadata
```

Sample API Response

The following sample response shows the metadata for a file with the file ID, `TEMP_SVR1.1VDVS`, in a TEMP storage:

```
{
    "fileName”: "file1.txt",
    "fileType”: "text",
    "fileContentType”: "text/plain"
}
```

The following sample response shows the metadata for the resource bundle file, `besMetadata`, in a BUNDLE storage:

```
{
    "fileName": "besMetadata.zip",
    "fileType": "BES Metadata Bundle",
    "fileContentType": "application/zip",
    "digest": "a08c5d97da7e6a780ed7c427ff14a8d2d396438cd65b654ad67424e226f64a41"
}
```

Update File Metadata

The Update File Metadata REST API updates the metadata for a file associated with a file ID.

Use the Update File Metadata REST API with a DB or TEMP storage.

The API uses the PUT method.

Request URL

The Update File Metadata URL has the following format:

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

Make the following HTTP PUT request to the Update File Metadata URL:

```
PUT
http://<host>:<port>/<context>/<database ID>/<storage>/<fileId>
```

Sample API Request

The following sample request updates the metadata for a file with the file ID, `TEMP_SVR1.1VDVS`, in a TEMP storage:

```
PUT http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/TEMP/TEMP_SVR1.1VDVS

{
    “fileName”: “file2.txt”,
    “fileType”: “text”,
    “fileContentType”: “text/plain”
}
```

The following sample request updates the metadata for a file with the file ID, `DB_SVR1.OJU1`, in a DB storage:

```
PUT http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/DB/DB_SVR1.OJU1
{
    "fileName": "Document_2.pdf",
    "fileType": "pdf",
    "fileContentType": "application/pdf"
}
```

Sample API Response

The API returns a 200 OK response code on successfully updating the file metadata. The response body is empty.

Upload File Content

The Upload File Content REST API uploads the content for a file associated to a file ID.

Use the Upload File Content REST API with a BUNDLE, DB, or TEMP storage.

The API uses the PUT method.

Request URL

The Upload File Content URL has the following format:

```
http://<host>:<port>/<context>/<database ID>/<storage>/<fileId>/content
```

Make the following HTTP PUT request to the Upload File Content URL:

```
PUT http://<host>:<port>/<context>/<database ID>/<storage>/<fileId>/content
```

Sample API Request

The following sample request uploads the content for a file with the file ID, `TEMP_SVR1.1VDVS`, in a TEMP storage:

```
PUT http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/TEMP/TEMP_SVR1.1VDVS/content

Test attachment content: file 1
```

The following sample request uploads the content for a file with the file ID, `DB_SVR1.OJU1`, in a DB storage:

```
PUT http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/DB/DB_SVR1.OJU1/content
Content-Type: application/octet-stream
<file object (upload using REST client)>
```

The following sample request uploads the content for a resource bundle file with the file ID, `besMetadata`, in a BUNDLE storage:

```
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
```

Sample API Response

The API returns a 200 OK response code on successfully uploading the content for a file. The response body is empty.

Get File Content

The Get File Content REST API returns the content for a file associated with a file ID.

Use the Get File Content REST API with a BPM, BUNDLE, DB, or TEMP storage.

The API uses the GET method.

Request URL

The Get File Content URL has the following format:

```
http://<host>:<port>/<context>/<database ID>/<storage>/<fileId>/content
```

Make the following HTTP GET request to the Get File Content URL:

```
GET http://<host>:<port>/<context>/<database ID>/<storage>/<fileId>/content
```

Sample API Request

The following sample request returns the content for a file with the file ID, `urn:b4p2:22203::file1.txt`, in a BPM storage:

```
GET http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/BPM/
urn:b4p2:22203::file1.txt/content
```

Note: Use the Read Task REST API to retrieve the file ID of a task attachment in a BPM storage.

The following sample request returns the content for a file with the file ID, `DB_SVR1.OJU1`, in a DB storage:

```
GET http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/DB/DB_SVR1.OJU1/content
```

Note: Use the Read Record REST API to retrieve the file ID of the file that you attach to a record.

The following sample request returns the content for the resource bundle file with the file ID, `besMetadata`, in a BUNDLE storage:

```
GET http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/BUNDLE/besMetadata/content
```

Sample API Response

The following sample response shows the content for a TXT file in a BPM storage:

```
Test attachment content: file 1
```

The following sample response shows the content for the resource bundle file in a BUNDLE storage:

```
Content-Disposition → attachment; filename=besMetadata.zip
Content-Type → application/octet-stream
Transfer-Encoding → chunked
```

Delete File

The Delete File REST API deletes the file associated with a file ID, which includes the file metadata and content.

Use the Delete File REST API with a BUNDLE, DB, or TEMP storage.

The API uses the DELETE method.

Request URL

The Delete File URL has the following format:

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

Make the following HTTP DELETE request to the Delete File URL:

```
DELETE http://<host>:<port>/<context>/<database ID>/<storage>/<fileId>
```

Sample API Request

The following sample request deletes the file associated with the file ID, `TEMP_SVR1.1VDVS`, in a TEMP storage, including the file metadata and content:

```
DELETE http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/TEMP/TEMP_SVR1.1VDVS
```

The following sample request deletes the file associated with the file ID, `DB_SVR1.OJU1`, in a DB storage, including the file metadata and content:

```
DELETE http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/DB/DB_SVR1.OJU1
```

The following sample request deletes the resource bundle file with the file ID, `besMetadata`, in a BUNDLE storage, including the file metadata and content:

```
DELETE http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/BUNDLE/besMetadata
```

Sample API Response

The API returns a 200 OK response code on successfully deleting a file. The response body is empty.

Preview Replaced Records

The Preview Replaced Records REST API returns business entity records that are edited and replaced with new values.

The API uses the POST method.

Request URL

The Preview Replaced Records URL has the following format:

```
http://<host>:<port>/find-replace/<orsId>/<entity>/<preview{?
firstRecord,order,recordsToReturn,returnTotal}>
```

Make the following HTTP POST request to the Preview Replaced Records URL:

```
POST http://<host>:<port>/find-replace/<orsId>/<entity>/<preview{?
firstRecord,order,recordsToReturn,returnTotal}>
```

Parameters

Specify the parameters to return business entity records that are edited and replaced with new values.

The following table describes the parameters in the request body.

Parameter	Type	Description
entity	Path	The name of the business entity records to return.
orsId	Path	Operational Reference Store ID of the database to return the replaced business entity records.
firstRecord	Query	Optional. The row of the records list.

Parameter	Type	Description
order	Query	Optional. Comma-separated list of field names with an optional prefix of + or –. The prefix + indicates to sort the results in ascending order, and the prefix – indicates to sort the results in descending order. Default is +. When you specify more than one parameter, the result set is ordered by the parameter that is first in the list, followed by the next.
recordsToReturn	Query	Optional. Specifies the number of rows to return.
returnTotal	Query	Optional. If set to `true`, returns the number of records in the result. Default is `false`

Sample API Request

The following sample request returns business entity records that are edited and replaced with new values:

```
POST /cmx/find-replace/localhost-orcl-MDM_SAMPLE/Person/preview?returnTotal=true HTTP/1.1
Content-Type: application/json
Host: localhost:8080
Content-Length: 234

{
  "scope" : {
    "type" : "query",
    "filter" : "lastName='jones'"
  },
  "findReplace" : {
    "field" : "Emails.electAddrTypeCd",
    "find" : null,
    "replacement" : {
      "electronicType" : "EMAIL"
    }
  }
}
```

Sample API Response

The following sample response shows the replaced business entity records:

```
HTTP/1.1 200 OK
Server: JBoss-EAP/7
Date: Mon, 13 Jan 2020 18:07:02 GMT
X-Powered-By: Undertow/1
Content-Type: application/json;charset=UTF-8
Content-Length: 4891

{
  "link" : [ ],
  "firstRecord" : 1,
  "recordCount" : 6,
  "pageSize" : 10,
  "foundRecords" : 2,
  "item" : [ {
    "origin" : {
      "key" : {
        "rowid" : "401924        "
      },
      "name" : "Person",
      "label" : "jones, alice",
      "object" : {
        "key" : {
          "rowid" : "426           "
```

```
      },
      "name" : "Emails",
      "label" : "Emails"
    }
  },
  "found" : false,
  "record" : {
    "rowidObject" : "426           ",
    "creator" : "admin",
    "createDate" : "2020-01-13T12:40:17.811-05:00",
    "updatedBy" : "admin",
    "lastUpdateDate" : "2020-01-13T12:40:18.583-05:00",
    "consolidationInd" : 4,
    "lastRowidSystem" : "SYS0          ",
    "hubStateInd" : 1,
    "label" : "Emails",
    "electronicAddress" : "alice@mail.com",
    "electAddrTypeCd" : {
      "electronicType" : "EMAIL"
    }
  }
}, {
  "origin" : {
    "key" : {
      "rowid" : "401925        "
    },
    "name" : "Person",
    "label" : "jones, bob",
    "object" : {
      "key" : {
        "rowid" : "427           "
      },
      "name" : "Emails",
      "label" : "Emails"
    }
  },
  "found" : false,
  "record" : {
    "rowidObject" : "427           ",
    "creator" : "admin",
    "createDate" : "2020-01-13T12:40:18.006-05:00",
    "updatedBy" : "admin",
    "lastUpdateDate" : "2020-01-13T12:40:18.655-05:00",
    "consolidationInd" : 4,
    "lastRowidSystem" : "SYS0          ",
    "hubStateInd" : 1,
    "label" : "Emails",
    "electronicAddress" : "bob@mail.com",
    "electAddrTypeCd" : {
      "electronicType" : "EMAIL"
    }
  }
}, {
  "origin" : {
    "key" : {
      "rowid" : "401926        "
    },
    "name" : "Person",
    "label" : "jones, charlie",
    "object" : {
      "key" : {
        "rowid" : "428           "
      },
      "name" : "Emails",
      "label" : "Emails"
    }
  },
  "found" : false,
  "record" : {
    "rowidObject" : "428           ",
    "creator" : "admin",
```

```
    "createDate" : "2020-01-13T12:40:18.216-05:00",
    "updatedBy" : "admin",
    "lastUpdateDate" : "2020-01-13T12:40:18.216-05:00",
    "consolidationInd" : 4,
    "lastRowidSystem" : "SYS0          ",
    "hubStateInd" : 1,
    "label" : "Emails",
    "electronicAddress" : "charlie@mail.com",
    "electAddrTypeCd" : {
      "electronicType" : "EMAIL"
    }
  }
}, {
  "origin" : {
    "key" : {
      "rowid" : "401935        "
    },
    "name" : "Person",
    "label" : "jones, alice",
    "object" : {
      "key" : {
        "rowid" : "437           "
      },
      "name" : "Emails",
      "label" : "Emails"
    }
  },
  "found" : true,
  "record" : {
    "rowidObject" : "437           ",
    "creator" : "admin",
    "createDate" : "2020-01-13T13:07:02.187-05:00",
    "updatedBy" : "admin",
    "lastUpdateDate" : "2020-01-13T13:07:02.187-05:00",
    "consolidationInd" : 4,
    "lastRowidSystem" : "SYS0          ",
    "hubStateInd" : 1,
    "label" : "Emails",
    "electronicAddress" : "alice@mail.com"
  }
}, {
  "origin" : {
    "key" : {
      "rowid" : "401936        "
    },
    "name" : "Person",
    "label" : "jones, bob",
    "object" : {
      "key" : {
        "rowid" : "438           "
      },
      "name" : "Emails",
      "label" : "Emails"
    }
  },
  "found" : true,
  "record" : {
    "rowidObject" : "438           ",
    "creator" : "admin",
    "createDate" : "2020-01-13T13:07:02.389-05:00",
    "updatedBy" : "admin",
    "lastUpdateDate" : "2020-01-13T13:07:02.390-05:00",
    "consolidationInd" : 4,
    "lastRowidSystem" : "SYS0          ",
    "hubStateInd" : 1,
    "label" : "Emails",
    "electronicAddress" : "bob@mail.com"
  }
}, {
  "origin" : {
    "key" : {
```

```
        "rowid" : "401937        "
      },
      "name" : "Person",
      "label" : "jones, charlie",
      "object" : {
        "key" : {
          "rowid" : "439           "
        },
        "name" : "Emails",
        "label" : "Emails"
      }
    },
    "found" : false,
    "record" : {
      "rowidObject" : "439           ",
      "creator" : "admin",
      "createDate" : "2020-01-13T13:07:02.587-05:00",
      "updatedBy" : "admin",
      "lastUpdateDate" : "2020-01-13T13:07:02.587-05:00",
      "consolidationInd" : 4,
      "lastRowidSystem" : "SYS0          ",
      "hubStateInd" : 1,
      "label" : "Emails",
      "electronicAddress" : "charlie@mail.com",
      "electAddrTypeCd" : {
        "electronicType" : "EMAIL"
      }
    }
  } ]
}
```


Update Find and Replace

The Update Replaced Records REST API returns filtered records for the business entities you want to edit and replace with new values.

The API uses the POST method.

Request URL

The Update Replaced Records URL has the following format:

```
http://<host>:<port>/find-replace/<orsId>/<entity>/{?systemName}
```

Make the following HTTP POST request to the Update Replaced Records URL:

```
POST http://<host>:<port>/find-replace/<orsId>/<entity>/{?systemName}
```

Parameters

Specify the parameters to update filtered records for the business entities.

The following table describes the parameters:

Parameter	Type	Description
entity	Path	The name of the business entity records to return.
orsId	Path	Operational Reference Store ID of the database to return the replaced business entity records.

Parameter	Type	Description
systemName	Query	Optional. Name of the source system.
If-Unmodified-Since	Header	Optional. Enables to skip records that were previously replaced.

Sample API Request

The following sample request updates filtered records for the business entities you want to edit and replace with new values:

```
POST /cmx/find-replace/localhost-orcl-MDM_SAMPLE/Person?systemName=Admin HTTP/1.1
Content-Type: application/json
Host: localhost:8080
Content-Length: 234

{
  "scope" : {
    "type" : "query",
    "filter" : "lastName='jones'"
  },
  "findReplace" : {
    "field" : "Emails.electAddrTypeCd",
    "find" : null,
    "replacement" : {
      "electronicType" : "EMAIL"
    }
  }
}
```

Sample API Response

The following sample response shows the updated filtered records for business entities:

```
HTTP/1.1 200 OK
Server: JBoss-EAP/7
X-Powered-By: Undertow/1
Content-Type: application/json;charset=UTF-8
Content-Length: 1339

{
  "successful" : 2,
  "failed" : 0,
  "skipped" : 0,
  "item" : [ {
    "object" : {
      "Person" : {
        "key" : {
          "rowid" : "401935        "
        },
        "rowidObject" : "401935        ",
        "Emails" : {
          "link" : [ ],
          "item" : [ {
            "key" : {
              "rowid" : "437",
              "sourceKey" : "480000041000"
            },
            "rowidObject" : "437",
            "electAddrTypeCd" : {
              "key" : {
                "rowid" : "1             "
              },
              "rowidObject" : "1             "
```

```
            }
          } ]
        }
      }
    },
    "parameters" : { }
  }, {
    "object" : {
      "Person" : {
        "key" : {
          "rowid" : "401936        "
        },
        "rowidObject" : "401936        ",
        "Emails" : {
          "link" : [ ],
          "item" : [ {
            "key" : {
              "rowid" : "438",
              "sourceKey" : "480000043000"
            },
            "rowidObject" : "438",
            "electAddrTypeCd" : {
              "key" : {
                "rowid" : "1             "
              },
              "rowidObject" : "1             "
            }
          } ]
        }
      }
    },
    "parameters" : { }
  } ]
}
```


Import New File

The Import New File REST API starts a new file import job and imports business entities from a file.

The API uses the POST method.

Request URL

The Import New File URL has the following format:

```
http://<host>:<port>/<context>/<orsId>/<job>
```

Make the following HTTP POST request to the Import New File URL:

```
POST http://<host>:<port>/<context>/<orsId>/<job>
```

Parameters

Specify the following parameters for a new file import in the request body.

The following table describes the parameters:

Parameter	Type	Description
orsID	Path	Operational Reference Store database ID to import the new file into.
fileID	Query	Optional. File ID of the imported file. Use this option if you are updating existing records.
mappingID	Query	Optional. Mapping ID of the imported file. Use this option if you are updating a previously imported file.
systemName	Query	Optional. Name of the system user who imports the file.

Sample API Request

The following sample request starts a new file import job and imports business entities:

```
POST /cmx/beimport/localhost-orcl-MDM_SAMPLE/job HTTP/1.1
Content-Type: application/json
Host: localhost:8080
Content-Length: 94

{
  "fileId" : "TEMP_SVR1.1G7UW",
  "mappingId" : "SVR1.1G7UX",
  "systemName" : "Admin"
}
```

Sample API Response

The following sample response shows the new file successfully imported:

```
HTTP/1.1 201 Created
Server: JBoss-EAP/7
X-Powered-By: Undertow/1
Location: http://localhost:8080/cmx/jobcontrol/localhost-orcl-MDM_SAMPLE/group/SVR1.1G7UY
```

Import Matched File

The Import Matched File REST API matches new business entities with existing business entities before importing the file. After the import operation is completed, the matched business entities are updated with new data.

You can import duplicate business entities, unique business entities or duplicate and unique business entities.

The API uses the POST method.

Request URL

The Import Matched File URL has the following format:

```
http://<host>:<port>/<context>/<orsId>/<aftermatch>
```

Make the following HTTP POST request to the Import Matched File URL:

```
POST http://<host>:<port>/<context>/<orsId>/<aftermatch>
```

Parameters

Specify the parameters to match new business entities with existing business before importing the file.

The following table describes the parameters:

Parameter	Type	Description
orsId	Path	Operational Reference Store database ID to import the file after matching existing records.
filter	Query	Optional. Use the following filter values: withMatches: Enter `true` if the records match existing records. withoutMatches: Enter `false` if the records do not match existing records.
jobGroupControlId	Query	Optional. Match job group ID.
systemName	Query	Optional. Source system name.

Sample API Request

The following sample request matches new business entities with existing business entities before importing the file:

```
POST /cmx/beimport/{ors}/aftermatch
{
    jobGroupControlId: "...",
    systemName: "SFA",
    filter: {
      withMatches: true,
      withoutMatches: false
    }
}
```

Sample API Response

The API returns a 200 OK response code after successfully matching new business entities with existing business entities and importing the file. The response body is empty.

Get File Properties

The Get File Properties REST API returns the file properties of the imported file. For example, The API returns delimiter, text qualifier and decimal separator file properties if you selected these properties when you import the file.

The API uses the GET method.

Note: The API returns existing file properties with delimiters only if there are five rows or more.

Request URL

The Get File Properties URL has the following format:

```
http://<host>:<port>/<context>/<orsId>/<fileId>/<properties{query}>
```

Make the following HTTP GET request to the Get File Properties URL:

```
GET http://<host>:<port>/<context>/<orsId>/<fileId>/<properties{query}>
```

Parameters

Specify the parameters to get the file properties of the imported file.

The following table describes the parameters:

Parameter	Type	Description
orsId	Path	Operational Reference Store database ID.
fileId	Path	File ID of the imported file.
suggest	Query	Optional. If set to true, the request returns the file properties based on the imported file. Else, the request returns suggested file properties based on the data in the imported file.

Sample API Request

The following sample request detects and returns the file properties of the imported file:

```
GET /cmx/file/parser/localhost-orcl-MDM_SAMPLE/TEMP_SVR1.1G7UW/properties?suggest=true
HTTP/1.1
Content-Type: application/json
Host: localhost:8080
```

Sample API Response

The following sample response shows the header and body after returning the file properties of the imported file:

```
HTTP/1.1 200 OK
Server: JBoss-EAP/7
X-Powered-By: Undertow/1
Content-Type: application/json;charset=UTF-8
Content-Length: 370

{
  "confidence" : "HIGH",
  "type" : "CSV",
  "properties" : {
    "headerRow" : 1,
    "dataStartsAtRow" : 2,
    "regionalSettings" : {
      "locale" : "en-CA",
      "datePattern" : "MM-dd-yyyy",
      "decimalSeparator" : ".",
      "thousandsSeparator" : ","
    },
    "delimiter" : ",",
    "encoding" : "UTF-8",
    "textQualifier" : "\""
  }
}
```

Save File Properties

The Save File Properties REST API parses and saves the imported file properties.

The API uses the PUT method.

Request URL

The Save File Properties URL has the following format:

```
http://<host>:<port>/<context>/<orsId>/<fileId>/<properties>
```

Make the following HTTP PUT request to the Save File Properties URL:

```
PUT http://<host>:<port>/<context>/<orsId>/<fileId>/<properties>
```

Parameters

Specify the parameters to save the file properties of the imported file.

The following table describes the parameters in the request body:

Parameter	Type	Description
orsId	Path	Operational Reference Store database ID.
fileId	Path	File ID of the imported file.
headerRow	Query	Optional. Row header number you want a column.
dataStartAtRow	Query	Optional. Row of the source file you want to import data from.
locale	Query	Optional. Localization language.
dataPattern	Query	Optional. Date format for the date fields in the import file.
decimalSeparator	Query	Optional. Decimal separator. Enter a comma or a period.
thousandsSeparator	Query	Optional. Thousands separator. Enter a comma or a period.
delimiter	Query	Optional. Character that represents the break between data values in the import file. Enter a predefined delimiter or define a custom delimiter.
encoding	Query	Optional. Unicode encoding standard.
textQualifier	Query	Optional. Symbols used in the file to enclose a string.

Sample API Request

The following sample request parses and saves the file properties:

```
PUT /cmx/file/parser/localhost-orcl-MDM_SAMPLE/TEMP_SVR1.1G7UW/properties HTTP/1.1
Content-Type: application/json
Host: localhost:8080
Content-Length: 344

{
  "type" : "CSV",
  "properties" : {
    "headerRow" : 1,
    "dataStartsAtRow" : 2,
    "regionalSettings" : {
      "locale" : "en-CA",
      "datePattern" : "MM-dd-yyyy",
      "decimalSeparator" : ".",
      "thousandsSeparator" : ","
    },
    "delimiter" : ",",
```

```
    "encoding" : "UTF-8",
    "textQualifier" : "\""
  }
}
```

Sample API Response

The following sample response shows the file properties are saved.

```
HTTP/1.1 200 OK
Server: JBoss-EAP/7
X-Powered-By: Undertow/1
```

Return File Properties

The Return File Properties REST API returns the properties of files that are persisted.

The API uses the GET method.

Request URL

The Return File Properties URL has the following format:

```
http://<host>:<port>/<context>/<orsId>/<fileId>/<properties>
```

Make the following HTTP GET request to the Return File Properties URL:

```
GET http://<host>:<port>/<context>/<orsId>/<fileId>/<properties>
```

Parameters

Specify the parameters to return the properties of files that are persisted.

The following table describes the parameters:

Parameter	Type	Description
orsId	Path	Operational Reference Store ID to retrieve the file properties from.
fileId	Path	File ID of the imported file.
suggest	Query	Optional. If set to true, the request returns properties of files that are persisted. Else, the request returns all file properties.

Sample API Request

The following sample request returns the properties of files that are persisted.

```
GET /cmx/file/parser/localhost-orcl-MDM_SAMPLE/TEMP_SVR1.1G7UW/properties?suggest=true
HTTP/1.1
Content-Type: application/json
Host: localhost:8080
```

Sample API Response

The following sample response shows the properties of files that are persisted:

```
HTTP/1.1 200 OK
Server: JBoss-EAP/7
```

```
X-Powered-By: Undertow/1
Content-Type: application/json;charset=UTF-8
Content-Length: 370

{
  "confidence" : "HIGH",
  "type" : "CSV",
  "properties" : {
    "headerRow" : 1,
    "dataStartsAtRow" : 2,
    "regionalSettings" : {
      "locale" : "en-CA",
      "datePattern" : "MM-dd-yyyy",
      "decimalSeparator" : ".",
      "thousandsSeparator" : ","
    },
    "delimiter" : ",",
    "encoding" : "UTF-8",
    "textQualifier" : "\""
  }
}
```

Preview Parsed Files

The Preview Parsed Files REST API parses file properties and returns flat records based on the properties provided in the request body. The API parses CSV and Excel files.

The API uses the POST method.

Request URL

The Preview Parsed Files API URL has the following format:

```
http://<host>:<port>/<context>/<orsId>/<fileId>/preview{?pageSize}
```

Make the following HTTP POST request to the Preview Parsed Files API URL:

```
POST http://<host>:<port>/<context>/<orsId>/<fileId>/preview{?pageSize}
```

Parameters

Specify the parameters to parse file properties and returns flat records based on the properties provided in the request body.

The following table describes the parameters in the request body:

Parameter	Type	Description
orsId	Path	Operational Reference Store database ID of the file stored in the database.
fileId	Path	File ID of the imported file.
pagesize	Query	Optional. Number of record rows to return.

Sample API Request

The following sample request parses file properties and returns flat records based on the properties provided in the request body:

```
POST /cmx/file/parser/localhost-orcl-MDM_SAMPLE/TEMP_SVR1.1G7UW/preview HTTP/1.1
Content-Type: application/json
```

```
Host: localhost:8080
Content-Length: 345

{
  "type" : "CSV",
  "properties" : {
    "headerRow" : 1,
    "dataStartsAtRow" : 2,
    "regionalSettings" : {
      "locale" : "en-CA",
      "datePattern" : "d MMM, yyyy",
      "decimalSeparator" : ".",
      "thousandsSeparator" : ","
    },
    "delimiter" : ",",
    "encoding" : "UTF-8",
    "textQualifier" : "\""
  }
}
```

Sample API Response

The following sample response shows the flat records retrieved based on the properties provided in the request body:

```
HTTP/1.1 200 OK
Server: JBoss-EAP/7
X-Powered-By: Undertow/1
Content-Type: application/json;charset=UTF-8
Content-Length: 381

{
  "header" : [ "first_name", "last_name", "job_title", "gender", "birth_date" ],
  "data" : [ [ "alice", "smith", "developer", "F", "01-01-1950" ], [ "bob", "smith",
"tester", "M", "02-02-1960" ], [ "charlie", "smith", "manager", "X", "03-03-1970" ],
[ "dave", "smith", "accountant", "M", "ten years ago" ], [ "eve", "smith", "", "F",
"04-04-1990" ] ],
  "numberOfRows" : 5
}
```

Get Parsed Files

The Get Parsed Files REST API parses file properties and returns flat records based on the properties. The API parses CSV and Excel files.

The API uses the GET method.

Request URL

The Get Parsed File Properties URL has the following format:

```
http://<host>:<port>/<context>/<orsId>/<fileId>/preview{?pageSize}
```

Make the following HTTP GET request to the Get Parsed File Properties URL:

```
GET http://<host>:<port>/<context>/<orsId>/<fileId>/preview{?pageSize}
```

Parameters

Specify the parameters to parse file properties and return flat records based on the file properties.

The following table describes the parameters:

Parameter	Type	Description
orsId	Path	Operational Reference Store database ID.
fileId	Path	File ID of the imported file.
pagesize	Query	Optional. Number of record rows to return.

Sample API Request

The following sample request parses files and returns flat records based on the properties associated with a file:

```
GET /cmx/file/parser/localhost-orcl-MDM_SAMPLE/TEMP_SVR1.1G7UW/preview HTTP/1.1
Content-Type: application/json
Host: localhost:8080
```

Sample API Response

The following sample response shows the flat records parsed based on the properties associated with a file:

```
HTTP/1.1 200 OK
Server: JBoss-EAP/7
X-Powered-By: Undertow/1
Content-Type: application/json;charset=UTF-8
Content-Length: 381

{
  "header" : [ "first_name", "last_name", "job_title", "gender", "birth_date" ],
  "data" : [ [ "alice", "smith", "developer", "F", "01-01-1950" ], [ "bob", "smith",
"tester", "M", "02-02-1960" ], [ "charlie", "smith", "manager", "X", "03-03-1970" ],
[ "dave", "smith", "accountant", "M", "ten years ago" ], [ "eve", "smith", "", "F",
"04-04-1990" ] ],
  "numberOfRows" : 5
}
```

Import File Parsing Errors

The Import File Parsing Errors REST API returns parsing errors from the file import operation in a CSV file format.

The API uses the GET method.

Request URL

The Import File Parsing Errors URL has the following format:

```
http://<host>:<port>/<context>/<orsId>/<parse-errors>/{jobGroupControlId}{?entity}
```

Make the following HTTP GET request to the Import File Parsing Errors URL:

```
GET http://<host>:<port>/<context>/<orsId>/<parse-errors>/{jobGroupControlId}{?entity}
```

Parameters

Specify the parameters to return parsing errors from the file import operation in a CSV file.

The following table describes the parameters in the request body:

Parameter	Type	Description
orsId	Path	Operational Reference Store database ID to return parsing errors from the file import operation.
jobGroupControlId	Query	Optional. Batch job group ID.
entity	Query	Optional. Name of the business entity.

Sample API Request

The following sample request returns parsing errors from the file import operation:

```
GET /cmx/beimport/localhost-orcl-MDM_SAMPLE/parse-errors/SVR1.1G7UY HTTP/1.1
Host: localhost:8080
```

Sample API Response

The following sample response shows parsing errors from the file import operation:

```
HTTP/1.1 200 OK
Server: JBoss-EAP/7
X-Powered-By: Undertow/1
Content-Type: text/csv
Content-Length: 200

first_name,last_name,job_title,gender,birth_date,ROW,COLUMN,ERROR,ENTITY
dave,smith,accountant,M,ten years ago,3,birth_date,Failed to convert value [ten years
ago] for field [birthdate].,PersonView
```

Import File Loading Errors

The Import File Loading Errors REST API returns errors from the file import operation in a CSV file format.

The API uses the GET method.

Request URL

The Import File Loading Errors URL has the following format:

```
http://<host>:<port>/<context>/<orsId>/<load-errors>/{jobGroupControlId}{?entity}
```

Make the following HTTP GET request to the Import File Loading Errors URL:

```
GET http://<host>:<port>/<context>/<orsId>/<load-errors>/{jobGroupControlId}{?entity}
```

Parameters

Specify the parameters to return loading errors from the file import operation in a CSV file.

The following table describes the parameters in the request body .

Parameter	Type	Description
orsId	Path	Operational Reference Store database ID to return loading errors from the file import operation.
jobGroupControlId	Query	Optional. Batch job group ID.
entity	Query	Optional. Name of the business entity.

Sample API Request

The following sample request returns loading errors from the file import operation:

```
GET /cmx/beimport/localhost-orcl-MDM_SAMPLE/load-errors/SVR1.1G7UY HTTP/1.1
Host: localhost:8080
```

Sample API Response

The following sample response shows loading errors from the file import operation:

```
HTTP/1.1 200 OK
Server: JBoss-EAP/7
X-Powered-By: Undertow/1
Content-Type: text/csv
Content-Length: 257

first_name,last_name,job_title,gender,birth_date,ROW,COLUMN,ERROR,ENTITY
charlie,smith,manager,X,03-03-1970,2,,SIP-50112: Could not run WriteCO business entity
service.  A record for [genderCd] could not be found in business entity
[LUGender].,PersonView
```

Preview Promote

The Preview Promote REST API returns a preview of a resulting record if you promote the pending changes.

The API uses the GET method. You can see how a record would look if you apply the pending changes to the record. The API response contains the record with the new values and a change summary with the old values. The API does not return information about data that you delete. Provide the interaction ID of the pending changes in the URL.

Request URL

The Preview Promote URL has the following format:

```
http://<host>:<port>/<context>/<database ID><business entity>/<rowId>?
action=previewPromote&interactionID=<interaction ID>
```

Make the following HTTP GET request to the Preview Promote URL:

```
GET http://<host>:<port>/<context>/<database ID><business entity>/<rowId>?
action=previewPromote&interactionID=<interaction ID>
```

Query Parameters

The interaction ID of the pending changes is a required parameter in the URL.

The following table lists the query parameters:

Parameter	Description
contentMetadata	Metadata for the merge preview. Provide a comma-separated list. You can use the following values: - BVT. Specify the rowid of the record that contains the most trustworthy value to use in the merge preview. Returns information about the cross-reference record and the original record ID. - MERGE. Specify the rowids of the records to merge. Returns information about how the descendant records were merged.
interactionId	Interaction ID of the pending changes.
effectiveDate	Optional. Date for which you want to preview the changes. Use the parameter for timeline-enabled base objects.

Related Topics:

- "Formats for Dates and Time in UTC" on page 34

Sample API Request

The following sample request creates a preview of a root record in the Person business entity:

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/233?
action=previewPromote&interactionId=72300000001000
```

Sample API Response

The following sample response returns a preview of the record with the new values and a change summary of old values:

```
{
    "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"
```

```
    }
}
```

Promote

The Promote REST API promotes all pending changes made to a record based on the interaction ID of the change request.

The API uses the POST method. Provide the interaction ID as a query parameter.

Request URL

The Promote URL has the following format:

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<rowId of the root
record>?action=promote&interactionId=<interaction ID>
```

Make the following HTTP POST request to the Promote URL:

```
POST http://<host>:<port>/<context>/<database ID>/<business entity>/<rowId of the root
record>?action=promote&interactionId=<interaction ID>
```

Query Parameter

The interaction ID of the pending changes is a required parameter. The API uses the interaction ID to find all the records related to the pending changes.

Sample API Request

The following sample request promotes all the pending changes based on the interaction ID of the change request:

```
POST http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/1038246?
action=promote&interactionId=69120000294000
```

Sample API Response

The following sample response contains the row ID of the record after promoting the pending changes:

```
{
            Person: {
                        rowidObject: "1038246"
                    }
}
```

Delete Pending

The Delete Pending REST API deletes all pending changes you make to a record based on the interaction ID of the change request.

The API uses the DELETE method and returns the row ID of the record.

Request URL

The Delete Pending URL has the following format:

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<rowid>?
action=deletePending&interactionId=<interaction ID>
```

Make the following DELETE request to the Delete Pending URL:

```
DELETE http://<host>:<port>/<context>/<database ID>/<business entity>/<rowid>?
action=deletePending&interactionId=<interaction ID>
```

Query Parameter

Provide the interaction ID of the pending changes that you want to delete.

Sample API Request

The following sample request deletes all the pending changes based on the interaction ID of the change request:

```
DELETE http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/233?
action=deletePending&interactionId=72300000001000
```

Sample API Response

The following sample response contains the row ID of the record after deleting the pending changes:

```
{
    Person: {
        rowidObject: "233"
    }
}
```

Preview Merge

The Preview Merge REST API returns a preview of a consolidated root record if you merge two or more root records.

The API uses the POST method and accepts a list of root records and field-level overrides to return a preview of the merged record. The row ID of the target record is a required parameter.

Request URL

The Preview Merge URL has the following format:

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<rowid of the record>?
action=previewMerge
```

The following Preview Merge URL format specifies the number of child levels to return:

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<rowid of the record>?
action=previewMerge&depth=2
```

Make the following HTTP POST request to the Preview Merge URL:

```
POST http://<host>:<port>/<context>/<database ID>/<business entity>/<rowid of the
record>?action=previewMerge
```

Note: In the request body, add the keys property and specify the root records that you want to merge with the target record.

To override matches for the child records, add the contentMetadata parameter:

```
POST http://<host>:<port>/<context>/<database ID>/<business entity>/<rowid of the
record>?action=previewMerge&contentMetadata=MERGE/<json/xml>
```

Note: In the request body, add the overrides property and specify the merge overrides.

To specify the media type of the data you want to send with the request, add the Content-Type header:

```
POST http://<host>:<port>/<context>/<database ID>/<business entity>/<rowid of the
record>?action=previewMerge
Content-Type: application/<json/xml>
```

Query Parameters

The row ID of the target record is a required parameter.

You can use the following query parameters:

Parameter	Description
contentMetadata	Metadata for the merge preview. Provide a comma-separated list. You can use the following values: - BVT. Returns information about the winning cross-reference record and the original record ID. - MERGE. Returns information about how the descendant records were merged.
depth	Number of child levels to return.
effectiveDate	Date for which you want to generate the preview.

Related Topics:

- "Formats for Dates and Time in UTC" on page 34

Request Body

Before you begin, use the Read Matched Records API to determine which matched records you can merge with the original root record. Send the list of records in the request body for the Preview Merge API.

You can override field values in the root record. For example, if none of the matched root records contain the correct spelling of the first name, you can specify the correct first name in the request body. Also, you can remove matched records or specify other matching records.

Use the following properties in the request body:

Properties / Elements	Type	Description
keys	array	Required. An ordered list of the matched root records that you want to merge. You can identify the records either by row ID or by a combination of the source system and the source key.
overrides	object	Overrides the field values in a root record and the matches for child records.
MERGE	object	Overrides the field values in child records that you want to merge. Add the type of child record within the `overrides` object and then add the `MERGE` object.

The following JSON code sample identifies a root record to merge with the target root record:

```
{
  keys: [
    {
       rowid: "P2"
    }
  ]
}
```

The following code shows how to override a field in the Party root record and how to override the merge candidates for Telephone child records:

```
{
  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
                ]
              }
            }
          }
        ]
      }
    }
  }
}
```

Sample API Request

The following sample request returns the preview of a consolidated record:

```
POST http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/2478245?
action=previewMerge
```

```
{
  keys: [
    {
        rowid: "2478246"
    },
   {
        rowid: "2478230"
    }
  ],
  overrides: {
    Person: {
       firstName: "Charlie"
    }
  }
}
```

Sample API Response

The following sample response shows the preview of the consolidated record:

```
{
    "Person": {
        "rowidObject": "2478245       ",
        "partyType": "Person",
        "lastName": "Smith",
        "firstName": "Charlie",
        "displayName": "ALICE SMITH"
    }
}
```

Update Pending Merge

To save changes to records that are part of a pending merge task, such as record value overrides, use the Update Pending Merge REST API. The API saves the changes based on the interaction ID of the records.

The API uses the POST method.

Request URL

The path component of the request URL must include the row ID of the target record.

The Update Pending Merge URL has the following format:

```
http://<host>:<port>/<context>/<database ID>/<business entity>
/<rowid of the root record>?
action=updatePendingMerge&interactionId=<interaction ID>
```

Make the following HTTP POST request to the Update Pending Merge URL:

```
POST http://<host>:<port>/<context>/<database ID>/<business entity>
/<rowid of the root record>?
action=updatePendingMerge&interactionId=<interaction ID>
```

In the request body, add the keys and specify the root records that you want to merge with the target root record. You can also specify the child records for which you want to override matches.

Query Parameter

The following table describes the query parameters that you can use in the URL:

Parameter	Description
action	Required. Saves changes, such as field-level overrides, to a pending merge task. Set to `updatePendingMerge`, and use the parameter with the `interactionId` parameter. For example, use the following query to save changes to the Person business entity records that are pending the merge action: `Person?action=updatePendingMerge&interactionId`
interactionId	Required. Interaction ID of the pending merge task.

Request Body

Before you use the Update Pending Merge API, use the Read Matched Records REST API to determine the matched records that you can merge with the target root record. Send the list of records in the request body for the Update Pending Merge API.

You can override field values in the root record. For example, if none of the matched root records contain the correct spelling of the first name, you can specify the correct first name in the request body. Also, you can remove matched records or specify other matching records.

Use the following properties in the request body:

Properties / Elements	Type	Description
keys	array	Required. An ordered list of the matched root records that you want to merge. You can identify the records either by row ID or by a combination of the source system and the source key.
overrides	object	Overrides the field values in a root record and the matches for child records.
MERGE	object	Overrides the field values in child records that you want to merge. Add the type of child record within the `overrides` object and then add the `MERGE` object.

The following JSON code sample identifies two root records to merge with the target root record:

```
{
  keys: [
    {rowid: "2478246"},
    {rowid: "2478230"}
  ]
}
```

The following sample request body shows how to override a field in the Party root record and how to override the matched records for Telephone child records:

```
{
  keys: [
    {
      rowid: "2478246"
    }
  ]
  overrides: {
    Party: {
      rowidObject: "2478230",
      firstName: "Charlie",   //Override the value for the first name
      Telephone: {   // Specifies the Telephone child records to merge
        item:[
          {
            rowidObject: "2511",
            MERGE: {
              item: [   // To remove the original merge candidates, specify null
                null,
                null
              ],
              $original: {
                item: [
                  {key:{rowid: "2822"}},
                  {key:{rowid: "2733"}}
                ]
              }
            }
          },
          {
            rowidObject: "2644",
            MERGE: {
```

```
              item: [   // To add or change merge candidates, specify matched records
                {key:{rowid: "2822"}}
              ],
            $original: {
              item: [
                null
              ]
            }
          }
        ]
      }
    }
  }
}
```

Sample API Request

The following sample request overrides the first name field of the target record with the value Charlie:

```
POST http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/2478245?
action=updatePendingMerge&interactionId=3982462873645
```

```
{
  keys: [
    {
        rowid: "2478246"
    },
   {
        rowid: "2478230"
    }
  ],
  overrides: {
    Person: {
       firstName: "Charlie"
    }
  }
}
```

Sample API Response

The following sample response contains the row ID of the target record for which the merge action is pending:

```
{
    "Person": {
        "key": {
            "rowid": "2478245"
        },
        "rowidObject": "2478245"
    }
}
```

Pending Merge

The Pending Merge REST API updates all pending merge tasks you make to a record based on the interaction ID of the change request. Pending Merge allows you to defer merge operations until the workflow process grants approval for all merge tasks.

The API uses the POST method and returns the row ID of the record.

Request URL

The Pending Merge URL has the following format:

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<rowid>?
action=PendingMerge&interactionId=<interaction ID>
```

Make the following HTTP POST request to the Pending Merge URL:

```
POST http://<host>:<port>/<context>/<database ID>/<business entity>/<rowid of the
record>?action=Merge
```

Query Parameter

The interaction ID of the pending merge is a required parameter.

Sample API Request

The following sample request updates all pending merge tasks associated with the interaction ID:

```
POST /Person/123?action=pendingMerge&interactionId=123
```

Sample API Response

The following sample response contains the row IDs of the affected root base objects:

```
{
    keys: [{rowid: "456"}, {rowid: "789"}],
    overrides: {...}
}
```

PromoteMerge

The Promote Merge REST API runs all pending merge tasks associated with the interaction ID of the change request.

The API uses the POST method and returns the row ID of the winning record.

Request URL

The Promote Merge URL has the following format:

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<rowid>?
action=PromoteMerge&interactionId=<interaction ID>
```

Make the following HTTP POST request to the Promote Merge URL:

```
POST http://<host>:<port>/<context>/<database ID>/<business entity>/<rowid of the
record>?action=Merge
```

Query Parameter

The interaction ID of the pending merge task is a required parameter. The API uses the interaction ID to find all pending merge tasks and runs the merges.

Sample API Request

The following sample request promotes all pending merge tasks associated with the interaction ID:

```
POST /Person/123?action=promoteMerge&interactionId=123
```

Sample API Response

The following sample response contains the row IDs of the records after promoting the pending merge tasks:

```
POST /Person/123?action=promoteMerge&interactionId=123
```

File Transformation

The File Transformation API retrieves the list of business entities from the imported file before mapping the business entities to the target fields.

The API uses the POST method.

Request URL

The Files Transformation API has the following format:

```
http://<host>:<port>/cmx/flat2be/{orsId}/preview{?entity,fileId,pageSize}
```

Make the following HTTP POST request to the Files Transformation API URL:

```
POST /cmx/flat2be/{orsId}/preview{?entity,fileId,pageSize}
```

Parameters

Specify the properties to preview the list of business entities from the imported file before mapping the business entities to the target fields.

The following table describes the parameters.

Parameter	Description
fileId	File ID that will be used to generate the preview.
orsId	Operational reference store ID of the database to preview the mapping from.
entity	Optional. Number of business entities or relationship to preview.
pageSize	Optional. Number of pages to return.

Sample API Request

The following sample request retrieves the list of mapped business entities from the imported file before mapping the business entities:

```
POST /cmx/flat2be/localhost-orcl-MDM_SAMPLE/preview?entity=Person&fileId=TEMP_SVR1.1G7UW
HTTP/1.1
Content-Type: application/json
Host: localhost:8080
Content-Length: 431

{
  "regionalSettings" : {
    "locale" : "en-CA",
    "datePattern" : "d MMM, yyyy",
    "decimalSeparator" : ".",
    "thousandsSeparator" : ","
  },
  "objects" : [ {
    "name" : "Person",
    "fields" : [ {
      "name" : "firstName",
```

```
      "fileColumn" : "first_name",
      "skipNulls" : false
    }, {
      "name" : "lastName",
      "fileColumn" : "last_name",
      "skipNulls" : false
    } ]
  } ]
}
```

Sample API Response

The following sample response shows the list of mapped business entities from the imported file before mapping the business entities to the target fields.

```
HTTP/1.1 200 OK
Server: JBoss-EAP/7
X-Powered-By: Undertow/1
Content-Type: application/json
Content-Length: 825

{
  "link" : [ ],
  "item" : [ {
    "object" : {
      "Person" : {
        "label" : "smith, alice",
        "lastName" : "smith",
        "firstName" : "alice"
      }
    }
  }, {
    "object" : {
      "Person" : {
        "label" : "smith, bob",
        "lastName" : "smith",
        "firstName" : "bob"
      }
    }
  }, {
    "object" : {
      "Person" : {
        "label" : "smith, charlie",
        "lastName" : "smith",
        "firstName" : "charlie"
      }
    }
  }, {
    "object" : {
      "Person" : {
        "label" : "smith, dave",
        "lastName" : "smith",
        "firstName" : "dave"
      }
    }
  }, {
    "object" : {
      "Person" : {
        "label" : "smith, eve",
        "lastName" : "smith",
        "firstName" : "eve"
      }
    }
  } ]
}
```

Suggest Mapping

The Suggest Mapping REST API suggests mapping for the business entities or relationships in the imported file.

The API uses the GET method.

Request URL

The Suggest Mapping API has the following format:

```
http://<host>:<port>/<context>/<orsId>/<suggest{?entity,fileId,purpose}>
```

Make the following HTTP GET request to the Suggest Mapping API URL:

```
GET http://<host>:<port>/<context>/<orsId>/<suggest{?entity,fileId,purpose}>
```

Request Body

Specify the parameters to suggest mapping for the business entities or relationships in the imported file.

The following table describes the parameters in the request body.

Parameter	Type	Description
orsId	Path	ID of the Operational Reference Store database to preview the mapping from.
fileId	Query	Optional. File ID of the imported file.
purpose	Query	Optional. The purpose of the mapping and how it will be used.
entity	Query	Optional. Number of business entities or relationships to preview. If you specify the entity parameter, the API detects and returns mapping for the specified entity only.

Sample API Request

The following sample request returns the suggested mapping for the business entities or relationships in the imported file:

```
GET /cmx/flat2be/localhost-orcl-MDM_SAMPLE/suggest?fileId=TEMP_SVR1.1G7UW HTTP/1.1
Host: localhost:8080
```

Sample API Response

The following sample response shows the suggested mapping for the business entities or relationships in the imported file:

```
HTTP/1.1 200 OK
Server: JBoss-EAP/7
X-Powered-By: Undertow/1
Content-Type: application/json;charset=UTF-8
Content-Length: 1315

{
  "name" : "my_mapping",
  "systemName" : "Admin",
  "mapping" : {
    "confidence" : "HIGH",
    "regionalSettings" : {
```

```
      "locale" : "en-CA",
      "datePattern" : "MM-dd-yyyy",
      "decimalSeparator" : ".",
      "thousandsSeparator" : ","
    },
    "objects" : [ {
      "confidence" : "HIGH",
      "name" : "PersonView",
      "fields" : [ {
        "confidence" : "HIGH",
        "name" : "firstName",
        "fileColumn" : "first_name",
        "skipNulls" : false
      }, {
        "confidence" : "HIGH",
        "name" : "jobTitle",
        "fileColumn" : "job_title",
        "skipNulls" : false
      }, {
        "confidence" : "HIGH",
        "name" : "lastName",
        "fileColumn" : "last_name",
        "skipNulls" : false
      }, {
        "confidence" : "HIGH",
        "name" : "birthdate",
        "fileColumn" : "birth_date",
        "skipNulls" : false
      }, {
        "confidence" : "HIGH",
        "name" : "genderCd",
        "fileColumn" : "gender",
        "skipNulls" : false
      } ],
      "children" : [ ]
    } ],
    "purpose" : "IMPORT"
  },
  "creator" : "admin",
  "createDate" : "2020-01-13T12:39:37.186-05:00",
  "updatedBy" : "admin",
  "lastUpdateDate" : "2020-01-13T12:39:37.186-05:00",
  "system" : false
}
```


Merge Records

The Merge Records REST API merges two or more root records to create a single consolidated record. The row ID of the consolidated record is the row ID of the record to which you merge the other records.

The API uses the POST method. You can specify field-level overrides for the merged record in the request body.

Request URL

The Merge Records URL has the following format:

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<rowid of the record>?
action=merge
```

Make the following HTTP POST request to the Merge Records URL:

```
POST http://<host>:<port>/<context>/<database ID>/<business entity>/<rowid of the
record>?action=merge
```

Add the Content-Type header to specify the media type of the data you want to send with the request:

```
POST http://<host>:<port>/<context>/<database ID>/<business entity>/<rowid of the
record>?action=merge
Content-Type: application/<json/xml>
```

Query Parameters

The following table lists the parameters that you can use in the URL:

Parameter	Description
taskComment	Add a comment to the workflow task triggered by the API.
taskAttachments	If task attachments are enabled, attach a file to the workflow task triggered by the API.

Request Body

Before you begin, use the Preview Merge API to preview the results of merging the selected root records. When you are happy with the preview, use the same properties in the request body for the Merge Records API.

You can override field values in the root record. For example, if none of the matched root records contain the correct spelling of the first name, you can specify the correct first name in the request body. Also, you can remove matched records or specify other matching records.

Use the following properties in the request body:

Properties / Elements	Type	Description
keys	array	Required. An ordered list of the matched root records that you want to merge. You can identify the records either by row ID or by a combination of the source system and the source key.
overrides	object	Overrides the field values in a root record and the matches for child records.
MERGE	object	Overrides the field values in child records that you want to merge. Add the type of child record within the `overrides` object and then add the `MERGE` object.

The following JSON code sample identifies two root records to merge with the target root record:

```
{
  keys: [
    {rowid: "2478246"},
    {rowid: "2478269"}
  ]
}
```

For an example of how to use the `overrides` and `MERGE` properties with the Merge Records API, see the request body for the Merge Preview API.

Sample API Request

The following sample request merges records to form a consolidated record. The request adds a comment and attachment to the workflow task triggered by the API.

```
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"
    }
  }
}
```

Sample API Response

The following sample response shows the consolidated record:

```
{
    "Person": {
        "link": [
            {
                "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/
2478245",
                "rel": "self"
            }
        ],
        "key": {
            "rowid": "2478245"
        },
        "rowidObject": "2478245"
    }
}
```

Unmerge Records

The Unmerge Records REST API unmerges a root record into the individual root records that existed before you merged the records.

The API uses the POST method.

Request URL

The Unmerge Records URL has the following format:

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

Make the following HTTP POST request to the Unmerge Records URL:

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

Add the Content-Type header to specify the media type of the data you want to send with the request:

```
POST http://<host>:<port>/<context>/<database ID>/<business entity>/<rowId>?
action=unmerge
Content-Type: application/<json/xml>
```

Query Parameters

The following table lists the parameters that you can use in the URL:

Parameter	Description
taskComment	Add a comment to the workflow task triggered by the API.
taskAttachments	If task attachments are enabled, attach a file to the workflow task triggered by the API.

Request Body

Send the list of the records that you want to unmerge from the consolidated record in the request body. Use the xref row ID or the source system and the source key to specify the records.

Use the Read Record API to get the xref rowId of the record to unmerge. The following sample request retrieves the XREF metadata of a record:

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/2638243?
contentMetadata=XREF
```

Sample API Request

The following sample request unmerges a record from the consolidated record. The request adds a comment and attachment to the workflow task triggered by the API.

```
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"
}
```

Sample API Response

The following sample response shows the record that you unmerge from the consolidated record:

```
{
    "Person": {
        "link": [
            {
                "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/
2478249",
                "rel": "self"
            }
        ],
        "key": {
            "rowid": "2478249       "
        },
        "rowidObject": "2478249       "
    }
}
```

Create Mapping

The Create Mapping REST API creates a new mapping when you imported a new file into the database.

The API uses the POST method.

Request URL

The Create Mapping URL has the following format:

```
http://<host>:<port>/<context>/<orsId>/<mapping>
```

Make the following HTTP POST request to the Create Mapping URL:

```
POST //<host>:<port>/<context>/<orsId>/<mapping>
```

Parameters

Specify the parameters to create a new mapping in the request body. Use JSON or XML format to create a mapping in the request.

The following table describes the parameters in the request body:

Parameter	Type	Description
orsID	Path	Operational Reference Store database ID to create a new mapping.
mapping	Query	Optional. Enter the properties of the mapping.
confidence	Query	Optional. Confidence level of the mapping. Use any of the following values: High, Medium or Low.
creator	Query	Optional. Your user name.
createDate	Query	Optional. Date when you create the mapping.
UpdatedBy	Query	Optional. If you are updating an existing mapping, enter your user name.
lastUpdateDate	Query	Optional. Date when the user updates the mapping.
modified	Query	Optional. Use the following values: true if you are modifying an existing mapping or false if you are creating a new mapping.
name	Query	Optional. Name of the mapping.
score	Query	Optional. Match score of the mapping.
system	Query	Optional. Use the following values: True: If the system creates the mapping. False: If a user manually creates the mapping.
systemName	Query	Optional. Name of the system that creates the mapping.
objects	Query	Optional. Enter the fields you want for the business entity or relationship base object.

Sample API Request

The following sample request creates a new mapping:

```
POST /cmx/flat2be/localhost-orcl-MDM_SAMPLE/mapping HTTP/1.1
Content-Type: application/json
Host: localhost:8080
Content-Length: 737
```

```
{
  "mapping" : {
    "regionalSettings" : {
      "locale" : "en-CA",
      "datePattern" : "dd-MM-yyyy",
      "decimalSeparator" : ".",
      "thousandsSeparator" : ","
    },
    "objects" : [ {
      "name" : "PersonView",
      "fields" : [ {
        "name" : "firstName",
        "fileColumn" : "first_name",
        "skipNulls" : false
      }, {
        "name" : "jobTitle",
        "fileColumn" : "job_title",
        "skipNulls" : false
      }, {
        "name" : "lastName",
        "fileColumn" : "last_name",
        "skipNulls" : false
      }, {
        "name" : "birthdate",
        "fileColumn" : "birth_date",
        "skipNulls" : false
      } ]
    } ],
    "purpose" : "IMPORT"
  }
}
```

Sample API Response

The following sample response shows the new mapping created in the database.

```
HTTP/1.1 201 Created
Server: JBoss-EAP/7
X-Powered-By: Undertow/1
Location: http://localhost:8080/cmx/flat2be/localhost-orcl-MDM_SAMPLE/mapping/SVR1.1G7UX
Content-Type: application/json;charset=UTF-8
Content-Length: 27

{
  "id" : "SVR1.1G7UX"
}
```

Preview Mapping

The Preview Mapping REST API retrieves the list of mapped business entities from the imported file.

The API uses the GET method.

Request URL

The Preview Mapping API has the following format:

```
http://<host>:<port>/<context>/<orsId>/<preview{?entity,fileId,mapping,pageSize}>
```

Make the following HTTP GET request to the Preview Mapping API URL:

```
GET http://<host>:<port>/<context>/<orsId>/<preview{?entity,fileId,mapping,pageSize}>
```

Parameters

Specify the parameters to preview the list of mapped business entities from the imported file.

The following table describes the parameters in the request body:

Parameter	Type	Description
orsId	Path	Operational reference store ID of the database to preview the mapping from.
fileId	Query	File ID. If specified, the API retrieves the mapped business entities from the specified file ID.
mappingId	Query	Optional. Enter the mapping ID of the imported file.
entity	Query	Optional. Enter the number of business entities or relationships to preview.
pageSize	Query	Optional. Enter the number of pages or the records to return.

Sample API Request

The following sample request provides a preview of the mapped business entities from the imported file:

```
GET /cmx/flat2be/localhost-orcl-MDM_SAMPLE/preview?
mapping=SVR1.1G7UX&entity=PersonView&fileId=TEMP_SVR1.1G7UW HTTP/1.1
Host: localhost:8080
```

Sample API Response

The following sample response shows a preview of the mapped business entities from the imported file:

```
HTTP/1.1 200 OK
Server: JBoss-EAP/7
X-Powered-By: Undertow/1
Content-Type: application/json
Content-Length: 1751

{
  "link" : [ ],
  "item" : [ {
    "object" : {
      "PersonView" : {
        "firstName" : "alice",
        "lastName" : "smith",
        "birthdate" : "1950-01-01T00:00:00-05:00",
        "genderCd" : {
          "genderCode" : "F",
          "genderDisp" : "F"
        },
        "jobTitle" : "developer"
      }
    }
  }, {
    "object" : {
      "PersonView" : {
        "firstName" : "bob",
        "lastName" : "smith",
        "birthdate" : "1960-02-02T00:00:00-05:00",
        "genderCd" : {
          "genderCode" : "M",
          "genderDisp" : "M"
```

```
        },
        "jobTitle" : "tester"
      }
    }
  }, {
    "object" : {
      "PersonView" : {
        "firstName" : "charlie",
        "lastName" : "smith",
        "birthdate" : "1970-03-03T00:00:00-05:00",
        "genderCd" : {
          "genderCode" : "X",
          "genderDisp" : "X"
        },
        "jobTitle" : "manager"
      }
    }
  }, {
    "object" : {
      "PersonView" : {
        "firstName" : "dave",
        "lastName" : "smith",
        "genderCd" : {
          "genderCode" : "M",
          "genderDisp" : "M"
        },
        "jobTitle" : "accountant"
      }
    },
    "errors" : [ {
      "code" : "SIP-55206",
      "message" : "Failed to convert value [ten years ago] for field [birthdate].",
      "row" : 3,
      "column" : "birth_date",
      "field" : "birthdate"
    } ]
  }, {
    "object" : {
      "PersonView" : {
        "firstName" : "eve",
        "lastName" : "smith",
        "birthdate" : "1990-04-04T00:00:00-04:00",
        "genderCd" : {
          "genderCode" : "F",
          "genderDisp" : "F"
        },
        "jobTitle" : ""
      }
    }
  } ]
}
```


Find Mapping

The Find Mapping REST API finds the existing user or system defined mapping by the mapping ID.

The API uses the GET method.

Request URL

The Find Mapping URL has the following format:

```
http://<host>:<port>/<context>/<orsId>/<mapping{?
createdBy,fileId,firstRecord,purpose,recordsToReturn,returnTotal}>
```

Make the following HTTP GET request to the Find Mapping URL:

```
GET http://<host>:<port>/<context>/<orsId>/<mapping {?
createdBy,fileId,firstRecord,purpose,recordsToReturn,returnTotal}>
```

Parameters

Specify the parameters to find the existing user or system defined mapping by the mapping ID.

The following table describes the parameters in the request body:

Parameter	Path	Description
orsId	Path	Operational Reference Store database ID to find the mapping.
fileId	Query	File ID. If specified, the API retrieves the mapping from the specified file ID.
createdBy	Query	Enter the user who created the mapping.
firstRecord	Query	Optional. Enter to return only the first row of the record.
purpose	Query	Optional. Use the following values: Import: If you are creating a new mapping. Match: If you are matching an existing mapping.
recordsToReturn	Query	Optional. Enter the number of records to return.
returnTotal	Query	Optional. Enter the total number of records.

Sample API Request

The following sample request finds the existing user or system defined mapping by the mapping ID:

```
GET http://<host>:<port>/cmx/flat2be/{orsId}/mapping{?
createdBy,fileId,firstRecord,purpose,recordsToReturn,returnTotal}
```

Sample API Response

The following sample response shows the existing user or system defined mapping by the mapping ID:

```
{
    mappings: [
        {
            id: 123, name: "....", ...
        },
        ...
    ]
}
```

Read Mapping

The Read Mapping REST API returns existing business entities or relationships mapping by the mapping ID.

The API uses the GET method.

Request URL

The Read Mapping URL has the following format:

```
http://<host>:<port>/<context>/<orsId>/<mapping>/<id><fileId>
```

Make the following HTTP GET request to the Read Mapping URL:

```
GET http://<host>:<port>/<context>/<orsId>/<mapping>/<id><fileId>
```

Parameters

Specify the parameters to return the mapped business entities or relationships from the imported file ID.

The following table describes the parameters in the request body.

Parameter	Type	Description
mappingId	Path	Name of the mapping given by the user.
orsId	Path	Operational Reference Store database ID to retrieve the mapping from.
fileId	Query	Optional. Specify the file ID associated with the existing mapping.

Sample API Request

The following sample request returns mapped business entities or relationships by the mapping ID:

```
GET /cmx/flat2be/localhost-orcl-MDM_SAMPLE/mapping/SVR1.1G7UX HTTP/1.1
Content-Type: application/json
Host: localhost:8080
```

Sample API Response

The following sample response returns existing business entities or relationships by the mapping ID:

```
HTTP/1.1 200 OK
Server: JBoss-EAP/7
Content-Disposition: inline;filename=f.txt
X-Powered-By: Undertow/1
Content-Type: application/json;charset=UTF-8
Content-Length: 993

{
  "id" : "SVR1.1G7UX    ",
  "systemName" : "Admin",
  "mapping" : {
    "regionalSettings" : {
      "locale" : "en-CA",
      "datePattern" : "dd-MM-yyyy",
      "decimalSeparator" : ".",
      "thousandsSeparator" : ","
    },
    "objects" : [ {
      "name" : "PersonView",
      "fields" : [ {
        "name" : "firstName",
        "fileColumn" : "first_name",
        "skipNulls" : false
      }, {
        "name" : "jobTitle",
        "fileColumn" : "job_title",
        "skipNulls" : false
```

```
      }, {
        "name" : "lastName",
        "fileColumn" : "last_name",
        "skipNulls" : false
      }, {
        "name" : "birthdate",
        "fileColumn" : "birth_date",
        "skipNulls" : false
      } ],
      "children" : [ ]
    } ],
    "purpose" : "IMPORT"
  },
  "creator" : "admin",
  "createDate" : "2020-01-13T13:06:26.153-05:00",
  "updatedBy" : "admin",
  "lastUpdateDate" : "2020-01-13T13:06:26.153-05:00",
  "system" : true
}
```

Update Mapping

The Update Mapping REST API updates the existing mapping with new data.

The API uses the PUT method.

Request URL

The Update Mapping URL has the following format:

```
http://<host>:<port>/<context>/<orsId>/<mapping>/<mappingId>
```

Make the following HTTP PUT request to the Update Mapping URL:

```
PUT http://<host>:<port>/<context>/<orsId>/<mapping>/<mappingId>
```

Parameters

Specify the parameters to update the user defined mapping in the request body. Use JSON or XML format to create a mapping in the request.

The following table describes the parameters in the request body.

Parameter	Type	Description
mappingID	Path	Mapping ID of the user data.
orsId	Path	Operational Reference Store database ID.
mapping	Query	Optional. Enter the properties of the mapping.
confidence	Query	Optional. Confidence level of the mapping. Use any of the following values: High, Medium or Low.
creator	Query	Optional. Your user name.
createDate	Query	Optional. Date when you create the mapping.
UpdatedBy	Query	Optional. If you are updating an existing mapping, enter your user name.

Parameter	Type	Description
lastUpdateDate	Query	Optional. Date when the user updates the mapping.
modified	Query	Optional. Use the following values: true if you are modifying an existing mapping or false if you are creating a new mapping.
name	Query	Optional. Name of the mapping.
score	Query	Optional. Match score of the mapping.
system	Query	Optional. Use the following values: True: If the system creates the mapping. False: If a user manually creates the mapping.
systemName	Query	Optional. Displays the name of the system that creates the mapping.
objects	Query	Optional. Enter the fields you want for the business entity or relationship base object.
purpose	Query	Optional. Use the following values: Import: If you are creating a new mapping. Match: If you are matching an existing mapping.

Sample API Request

The following sample request updates existing mapping with new data.

```
PUT /cmx/flat2be/localhost-orcl-MDM_SAMPLE/mapping/SVR1.1G7UX HTTP/1.1
Content-Type: application/json
Host: localhost:8080
Content-Length: 889

{
  "system" : false,
  "name" : "my_mapping",
  "mapping" : {
    "regionalSettings" : {
      "locale" : "en-CA",
      "datePattern" : "dd-MM-yyyy",
      "decimalSeparator" : ".",
      "thousandsSeparator" : ","
    },
    "objects" : [ {
      "name" : "PersonView",
      "fields" : [ {
        "name" : "firstName",
        "fileColumn" : "first_name",
        "skipNulls" : false
      }, {
        "name" : "jobTitle",
        "fileColumn" : "job_title",
        "skipNulls" : false
      }, {
        "name" : "lastName",
        "fileColumn" : "last_name",
        "skipNulls" : false
      }, {
        "name" : "birthdate",
        "fileColumn" : "birth_date",
        "skipNulls" : false
      }, {
```

```
        "name" : "genderCd",
        "fileColumn" : "gender",
        "skipNulls" : false
      } ]
    } ],
    "purpose" : "IMPORT"
  }
}
```

Sample API Response

The following is the sample response after successfully updating the existing mapping with new data:

```
HTTP/1.1 204 No Content
Server: JBoss-EAP/7
X-Powered-By: Undertow/1
```

File Transformation

The File Transformation API retrieves the list of business entities from the imported file before mapping the business entities to the target fields.

The API uses the POST method.

Request URL

The Files Transformation API has the following format:

```
http://<host>:<port>/cmx/flat2be/{orsId}/preview{?entity,fileId,pageSize}
```

Make the following HTTP POST request to the Files Transformation API URL:

```
POST /cmx/flat2be/{orsId}/preview{?entity,fileId,pageSize}
```

Parameters

Specify the properties to preview the list of business entities from the imported file before mapping the business entities to the target fields.

The following table describes the parameters.

Parameter	Description
fileId	File ID that will be used to generate the preview.
orsId	Operational reference store ID of the database to preview the mapping from.
entity	Optional. Number of business entities or relationship to preview.
pageSize	Optional. Number of pages to return.

Sample API Request

The following sample request retrieves the list of mapped business entities from the imported file before mapping the business entities:

```
POST /cmx/flat2be/localhost-orcl-MDM_SAMPLE/preview?entity=Person&fileId=TEMP_SVR1.1G7UW
HTTP/1.1
Content-Type: application/json
Host: localhost:8080
```

```
Content-Length: 431

{
  "regionalSettings" : {
    "locale" : "en-CA",
    "datePattern" : "d MMM, yyyy",
    "decimalSeparator" : ".",
    "thousandsSeparator" : ","
  },
  "objects" : [ {
    "name" : "Person",
    "fields" : [ {
      "name" : "firstName",
      "fileColumn" : "first_name",
      "skipNulls" : false
    }, {
      "name" : "lastName",
      "fileColumn" : "last_name",
      "skipNulls" : false
    } ]
  } ]
}
```

Sample API Response

The following sample response shows the list of mapped business entities from the imported file before mapping the business entities to the target fields.

```
HTTP/1.1 200 OK
Server: JBoss-EAP/7
X-Powered-By: Undertow/1
Content-Type: application/json
Content-Length: 825

{
  "link" : [ ],
  "item" : [ {
    "object" : {
      "Person" : {
        "label" : "smith, alice",
        "lastName" : "smith",
        "firstName" : "alice"
      }
    }
  }, {
    "object" : {
      "Person" : {
        "label" : "smith, bob",
        "lastName" : "smith",
        "firstName" : "bob"
      }
    }
  }, {
    "object" : {
      "Person" : {
        "label" : "smith, charlie",
        "lastName" : "smith",
        "firstName" : "charlie"
      }
    }
  }, {
    "object" : {
      "Person" : {
        "label" : "smith, dave",
        "lastName" : "smith",
        "firstName" : "dave"
      }
    }
  }, {
```

```
    "object" : {
      "Person" : {
        "label" : "smith, eve",
        "lastName" : "smith",
        "firstName" : "eve"
      }
    }
  } ]
}
```

Read a Relationship

The Read Relationship REST API returns the details of a relationship record, such as the party types, rowIDs, and display names of the two records.

The API uses the GET method.

Request URL

The Read Relationship URL has the following format:

```
http://<host>:<port>/<context>/<database ID>/<relationship>/<row ID of the relationship record>
```

Make the following HTTP GET request to the URL:

```
GET http://<host>:<port>/<context>/<database ID>/<relationship>/<row ID of the relationship record>
```

Query Parameters

The following table lists the query parameters:

Parameter	Description
suppressLinks	Optional. Indicates whether the parent-child links are visible in the API response. Set the parameter to true to suppress all the parent-child links in the response. Set the parameter to false to not display the links in the API response. Default is false.
depth	Optional. Number of child levels to return.

Sample API Request

The following sample request returns the details of the relationship record with the row ID 85, which is of the relationship type `ProductGroupProductGroupIsParentOfProductProducts`:

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/
ProductGroupProductGroupIsParentOfProductProducts/85
```

The following sample request returns the details with depth equal to 2:

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/
ProductGroupProductGroupIsParentOfProductProducts/85?depth=2
```

Sample API Response

The following example shows the details of the relationship record with the row ID 85, which is of the relationship type `ProductGroupProductGroupIsParentOfProductProducts`:

```
{
  "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"
    }
  }
}
```


Create a Relationship

The Create Relationship REST API creates a relationship between the specified records. To create a relationship between records, relationships must exist between the business entities to which the records belong. For example, if you want to specify a relationship between Informatica and John Smith, a relationship must exist between the Organization and the Person business entities. You must send the relationship data in the request body.

The API uses the PUT and the POST methods.

Request URL

The Create Relationship REST URL has the following format:

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

Note: The name of the source system is a required parameter in the URL.

Make the following HTTP POST or PUT request to the URL:

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

Add the Content-Type header to specify the media type of the data you want to send with the request:

```
Content-Type: application/<json/xml>
```

URL Parameters

The name of the source system is a required parameter in the request URL.

Request Body

Send data for the relationship record in the REST request body. Use the JSON format or the XML format to send data. Provide the required parameter values in the request body.

Sample API Request

The following sample request creates the `OrganizationEmploysPerson` relationship between an Organization business entity with row ID 101 and a Person business entity with row ID 1101:

```
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"
}
```

The `OrganizationEmploysPerson` relationship defines a relationship from an Organization business entity to a Person business entity. The `from` element specifies the record from which the relationship originates and the `to` element specifies the record at which the relationship ends.

Sample API Response

The following sample response shows the response header and body after successfully creating a relationship between an Organization business entity with row ID 101 and a Person business entity with row ID 1101:

```
{
        "OrganizationEmployesPerson": {
        "key": {
        "rowid": "414721"
        "sourceKey": "SVR1.1E7UW"
 }-

"rowidObject": "414721"

    "from": {
                    "key": {
                                "rowid": "101 "
                                                }-
        "rowidObject": "101 "
        }-

 "to": {
                "key": {
                            "rowid": "1101 "
        }-

"rowidObject": "1101 "
               }-
```

```
    }-

}
```

Update a Relationship

The Update Relationship REST API updates the relationship between two records. The API updates the additional attributes defined for the relationship.

The API uses the POST and the PUT methods.

Request URL

Update Relationship URL has the following format:

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

Note: The name of the source system is a required parameter in the URL.

Make the following HTTP POST or PUT call to the URL:

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

Add the Content-Type header to specify the media type of the data you want to send with the request.

Request Body

Send the updates to the relationship record in the request body. Use the JSON format or the XML format to send data. Provide the required parameter values in the request body.

Sample API Request

The relationship with the row ID 414721 is a OrganizationEmploysPerson relationship between an Organization entity with row ID 101 and a Person entity with row ID 1101.

The following sample request updates the relationship record with the row ID 414721:

```
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"}


}
```

Sample API Response

The following sample response is received after successfully updating the relationship with the row ID 414721:

```
{
        "OrganizationEmployesPerson": {
                "key": {
                                "rowid": "414721"
                                "sourceKey": "SVR1.1E7UW"
                                        }-
            "rowidObject": "414721"
        "from": {
            "key": {
                            "rowid": "101"
                                    }-
                "rowidObject": "101"
                }-
        "to": {
                "key": {
                            "rowid": "1101 "
                                }-
        "rowidObject": "1101 "
                       }-
            }-
}
```

Delete a Relationship

The Delete Relationship REST API deletes the relationship between two records.

The API uses the DELETE method.

Request URL

The Delete Relationship URL has the following format:

```
http://<host>:<port>/<context>/<database ID>/<relationship>/<rowID of the relationship record>?systemName=<name of the source system>
```

Note: The name of the source system as a required parameter in the URL.

Make the following HTTP DELETE request to the Delete URL:

```
DELETE http://<host>:<port>/<context>/<database ID>/<relationship>/<rowID of the relationship record>?systemName=<name of the source system>
```

Query Parameter

The name of the source system is a required URL parameter. Use the `systemName` parameter to specify the source system.

Sample API Request

The following sample request deletes a relationship record with the row ID 414721:

```
DELETE http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/OrganizationEmploysPerson/414721?systemName=SFA
```

Sample API Response

The following sample response shows the response after successfully deleting the relationship record with the row ID 414721:

```
{
"OrganizationEmployesPerson": {
        "key": {
                "rowid": "414721"
                }-
"rowidObject": "414721"
            }-
}
```

Get Related Records

The Get Related Records REST API returns a list of records related to a specified root record based on the relationships that you have configured. The API also returns the details of the relationships.

The API uses the GET method.

Request URL

The Get Related Records URL has the following format:

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

Make the following HTTP GET request to the Get Related Records URL:

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

Query Parameters

You can append the query parameters to the request URL.

The following table lists the query parameters that you can use:

Parameter	Description
recordsToReturn	Number of records for the `many` child that you want to read.
searchToken	Search token to fetch subsequent pages of the result set.
returnTotal	Returns the number of records in the result set. Set to true to get the number of records in the result set. Default is false.

Filter Parameters

You can append parameters to the URL to filter the related records.

The following table lists the filter parameters that you can use:

Parameter	Description
recordStates	A comma-separated list of states of records that you want to retrieve. The supported record states are ACTIVE, PENDING, and DELETED. Default is ACTIVE. For example, the `/Party/123?action=related&recordStates=ACTIVE,PENDING` query returns records that are active or pending.
entityLabel	Label of the entity.
relationshipLabel	Label of the relationship.
entityType	Comma-separated list of entity types. For example, the `entityType=Person,Organization` list returns related records of the `Person` and `Organization` entity type.
relationshipType	Comma-separated list of relationship types. For example, the `relationshipType=Employee,Employer` list returns related records of the `Employee` and `Employer` relationship types.

Note: If you specify multiple filter conditions, the result contains all the records that satisfy the AND condition.

Response Body

The response body contains the list of related records, details of the related records and the relationships, and a search token. Use the search token to fetch the subsequent pages of the results.

Sample API Request

The following sample request fetches the related records and relationships configured for the Organization business entity with the row ID 101:

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Organization/101?action=related
```

Sample API Response

The following sample response shows the related records and relationships for the Organization business entity type with the row ID 101:

```
{
  "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"
      }
    ]
}
```


Export Related Business Entities

Enter a short description of the concept here (required).

This is the start of the concept.

Request URL

Enter a short description of the concept here (required).

This is the start of the concept.

Sample API Request

Enter a short description of the concept here (required).

This is the start of the concept.

Sample API Response

Enter a short description of the concept here (required).

This is the start of the concept.

List Hierarchies

The List Hierarchies REST API returns all hierarchies. You can also use the API to return the hierarchies that contain a specified business entity at any level.

The API uses the GET method.

Request URL

The List Hierarchies URL can have the following formats:

URL to return all hierarchies

Use the following URL to list all hierarchies:

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

URL to return all hierarchies that contain a specified business entity

Use the following URL to list all hierarchies that contain a specified business entity at any level:

```
http://<host>:<port>/<context>/<database ID>/metadata/hierarchy?entityName=<entity
name>
```

Sample API Request

The following sample request lists all hierarchies:

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

The following sample request lists all hierarchies that contain the Product business entity:

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/metadata/hierarchy?
entityName=Product
```

Sample API Response

The following sample response shows the hierarchies that contain the Product business entity:

```
{
  "link":[],
  "item":[
    {
      "operations":{
        "read":{
          "allowed":true
        },
        "update":{
          "allowed":true
        }
      },
      "root":"ProductGroup",
      "name":"Product"
    }
  ]
}
```

Get Hierarchy Metadata

The Get Hierarchy Metadata REST API returns the metadata for a hierarchy.

The API uses the GET method.

Request URL

The Get Hierarchy Metadata URL has the following format:

```
http://<host>:<port>/<context>/<database ID>/metadata/hierarchy/<hierarchy name>
```

Make the following HTTP GET request to the Get Hierarchy Metadata URL:

```
GET http://<host>:<port>/<context>/<database ID>/metadata/hierarchy/<hierarchy name>
```

Sample API Request

The following sample request returns the metadata for a hierarchy:

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/metadata/hierarchy/ODI
```

Sample API Response

The following sample response shows the metadata for a hierarchy:

```
{
  "relationship":[
    {
      "field":[
        {
          "allowedValues":[
            "2"
          ],
          "name":"rowidRelType",
          "label":"Rowid Rel Type",
          "dataType":"String",
          "length":14,
          "readOnly":false,
          "required":false,
          "system":false,
          "trust":false,
          "applyNullValues":false,
          "displayFormat":"DATETIME_LONG_FORMAT",
          "filterable":true,
          "sortable":true,
          "lookup":{
            "link":[
              {
                "href":"http://localhost:8080/cmx/lookup/localhost-orcl-MDM_SAMPLE/id-
label/IsOdiParentOf/rowidRelType",
                "rel":"lookup"
              },
              {
                "href":"http://localhost:8080/cmx/lookup/localhost-orcl-MDM_SAMPLE/
object/IsOdiParentOf/rowidRelType",
                "rel":"list"
              }
            ],
            "object":"RboRelType",
            "key":"rowidObject",
            "value":"displayName"
          }
        },
        {
          "allowedValues":[
            "1"
          ],
          "name":"rowidHierarchy",
          "label":"Rowid Hierarchy",
          "dataType":"String",
          "length":14,
          "readOnly":false,
          "required":false,
          "system":false,
          "trust":false,
          "applyNullValues":false,
          "displayFormat":"DATETIME_LONG_FORMAT",
          "filterable":true,
          "sortable":true,
          "lookup":{
            "link":[
              {
                "href":"http://localhost:8080/cmx/lookup/localhost-orcl-MDM_SAMPLE/id-
label/IsOdiParentOf/rowidHierarchy",
                "rel":"lookup"
```

```
              },
              {
                "href":"http://localhost:8080/cmx/lookup/localhost-orcl-MDM_SAMPLE/
object/IsOdiParentOf/rowidHierarchy",
                "rel":"list"
              }
            ],
            "object":"RboHierarchy",
            "key":"rowidObject",
            "value":"displayName"
          }
        },
        {
          "name":"relName",
          "label":"Rel Name",
          "dataType":"String",
          "length":50,
          "trust":false,
          "applyNullValues":false,
          "displayFormat":"DATETIME_LONG_FORMAT",
          "filterable":true,
          "sortable":true
        },
        {
          "name":"relDesc",
          "label":"Rel Desc",
          "dataType":"String",
          "length":200,
          "trust":false,
          "applyNullValues":false,
          "displayFormat":"DATETIME_LONG_FORMAT",
          "filterable":true,
          "sortable":true
        },
        {
          "name":"consolidationInd",
          "label":"Consolidation Ind",
          "dataType":"Integer",
          "length":38,
          "readOnly":true,
          "system":true,
          "trust":false,
          "applyNullValues":false,
          "filterable":true,
          "sortable":true
        },
        {
          "name":"creator",
          "label":"Creator",
          "dataType":"String",
          "length":50,
          "readOnly":true,
          "system":true,
          "trust":false,
          "applyNullValues":false,
          "filterable":true,
          "sortable":true
        },
        {
          "name":"interactionId",
          "label":"Interaction Id",
          "dataType":"Integer",
          "length":38,
          "readOnly":true,
          "system":true,
          "trust":false,
          "applyNullValues":false,
          "filterable":true,
          "sortable":true
        },
        {
```

```
  "name":"updatedBy",
  "label":"Updated By",
  "dataType":"String",
  "length":50,
  "readOnly":true,
  "system":true,
  "trust":false,
  "applyNullValues":false,
  "filterable":true,
  "sortable":true
},
{
  "name":"lastUpdateDate",
  "label":"Last Update Date",
  "dataType":"Date",
  "readOnly":true,
  "system":true,
  "trust":false,
  "applyNullValues":false,
  "filterable":true,
  "sortable":true
},
{
  "name":"lastRowidSystem",
  "label":"Last Rowid System",
  "dataType":"String",
  "length":14,
  "readOnly":true,
  "system":true,
  "trust":false,
  "applyNullValues":false,
  "filterable":true,
  "sortable":true
},
{
  "name":"dirtyIndicator",
  "label":"Dirty Indicator",
  "dataType":"Integer",
  "length":38,
  "readOnly":true,
  "system":true,
  "trust":false,
  "applyNullValues":false,
  "filterable":true,
  "sortable":true
},
{
  "name":"deletedBy",
  "label":"Deleted By",
  "dataType":"String",
  "length":50,
  "readOnly":true,
  "system":true,
  "trust":false,
  "applyNullValues":false,
  "filterable":true,
  "sortable":true
},
{
  "name":"deletedInd",
  "label":"Deleted Indicator",
  "dataType":"Integer",
  "length":38,
  "readOnly":true,
  "system":true,
  "trust":false,
  "applyNullValues":false,
  "filterable":true,
  "sortable":true
},
{
```

```
      "name":"hubStateInd",
      "label":"Hub State Ind",
      "dataType":"Integer",
      "length":38,
      "readOnly":true,
      "system":true,
      "trust":false,
      "applyNullValues":false,
      "filterable":true,
      "sortable":true
    },
    {
      "name":"deletedDate",
      "label":"Deleted Date",
      "dataType":"Date",
      "readOnly":true,
      "system":true,
      "trust":false,
      "applyNullValues":false,
      "filterable":true,
      "sortable":true
    },
    {
      "name":"rowidObject",
      "label":"Rowid Object",
      "dataType":"String",
      "length":14,
      "readOnly":true,
      "system":true,
      "trust":false,
      "applyNullValues":false,
      "filterable":true,
      "sortable":true
    },
    {
      "name":"cmDirtyInd",
      "label":"Content metadata dirty Ind",
      "dataType":"Integer",
      "length":38,
      "readOnly":true,
      "system":true,
      "trust":false,
      "applyNullValues":false,
      "filterable":true,
      "sortable":true
    },
    {
      "name":"createDate",
      "label":"Create Date",
      "dataType":"Date",
      "readOnly":true,
      "system":true,
      "trust":false,
      "applyNullValues":false,
      "filterable":true,
      "sortable":true
    }
  ],
  "contentMetadata":[
    {
      "operations":{
        "read":{
          "allowed":true
        },
        "create":{
          "allowed":true
        },
        "update":{
          "allowed":true
        },
        "delete":{
```

```
            "allowed":true
          }
        },
        "name":"XREF"
      }
    ],
    "name":"IsOdiParentOf",
    "label":"is ODI parent of",
    "color":"#990066",
    "direction":"ENTITY_1_TO_ENTITY_2",
    "bidirectional":false,
    "from":{
      "dataType":"BusinessEntity",
      "required":true,
      "lookup":{
        "object":"Organization"
      }
    },
    "to":{
      "dataType":"BusinessEntity",
      "required":true,
      "lookup":{
        "object":"Organization"
      }
    },
    "hierarchy":"ODI"
  }
 ],
 "operations":{
   "read":{
     "allowed":true
   },
   "update":{
     "allowed":true
   }
 },
 "root":"Organization",
 "name":"ODI"
}
```


Get Hierarchy Path

The Get Hierarchy Paths REST API returns the paths from a business entity record to the hierarchy root.

The API uses the GET method.

Request URL

The Get Hierarchy Paths URL has the following format:

```
http://<host>:<port>/<context>/<database ID>/hierarchy/<hierarchy name>/entity/<entity
name>/<entity ID>/path
```

Make the following HTTP GET request to the Get Hierarchy Paths URL:

```
GET http://<host>:<port>/<context>/<database ID>/hierarchy/<hierarchy name>/entity/
<entity name>/<entity ID>/path
```

Query Parameters

You can append query parameters to the request URL to filter the relationships.

The following table describes the query parameters:

Parameter	Description
effectiveDate	Date for which you want to retrieve the relationships.
interactionId	ID of the interaction.
readLabelsOnly	Indicates whether to return only labels in the result. Values are `true` or `false`.
readSystemFields	Indicates whether to return the system fields in the result. Values are `true` or `false`.
recordsToReturn	Specifies the number of rows to return.
rejectInteractionId	ID of the interaction for rejecting changes.
relationshipType	Comma-separated list of relationship types. For example, the `relationshipType=Employee,Employer list` returns related records of the `Employee` and `Employer` relationship types.
returnTotal	Returns the number of records in the result set. Set to `true` to get the number of records in the result set.
rootId	ID of the hierarchy root.

Sample API Request

The following sample request lists the paths from a business entity record to the hierarchy root:

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/hierarchy/ODI/entity/
Organization/35/path
```

Sample API Response

The following sample response shows the paths from the business entity record to the hierarchy root:

```
{
  "link":[],
  "item":[
    {
      "businessEntity":{
        "Organization":{
          "link":[
            {
              "href":"http://localhost:8080/cmx/request/hm_icons/hierarchymanager/
Hospital/Hospital.png?ors=localhost-orcl-MDM_SAMPLE",
              "rel":"icon"
            },
            {
              "href":"http://localhost:8080/cmx/cs/localhost-orcl-MDM_SAMPLE/
Organization/584",
              "rel":"self"
            }
          ],
          "rowidObject":"584",
          "label":"Time Warner Inc",
          "partyType":"Organization",
          "displayName":"Time Warner Inc",
```

```
          "dunsNumber":"799527630"
        }
      },
      "entityLabel":"Time Warner Inc",
      "entityType":"Organization",
      "depth":2,
      "object":{
        "businessEntity":{
          "Organization":{
            "link":[
              {
                "href":"http://localhost:8080/cmx/request/hm_icons/hierarchymanager/
Hospital/Hospital.png?ors=localhost-orcl-MDM_SAMPLE",
                "rel":"icon"
              },
              {
                "href":"http://localhost:8080/cmx/cs/localhost-orcl-MDM_SAMPLE/
Organization/55",
                "rel":"self"
              }
            ],
            "rowidObject":"55",
            "label":"Historic TW Inc",
            "partyType":"Organization",
            "displayName":"Historic TW Inc",
            "dunsNumber":"958466278"
          }
        },
        "entityLabel":"Historic TW Inc",
        "relationshipLabel":"is ODI parent of",
        "relationship":{
          "rowidObject":"834",
          "label":"is ODI parent of",
          "rowidRelType":"2",
          "rowidHierarchy":"1",
          "relName":"Parent",
          "relDesc":"Parent",
          "from":{
            "rowidObject":"584"
          },
          "to":{
            "rowidObject":"55"
          }
        },
        "entityType":"Organization",
        "relationshipType":"IsOdiParentOf",
        "object":{
          "businessEntity":{
            "Organization":{
              "link":[
                {
                  "href":"http://localhost:8080/cmx/request/hm_icons/hierarchymanager/
Hospital/Hospital.png?ors=localhost-orcl-MDM_SAMPLE",
                  "rel":"icon"
                },
                {
                  "href":"http://localhost:8080/cmx/cs/localhost-orcl-MDM_SAMPLE/
Organization/35",
                  "rel":"self"
                }
              ],
              "rowidObject":"35",
              "effectivePeriod":{

              },
              "label":"People Magazine",
              "partyType":"Organization",
              "displayName":"People Magazine",
              "dunsNumber":"011584096"
            }
          },
```

```
            "entityLabel":"People Magazine",
            "relationshipLabel":"is ODI parent of",
            "relationship":{
              "rowidObject":"889",
              "label":"is ODI parent of",
              "rowidRelType":"2",
              "rowidHierarchy":"1",
              "relName":"Parent",
              "relDesc":"Parent",
              "from":{
                "rowidObject":"55"
              },
              "to":{
                "rowidObject":"35"
              }
            },
            "entityType":"Organization",
            "relationshipType":"IsOdiParentOf"
          }
        }
      }
    ]
  }
```

Get Parents

The Get Parents REST API returns the direct parent relationships for a business entity record.

The API uses the GET method.

Request URL

The Get Parents URL has the following format:

```
http://<host>:<port>/<context>/<database ID>/hierarchy/<hierarchy name>/entity/<entity name>/<entity ID>/parent
```

Make the following HTTP GET request to the Get Parents URL:

```
GET http://<host>:<port>/<context>/<database ID>/hierarchy/<hierarchy name>/entity/<entity name>/<entity ID>/parent
```

Query Parameters

You can append query parameters to the request URL to filter the relationships.

The following table describes the query parameters:

Parameter	Description
effectiveDate	Date for which you want to retrieve the relationships.
interactionId	ID of the interaction.
readLabelsOnly	Indicates whether to return only labels in the result. Values are `true` or `false`.
readSystemFields	Indicates whether to return the system fields in the result. Values are `true` or `false`.
recordsToReturn	Specifies the number of rows to return.
rejectInteractionId	ID of the interaction for rejecting changes.

Parameter	Description
relationshipType	Comma-separated list of relationship types. For example, the `relationshipType=Employee,Employer list` returns related records of the `Employee` and `Employer` relationship types.
returnTotal	Returns the number of records in the result set. Set to `true` to get the number of records in the result set.
rootId	ID of the hierarchy root.

Sample API Request

The following sample request lists the direct parent relationships for a business entity record:

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/hierarchy/ODI/entity/
Organization/55/parent
```

Sample API Response

The following sample response shows the direct parent relationships for a business entity record:

```
{
  "link":[],
  "firstRecord":1,
  "pageSize":10,
  "searchToken":"SVR1.1G7VH",
  "relatedEntity":[
    {
      "businessEntity":{
        "Organization":{
          "link":[
            {
              "href":"http://localhost:8080/cmx/cs/localhost-orcl-MDM_SAMPLE/
Organization/584",
              "rel":"self"
            },
            {
              "href":"http://localhost:8080/cmx/request/hm_icons/hierarchymanager/
Hospital/Hospital.png?ors=localhost-orcl-MDM_SAMPLE",
              "rel":"icon"
            }
          ],
          "rowidObject":"584",
          "label":"Time Warner Inc",
          "partyType":"Organization",
          "displayName":"Time Warner Inc",
          "dunsNumber":"799527630"
        }
      },
      "entityLabel":"Time Warner Inc",
      "relationshipLabel":"is ODI parent of",
      "relationship":{
        "rowidObject":"834",
        "label":"is ODI parent of",
        "rowidRelType":"2",
        "rowidHierarchy":"1",
        "relName":"Parent",
        "relDesc":"Parent",
        "from":{
          "rowidObject":"584"
        },
        "to":{
          "rowidObject":"55"
        }
```

```
        },
        "entityType":"Organization",
        "relationshipType":"IsOdiParentOf"
      }
    ]
}
```

Get Children

The Get Children REST API returns the children relationships for a business entity record.

The API uses the GET method.

Request URL

The Get Children URL has the following format:

```
http://<host>:<port>/<context>/<database ID>/hierarchy/<hierarchy name>/entity/<entity name>/<entity ID>/children
```

Make the following HTTP GET request to the Get Children URL:

```
GET http://<host>:<port>/<context>/<database ID>/hierarchy/<hierarchy name>/entity/<entity name>/<entity ID>/children
```

Query Parameters

You can append query parameters to the request URL to filter the relationships.

The following table describes the query parameters:

Parameter	Description
effectiveDate	Date for which you want to retrieve the relationships.
interactionId	ID of the interaction.
readLabelsOnly	Indicates whether to return only labels in the result. Values are `true` or `false`.
readSystemFields	Indicates whether to return the system fields in the result. Values are `true` or `false`.
recordsToReturn	Specifies the number of rows to return.
rejectInteractionId	ID of the interaction for rejecting changes.
relationshipType	Comma-separated list of relationship types. For example, the `relationshipType=Employee,Employer list` returns related records of the `Employee` and `Employer` relationship types.
returnTotal	Returns the number of records in the result set. Set to `true` to get the number of records in the result set.
rootId	ID of the hierarchy root.

Sample API Request

The following sample request lists the children relationships for a business entity record:

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/hierarchy/ODI/entity/
Organization/55/children
```

Sample API Response

The following sample response shows the children relationships for a business entity record:


```
{
  "link":[],
  "firstRecord":1,
  "pageSize":10,
  "searchToken":"SVR1.1G7VF",
  "relatedEntity":[
    {
      "businessEntity":{
        "Organization":{
          "link":[
            {
              "href":"http://localhost:8080/cmx/request/hm_icons/hierarchymanager/
Hospital/Hospital.png?ors=localhost-orcl-MDM_SAMPLE",
              "rel":"icon"
            },
            {
              "href":"http://localhost:8080/cmx/cs/localhost-orcl-MDM_SAMPLE/
Organization/593",
              "rel":"self"
            }
          ],
          "rowidObject":"593           ",
          "label":"Turner Broadcasting System Inc",
          "partyType":"Organization",
          "displayName":"Turner Broadcasting System Inc",
          "dunsNumber":"003319068"
        }
      },
      "entityLabel":"Turner Broadcasting System Inc",
      "relationshipLabel":"is ODI parent of",
      "relationship":{
        "rowidObject":"887           ",
        "label":"is ODI parent of",
        "rowidRelType":"2             ",
        "rowidHierarchy":"1             ",
        "relName":"Parent",
        "relDesc":"Parent",
        "from":{
          "rowidObject":"55            "
        },
        "to":{
          "rowidObject":"593           "
        }
      },
      "entityType":"Organization",
      "relationshipType":"IsOdiParentOf"
    },
    {
      "businessEntity":{
        "Organization":{
          "link":[
            {
              "href":"http://localhost:8080/cmx/request/hm_icons/hierarchymanager/
Hospital/Hospital.png?ors=localhost-orcl-MDM_SAMPLE",
              "rel":"icon"
            },
            {
              "href":"http://localhost:8080/cmx/cs/localhost-orcl-MDM_SAMPLE/
Organization/414",
```

```
            "rel":"self"
          }
        ],
        "rowidObject":"414           ",
        "label":"Historic TW Inc",
        "partyType":"Organization",
        "displayName":"Historic TW Inc",
        "dunsNumber":"007305290"
      }
    },
    "entityLabel":"Historic TW Inc",
    "relationshipLabel":"is ODI parent of",
    "relationship":{
      "rowidObject":"888           ",
      "label":"is ODI parent of",
      "rowidRelType":"2             ",
      "rowidHierarchy":"1             ",
      "relName":"HQ",
      "relDesc":"HQ",
      "from":{
        "rowidObject":"55            "
      },
      "to":{
        "rowidObject":"414           "
      }
    },
    "entityType":"Organization",
    "relationshipType":"IsOdiParentOf"
  },
  {
    "businessEntity":{
      "Organization":{
        "link":[
          {
            "href":"http://localhost:8080/cmx/cs/localhost-orcl-MDM_SAMPLE/
Organization/35",
            "rel":"self"
          },
          {
            "href":"http://localhost:8080/cmx/request/hm_icons/hierarchymanager/
Hospital/Hospital.png?ors=localhost-orcl-MDM_SAMPLE",
            "rel":"icon"
          }
        ],
        "rowidObject":"35            ",
        "label":"People Magazine",
        "partyType":"Organization",
        "displayName":"People Magazine",
        "dunsNumber":"011584096"
      }
    },
    "entityLabel":"People Magazine",
    "relationshipLabel":"is ODI parent of",
    "relationship":{
      "rowidObject":"889           ",
      "label":"is ODI parent of",
      "rowidRelType":"2             ",
      "rowidHierarchy":"1             ",
      "relName":"Parent",
      "relDesc":"Parent",
      "from":{
        "rowidObject":"55            "
      },
      "to":{
        "rowidObject":"35            "
      }
    },
    "entityType":"Organization",
    "relationshipType":"IsOdiParentOf"
  },
  {
```

```
      "businessEntity":{
        "Organization":{
          "link":[
            {
              "href":"http://localhost:8080/cmx/cs/localhost-orcl-MDM_SAMPLE/
Organization/391",
              "rel":"self"
            },
            {
              "href":"http://localhost:8080/cmx/request/hm_icons/hierarchymanager/
Hospital/Hospital.png?ors=localhost-orcl-MDM_SAMPLE",
              "rel":"icon"
            }
          ],
          "rowidObject":"391           ",
          "label":"Historic TW Inc",
          "partyType":"Organization",
          "displayName":"Historic TW Inc",
          "dunsNumber":"011816704"
        }
      },
      "entityLabel":"Historic TW Inc",
      "relationshipLabel":"is ODI parent of",
      "relationship":{
        "rowidObject":"890           ",
        "label":"is ODI parent of",
        "rowidRelType":"2             ",
        "rowidHierarchy":"1             ",
        "relName":"HQ",
        "relDesc":"HQ",
        "from":{
          "rowidObject":"55            "
        },
        "to":{
          "rowidObject":"391           "
        }
      },
      "entityType":"Organization",
      "relationshipType":"IsOdiParentOf"
    },
    {
      "businessEntity":{
        "Organization":{
          "link":[
            {
              "href":"http://localhost:8080/cmx/cs/localhost-orcl-MDM_SAMPLE/
Organization/448",
              "rel":"self"
            },
            {
              "href":"http://localhost:8080/cmx/request/hm_icons/hierarchymanager/
Hospital/Hospital.png?ors=localhost-orcl-MDM_SAMPLE",
              "rel":"icon"
            }
          ],
          "rowidObject":"448           ",
          "label":"Historic TW Inc",
          "partyType":"Organization",
          "displayName":"Historic TW Inc",
          "dunsNumber":"069110711"
        }
      },
      "entityLabel":"Historic TW Inc",
      "relationshipLabel":"is ODI parent of",
      "relationship":{
        "rowidObject":"891           ",
        "label":"is ODI parent of",
        "rowidRelType":"2             ",
        "rowidHierarchy":"1             ",
        "relName":"HQ",
        "relDesc":"HQ",
```

```
        "from":{
          "rowidObject":"55            "
        },
        "to":{
          "rowidObject":"448           "
        }
      },
      "entityType":"Organization",
      "relationshipType":"IsOdiParentOf"
    },
    {
      "businessEntity":{
        "Organization":{
          "link":[
            {
              "href":"http://localhost:8080/cmx/request/hm_icons/hierarchymanager/
Hospital/Hospital.png?ors=localhost-orcl-MDM_SAMPLE",
              "rel":"icon"
            },
            {
              "href":"http://localhost:8080/cmx/cs/localhost-orcl-MDM_SAMPLE/
Organization/545",
              "rel":"self"
            }
          ],
          "rowidObject":"545           ",
          "label":"Historic TW Inc",
          "partyType":"Organization",
          "displayName":"Historic TW Inc",
          "dunsNumber":"072077139"
        }
      },
      "entityLabel":"Historic TW Inc",
      "relationshipLabel":"is ODI parent of",
      "relationship":{
        "rowidObject":"892           ",
        "label":"is ODI parent of",
        "rowidRelType":"2             ",
        "rowidHierarchy":"1             ",
        "relName":"HQ",
        "relDesc":"HQ",
        "from":{
          "rowidObject":"55            "
        },
        "to":{
          "rowidObject":"545           "
        }
      },
      "entityType":"Organization",
      "relationshipType":"IsOdiParentOf"
    },
    {
      "businessEntity":{
        "Organization":{
          "link":[
            {
              "href":"http://localhost:8080/cmx/cs/localhost-orcl-MDM_SAMPLE/
Organization/349",
              "rel":"self"
            },
            {
              "href":"http://localhost:8080/cmx/request/hm_icons/hierarchymanager/
Hospital/Hospital.png?ors=localhost-orcl-MDM_SAMPLE",
              "rel":"icon"
            }
          ],
          "rowidObject":"349           ",
          "label":"Historic TW Inc",
          "partyType":"Organization",
          "displayName":"Historic TW Inc",
          "dunsNumber":"074056123"
```

```
        }
      },
      "entityLabel":"Historic TW Inc",
      "relationshipLabel":"is ODI parent of",
      "relationship":{
        "rowidObject":"893           ",
        "label":"is ODI parent of",
        "rowidRelType":"2             ",
        "rowidHierarchy":"1             ",
        "relName":"HQ",
        "relDesc":"HQ",
        "from":{
          "rowidObject":"55            "
        },
        "to":{
          "rowidObject":"349           "
        }
      },
      "entityType":"Organization",
      "relationshipType":"IsOdiParentOf"
    },
    {
      "businessEntity":{
        "Organization":{
          "link":[
            {
              "href":"http://localhost:8080/cmx/request/hm_icons/hierarchymanager/
Hospital/Hospital.png?ors=localhost-orcl-MDM_SAMPLE",
              "rel":"icon"
            },
            {
              "href":"http://localhost:8080/cmx/cs/localhost-orcl-MDM_SAMPLE/
Organization/298",
              "rel":"self"
            }
          ],
          "rowidObject":"298           ",
          "label":"WF Cinema Holdings LP",
          "partyType":"Organization",
          "displayName":"WF Cinema Holdings LP",
          "dunsNumber":"075408711"
        }
      },
      "entityLabel":"WF Cinema Holdings LP",
      "relationshipLabel":"is ODI parent of",
      "relationship":{
        "rowidObject":"894           ",
        "label":"is ODI parent of",
        "rowidRelType":"2             ",
        "rowidHierarchy":"1             ",
        "relName":"Parent",
        "relDesc":"Parent",
        "from":{
          "rowidObject":"55            "
        },
        "to":{
          "rowidObject":"298           "
        }
      },
      "entityType":"Organization",
      "relationshipType":"IsOdiParentOf"
    },
    {
      "businessEntity":{
        "Organization":{
          "link":[
            {
              "href":"http://localhost:8080/cmx/request/hm_icons/hierarchymanager/
Hospital/Hospital.png?ors=localhost-orcl-MDM_SAMPLE",
              "rel":"icon"
            },
```

```
            {
              "href":"http://localhost:8080/cmx/cs/localhost-orcl-MDM_SAMPLE/
Organization/596",
              "rel":"self"
            }
          ],
          "rowidObject":"596           ",
          "label":"Historic TW Inc",
          "partyType":"Organization",
          "displayName":"Historic TW Inc",
          "dunsNumber":"076749246"
        }
      },
      "entityLabel":"Historic TW Inc",
      "relationshipLabel":"is ODI parent of",
      "relationship":{
        "rowidObject":"895           ",
        "label":"is ODI parent of",
        "rowidRelType":"2             ",
        "rowidHierarchy":"1             ",
        "relName":"HQ",
        "relDesc":"HQ",
        "from":{
          "rowidObject":"55            "
        },
        "to":{
          "rowidObject":"596           "
        }
      },
      "entityType":"Organization",
      "relationshipType":"IsOdiParentOf"
    },
    {
      "businessEntity":{
        "Organization":{
          "link":[
            {
              "href":"http://localhost:8080/cmx/request/hm_icons/hierarchymanager/
Hospital/Hospital.png?ors=localhost-orcl-MDM_SAMPLE",
              "rel":"icon"
            },
            {
              "href":"http://localhost:8080/cmx/cs/localhost-orcl-MDM_SAMPLE/
Organization/699",
              "rel":"self"
            }
          ],
          "rowidObject":"699           ",
          "label":"Historic TW Inc",
          "partyType":"Organization",
          "displayName":"Historic TW Inc",
          "dunsNumber":"083115568"
        }
      },
      "entityLabel":"Historic TW Inc",
      "relationshipLabel":"is ODI parent of",
      "relationship":{
        "rowidObject":"896           ",
        "label":"is ODI parent of",
        "rowidRelType":"2             ",
        "rowidHierarchy":"1             ",
        "relName":"HQ",
        "relDesc":"HQ",
        "from":{
          "rowidObject":"55            "
        },
        "to":{
          "rowidObject":"699           "
        }
      },
      "entityType":"Organization",
```

```
      "relationshipType":"IsOdiParentOf"
    }
  ]
}
```

Export Hierarchy

The Export Hierarchy REST API exports a hierarchy in CSV format.

The API uses the GET method.

Request URL

The Export Hierarchy URL has the following format:

```
http://<host>:<port>/<context>/<database ID>/hierarchy/<hierarchy name>/entity/<entity name>/<entity ID>/exportHierarchy
```

Make the following HTTP GET request to the Export Hierarchy URL:

```
GET http://<host>:<port>/<context>/<database ID>/hierarchy/<hierarchy name>/entity/<entity name>/<entity ID>/exportHierarchy
```

Query Parameters

You can append query parameters to the request URL to filter the relationships.

The following table describes the query parameters:

Parameter	Description
effectiveDate	Date for which you want to retrieve the relationships.
interactionId	ID of the interaction.
readLabelsOnly	Indicates whether to return only labels in the result. Values are `true` or `false`.
readSystemFields	Indicates whether to return the system fields in the result. Values are `true` or `false`.
recordsToReturn	Specifies the number of rows to return.
rejectInteractionId	ID of the interaction for rejecting changes.
relationshipType	Comma-separated list of relationship types. For example, the `relationshipType=Employee,Employer list` returns related records of the `Employee` and `Employer` relationship types.
returnTotal	Returns the number of records in the result set. Set to `true` to get the number of records in the result set.
rootId	ID of the hierarchy root.

Sample API Request

The following sample request exports the hierarchy for a business entity record:

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/hierarchy/ODI/entity/Organization/584/exportHierarchy
```

Sample API Response

The following sample response shows the exported hierarchy in the CSV format:

```
Level,Business Entity,ID,Label,Parent ID,Parent Label,Type of Relationship to
Parent,issuingCompany,expirationYear,accountType,accountNumber,securityCode,expirationMon
th,cardholderName,productType,productNumber,productName,inceptionDate,productDesc,groupTy
pe,groupName,partyType,lastName,displayName,middleName,birthdate,firstName,dunsNumber
Root,Organization,584,Time Warner Inc,,,,,,,,,,,,,,,,,Organization,,Time Warner Inc,,,,
799527630
1,Organization,55,Historic TW Inc,584,Time Warner Inc,is ODI parent
of,,,,,,,,,,,,,,,Organization,,Historic TW Inc,,,,958466278
1,Organization,206,Courtroom Television Network LLC,584,Time Warner Inc,is ODI parent
of,,,,,,,,,,,,,,,Organization,,Courtroom Television Network LLC,,,,043905707
1,Organization,674,Rebellion Pictures LLC,584,Time Warner Inc,is ODI parent
of,,,,,,,,,,,,,,,Organization,,Rebellion Pictures LLC,,,,557414278
1,Organization,881,"Time Warner Cable Programming, Inc",584,Time Warner Inc,is ODI
parent of,,,,,,,,,,,,,,,Organization,,"Time Warner Cable Programming, Inc",,,,787474949
...
```

Export Direct Children and Parents

The Export Direct Children and Parents REST API exports the direct children and parent relationships for a business entity record in CSV format.

The API uses the GET method.

Request URL

The Export Direct Children and Parents URL has the following format:

```
http://<host>:<port>/<context>/<database ID>/hierarchy/<hierarchy name>/entity/<entity
name>/<entity ID>/exportRelated
```

Make the following HTTP GET request to the Export Direct Children and Parents URL:

```
GET http://<host>:<port>/<context>/<database ID>/hierarchy/<hierarchy name>/entity/
<entity name>/<entity ID>/exportRelated
```

Query Parameters

You can append query parameters to the request URL to filter the relationships.

The following table describes the query parameters:

Parameter	Description
effectiveDate	Date for which you want to retrieve the relationships.
interactionId	ID of the interaction.
readLabelsOnly	Indicates whether to return only labels in the result. Values are `true` or `false`.
readSystemFields	Indicates whether to return the system fields in the result. Values are `true` or `false`.
recordsToReturn	Specifies the number of rows to return.
rejectInteractionId	ID of the interaction for rejecting changes.

Parameter	Description
relationshipType	Comma-separated list of relationship types. For example, the `relationshipType=Employee,Employer list` returns related records of the `Employee` and `Employer` relationship types.
returnTotal	Returns the number of records in the result set. Set to `true` to get the number of records in the result set.
rootId	ID of the hierarchy root.

Sample API Request

The following sample request exports the direct children and parent relationships for a business entity record:

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/hierarchy/ODI/entity/
Organization/55/exportRelated
```

Sample API Response

The following sample response shows the exported direct children and parent relationships for a business entity record in CSV format:

```
Level,Business Entity,ID,Label,Parent ID,Parent Label,Type of Relationship to
Parent,issuingCompany,expirationYear,accountType,accountNumber,securityCode,expirationMon
th,cardholderName,productType,productNumber,productName,inceptionDate,productDesc,groupTy
pe,groupName,partyType,lastName,displayName,middleName,birthdate,firstName,dunsNumber
Root,Organization,584,Time Warner Inc,,,,,,,,,,,,,,,,Organization,,Time Warner Inc,,,,
799527630
1,Organization,55,Historic TW Inc,584,Time Warner Inc,is ODI parent
of,,,,,,,,,,,,,,,Organization,,Historic TW Inc,,,,958466278
2,Organization,593,Turner Broadcasting System Inc,55,Historic TW Inc,is ODI parent
of,,,,,,,,,,,,,,,Organization,,Turner Broadcasting System Inc,,,,003319068
2,Organization,414,Historic TW Inc,55,Historic TW Inc,is ODI parent
of,,,,,,,,,,,,,,,Organization,,Historic TW Inc,,,,007305290
2,Organization,35,People Magazine,55,Historic TW Inc,is ODI parent
of,,,,,,,,,,,,,,,Organization,,People Magazine,,,,011584096
2,Organization,391,Historic TW Inc,55,Historic TW Inc,is ODI parent
of,,,,,,,,,,,,,,,Organization,,Historic TW Inc,,,,011816704
...
```

Get Hierarchy Changes

The Get Hierarchy Changes API retrieves pending changes to a hierarchy.

The API uses the GET method.

Request URL

The Get Hierarchy Changes URL has the following format:

```
http://<host>:<port>/<context>/<database ID>/hierarchy/{hierarchy name}/changes
```

Make the following HTTP GET request to the Get Hierarchy Changes URL:

```
GET http://<host>:<port>/<context>/<database ID>/hierarchy/{hierarchy name}/changes
```

Query Parameters

You can append query parameters to the request URL to filter the relationships.

The following table describes the query parameters:

Parameter	Description
effectiveDate	Date for which you want to retrieve the relationships.
interactionId	ID of the interaction.
readLabelsOnly	Indicates whether to return only labels in the result. Values are `true` or `false`.
readSystemFields	Indicates whether to return the system fields in the result. Values are `true` or `false`.
recordsToReturn	Specifies the number of rows to return.
rejectInteractionId	ID of the interaction for rejecting changes.
relationshipType	Comma-separated list of relationship types. For example, the `relationshipType=Employee,Employer list` returns related records of the `Employee` and `Employer` relationship types.
returnTotal	Returns the number of records in the result set. Set to `true` to get the number of records in the result set.
rootId	ID of the hierarchy root.

Sample API Request

The following request returns pending changes to a hierarchy filtered by interaction ID, root ID, and rejection interaction ID:

```
GET http://localhost:8080/cmx/bulk/localhost-orcl-DS_UI1/hierarchy/Product/changes?
interactionId=480003&rootId=165&rejectInteractionId=480004
```

Sample API Response

The following sample response shows the pending changes:

```
{
  "interactionId":"480003",
  "rejectInteractionId":"480004",
  "type":"hierarchy",
  "name":"Product",
  "objects":[
    {
      "before":[

      ],
      "after":[
        {
          "businessEntity":{
            "ProductGroup":{
              "link":[
                {
                  "href":"http://localhost:8080/cmx/request/hm_icons/hierarchymanager/
Group/Group.png?ors=localhost-orcl-MDM_SAMPLE",
                  "rel":"icon"
                }
              ],
```

```
            "rowidObject":"165           ",
            "creator":"admin",
            "createDate":"2020-01-13T13:07:35.128-05:00",
            "updatedBy":"admin",
            "lastUpdateDate":"2020-01-13T13:07:35.128-05:00",
            "consolidationInd":4,
            "lastRowidSystem":"PRODUCT       ",
            "hubStateInd":1,
            "label":"Vehicles",
            "productType":"Product Group",
            "productName":"Vehicles"
          }
        },
        "entityLabel":"Vehicles",
        "entityType":"ProductGroup",
        "depth":1,
        "object":{
          "businessEntity":{
            "ProductGroup":{
              "link":[
                {
                  "href":"http://localhost:8080/cmx/request/hm_icons/hierarchymanager/
Group/Group.png?ors=localhost-orcl-MDM_SAMPLE",
                  "rel":"icon"
                }
              ],
              "rowidObject":"166           ",
              "label":"Scooter",
              "productType":"Product Group",
              "productName":"Scooter"
            }
          },
          "entityLabel":"Scooter",
          "relationshipLabel":"Product Group is parent of Product Group",
          "relationship":{
            "rowidObject":"124           ",
            "creator":"datasteward",
            "createDate":"2020-01-13T13:07:35.864-05:00",
            "updatedBy":"datasteward",
            "lastUpdateDate":"2020-01-13T13:07:35.865-05:00",
            "consolidationInd":4,
            "lastRowidSystem":"PRODUCT       ",
            "interactionId":"480003",
            "hubStateInd":0,
            "label":"Product Group is parent of Product Group",
            "rowidRelType":"7             ",
            "rowidHierarchy":"3              ",
            "from":{
              "rowidObject":"165           "
            },
            "to":{
              "rowidObject":"166           "
            }
          },
          "entityType":"ProductGroup",
          "relationshipType":"ProductGroupIsParentOfProductGroup"
        }
      },
      {
        "businessEntity":{
          "ProductGroup":{
            "link":[
              {
                "href":"http://localhost:8080/cmx/request/hm_icons/hierarchymanager/
Group/Group.png?ors=localhost-orcl-MDM_SAMPLE",
                "rel":"icon"
              }
            ],
            "rowidObject":"165           ",
            "creator":"admin",
            "createDate":"2020-01-13T13:07:35.128-05:00",
```

```
              "updatedBy":"admin",
              "lastUpdateDate":"2020-01-13T13:07:35.128-05:00",
              "consolidationInd":4,
              "lastRowidSystem":"PRODUCT       ",
              "hubStateInd":1,
              "label":"Vehicles",
              "productType":"Product Group",
              "productName":"Vehicles"
            }
          },
          "entityLabel":"Vehicles",
          "entityType":"ProductGroup",
          "depth":2,
          "object":{
            "businessEntity":{
              "ProductGroup":{
                "link":[
                  {
                    "href":"http://localhost:8080/cmx/request/hm_icons/hierarchymanager/
Group/Group.png?ors=localhost-orcl-MDM_SAMPLE",
                    "rel":"icon"
                  }
                ],
                "rowidObject":"166           ",
                "creator":"admin",
                "createDate":"2020-01-13T13:07:35.250-05:00",
                "updatedBy":"admin",
                "lastUpdateDate":"2020-01-13T13:07:35.250-05:00",
                "consolidationInd":4,
                "lastRowidSystem":"PRODUCT       ",
                "hubStateInd":1,
                "label":"Scooter",
                "productType":"Product Group",
                "productName":"Scooter"
              }
            },
            "entityLabel":"Scooter",
            "relationshipLabel":"Product Group is parent of Product Group",
            "relationship":{
              "rowidObject":"124           ",
              "creator":"datasteward",
              "createDate":"2020-01-13T13:07:35.864-05:00",
              "updatedBy":"datasteward",
              "lastUpdateDate":"2020-01-13T13:07:35.865-05:00",
              "consolidationInd":4,
              "lastRowidSystem":"PRODUCT       ",
              "interactionId":"480003",
              "hubStateInd":0,
              "label":"Product Group is parent of Product Group",
              "rowidRelType":"7             ",
              "rowidHierarchy":"3             ",
              "from":{
                "rowidObject":"165           "
              },
              "to":{
                "rowidObject":"166           "
              }
            },
            "entityType":"ProductGroup",
            "relationshipType":"ProductGroupIsParentOfProductGroup",
            "object":{
              "businessEntity":{
                "ProductGroup":{
                  "link":[
                    {
                      "href":"http://localhost:8080/cmx/request/hm_icons/
hierarchymanager/Group/Group.png?ors=localhost-orcl-MDM_SAMPLE",
                      "rel":"icon"
                    }
                  ],
                  "rowidObject":"167           ",
```

```
                    "label":"E-Scooter",
                    "productType":"Product Group",
                    "productName":"E-Scooter"
                  }
                },
                "entityLabel":"E-Scooter",
                "relationshipLabel":"Product Group is parent of Product Group",
                "relationship":{
                  "rowidObject":"125           ",
                  "creator":"datasteward",
                  "createDate":"2020-01-13T13:07:36.033-05:00",
                  "updatedBy":"datasteward",
                  "lastUpdateDate":"2020-01-13T13:07:36.034-05:00",
                  "consolidationInd":4,
                  "lastRowidSystem":"PRODUCT       ",
                  "interactionId":"480003",
                  "hubStateInd":0,
                  "label":"Product Group is parent of Product Group",
                  "rowidRelType":"7             ",
                  "rowidHierarchy":"3             ",
                  "from":{
                    "rowidObject":"166           "
                  },
                  "to":{
                    "rowidObject":"167           "
                  }
                },
                "entityType":"ProductGroup",
                "relationshipType":"ProductGroupIsParentOfProductGroup"
              }
            }
          },
          {
            "businessEntity":{
              "ProductGroup":{
                "link":[
                  {
                    "href":"http://localhost:8080/cmx/request/hm_icons/hierarchymanager/
Group/Group.png?ors=localhost-orcl-MDM_SAMPLE",
                    "rel":"icon"
                  }
                ],
                "rowidObject":"165           ",
                "creator":"admin",
                "createDate":"2020-01-13T13:07:35.128-05:00",
                "updatedBy":"admin",
                "lastUpdateDate":"2020-01-13T13:07:35.128-05:00",
                "consolidationInd":4,
                "lastRowidSystem":"PRODUCT       ",
                "hubStateInd":1,
                "label":"Vehicles",
                "productType":"Product Group",
                "productName":"Vehicles"
              }
            },
            "entityLabel":"Vehicles",
            "entityType":"ProductGroup",
            "depth":3,
            "object":{
              "businessEntity":{
                "ProductGroup":{
                  "link":[
                    {
                      "href":"http://localhost:8080/cmx/request/hm_icons/hierarchymanager/
Group/Group.png?ors=localhost-orcl-MDM_SAMPLE",
                      "rel":"icon"
                    }
                  ],
                  "rowidObject":"166           ",
                  "creator":"admin",
                  "createDate":"2020-01-13T13:07:35.250-05:00",
```

```
                    "updatedBy":"admin",
                    "lastUpdateDate":"2020-01-13T13:07:35.250-05:00",
                    "consolidationInd":4,
                    "lastRowidSystem":"PRODUCT       ",
                    "hubStateInd":1,
                    "label":"Scooter",
                    "productType":"Product Group",
                    "productName":"Scooter"
                  }
                },
                "entityLabel":"Scooter",
                "relationshipLabel":"Product Group is parent of Product Group",
                "relationship":{
                  "rowidObject":"124           ",
                  "creator":"datasteward",
                  "createDate":"2020-01-13T13:07:35.864-05:00",
                  "updatedBy":"datasteward",
                  "lastUpdateDate":"2020-01-13T13:07:35.865-05:00",
                  "consolidationInd":4,
                  "lastRowidSystem":"PRODUCT       ",
                  "interactionId":"480003",
                  "hubStateInd":0,
                  "label":"Product Group is parent of Product Group",
                  "rowidRelType":"7             ",
                  "rowidHierarchy":"3             ",
                  "from":{
                    "rowidObject":"165           "
                  },
                  "to":{
                    "rowidObject":"166           "
                  }
                },
                "entityType":"ProductGroup",
                "relationshipType":"ProductGroupIsParentOfProductGroup",
                "object":{
                  "businessEntity":{
                    "ProductGroup":{
                      "link":[
                        {
                          "href":"http://localhost:8080/cmx/request/hm_icons/
hierarchymanager/Group/Group.png?ors=localhost-orcl-MDM_SAMPLE",
                          "rel":"icon"
                        }
                      ],
                      "rowidObject":"167           ",
                      "creator":"admin",
                      "createDate":"2020-01-13T13:07:35.364-05:00",
                      "updatedBy":"admin",
                      "lastUpdateDate":"2020-01-13T13:07:35.364-05:00",
                      "consolidationInd":4,
                      "lastRowidSystem":"PRODUCT       ",
                      "hubStateInd":1,
                      "label":"E-Scooter",
                      "productType":"Product Group",
                      "productName":"E-Scooter"
                    }
                  },
                  "entityLabel":"E-Scooter",
                  "relationshipLabel":"Product Group is parent of Product Group",
                  "relationship":{
                    "rowidObject":"125           ",
                    "creator":"datasteward",
                    "createDate":"2020-01-13T13:07:36.033-05:00",
                    "updatedBy":"datasteward",
                    "lastUpdateDate":"2020-01-13T13:07:36.034-05:00",
                    "consolidationInd":4,
                    "lastRowidSystem":"PRODUCT       ",
                    "interactionId":"480003",
                    "hubStateInd":0,
                    "label":"Product Group is parent of Product Group",
                    "rowidRelType":"7             ",
```

```
                  "rowidHierarchy":"3             ",
                  "from":{
                    "rowidObject":"166           "
                  },
                  "to":{
                    "rowidObject":"167           "
                  }
                },
                "entityType":"ProductGroup",
                "relationshipType":"ProductGroupIsParentOfProductGroup",
                "object":{
                  "businessEntity":{
                    "Products":{
                      "link":[
                        {
                          "href":"http://localhost:8080/cmx/request/hm_icons/HM_Icons/
BulletSquareBlue/bullet_square_blue.png?ors=localhost-orcl-MDM_SAMPLE",
                          "rel":"icon"
                        }
                      ],
                      "rowidObject":"168           ",
                      "label":"Xiaomi Mi Electric Scooter",
                      "productType":"Product",
                      "productName":"Xiaomi Mi Electric Scooter"
                    }
                  },
                  "entityLabel":"Xiaomi Mi Electric Scooter",
                  "relationshipLabel":"Product Group is parent of Product",
                  "relationship":{
                    "rowidObject":"126           ",
                    "creator":"datasteward",
                    "createDate":"2020-01-13T13:07:36.096-05:00",
                    "updatedBy":"datasteward",
                    "lastUpdateDate":"2020-01-13T13:07:36.096-05:00",
                    "consolidationInd":4,
                    "lastRowidSystem":"PRODUCT       ",
                    "interactionId":"480003",
                    "hubStateInd":0,
                    "label":"Product Group is parent of Product",
                    "rowidRelType":"9             ",
                    "rowidHierarchy":"3             ",
                    "from":{
                      "rowidObject":"167           "
                    },
                    "to":{
                      "rowidObject":"168           "
                    }
                  },
                  "entityType":"Products",
                  "relationshipType":"ProductGroupIsParentOfProduct"
                }
              }
            }
          }
        ]
      }
    ]
}
```


Bulk Relationship Changes

The Bulk Relationship Changes API creates, updates, and deletes relationships in the network and in hierarchies.

The API uses the POST method.

Request URL

The Bulk Relationship Changes URL has the following format:

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

Make the following HTTP POST request to the Get Hierarchy Changes URL:

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

Parameters

Specify the type of change you want to perform and the relationship you want to modify.

The following table describes the parameters in the request body:

Parameter	Description
operation	Type of change you want to perform. Values are `createRelationship`, `updateRelationship`, and `deleteRelationship`.
relationship	Name of the relationship.
rowidObject	Row ID of the object.
startDate	Optional. Specifies the date from which the record is effective. Provide this parameters for a timeline-enabled base object.
endDate	Optional. Specifies the date from which the record is no longer effective. Provide this parameters for a timeline-enabled base object.

Sample API Request

The following request creates new relationships:

```
POST http://localhost:8080/cmx/bulk/localhost-orcl-DS_UI1

{
  "systemName":"Product",
  "rootRowId":"165",
  "items":[
    {
      "operation":"createRelationship",
      "relationship":"ProductGroupIsParentOfProductGroup",
      "payload":{
        "from":{
          "rowidObject":"165"
        },
        "to":{
          "rowidObject":"166"
        }
      }
    },
    {
      "operation":"createRelationship",
      "relationship":"ProductGroupIsParentOfProductGroup",
      "payload":{
        "from":{
          "rowidObject":"166"
        },
        "to":{
          "rowidObject":"167"
        }
      }
    },
```

```
    {
      "operation":"createRelationship",
      "relationship":"ProductGroupIsParentOfProduct",
      "payload":{
        "from":{
          "rowidObject":"167"
        },
        "to":{
          "rowidObject":"168"
        }
      },
      "effectivePeriod":{
        "startDate":"2019-01-01T11:11:53.974-04:00",
        "endDate":"2039-12-31T11:11:53.974-04:00"
      }
    }
  ]
}
```

Sample API Response

The following sample response shows the interaction ID associated to the changes, which you can use to promote and reject changes:

```
{
  "processId":"14068",
  "interactionId":"480003",
  "rejectInteractionId":"480004",
  "items":[
    {
      "payload":{
        "key":{
          "rowid":"124",
          "sourceKey":"SVR1.1G9AG"
        },
        "rowidObject":"124",
        "from":{
          "key":{
            "rowid":"165"
          },
          "rowidObject":"165"
        },
        "to":{
          "key":{
            "rowid":"166"
          },
          "rowidObject":"166"
        }
      }
    },
    {
      "payload":{
        "key":{
          "rowid":"125",
          "sourceKey":"SVR1.1G9AH"
        },
        "rowidObject":"125",
        "from":{
          "key":{
            "rowid":"166"
          },
          "rowidObject":"166"
        },
        "to":{
          "key":{
            "rowid":"167"
          },
          "rowidObject":"167"
        }
```

```
        }
      },
      {
        "payload":{
          "key":{
            "rowid":"126",
            "sourceKey":"SVR1.1G9AI"
          },
          "rowidObject":"126",
          "from":{
            "key":{
              "rowid":"167"
            },
            "rowidObject":"167"
          },
          "to":{
            "key":{
              "rowid":"168"
            },
            "rowidObject":"168"
          }
        }
      }
    ]
}
```

Bulk Promote

The Bulk Promote REST API promotes changes to the network or hierarchy.

The API uses the POST method.

Request URL

The Bulk Promote URL has the following format:

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

Make the following HTTP GET request to the Bulk Promote URL:

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

Sample API Request

The following sample request promotes changes:

```
POST http://localhost:8080/cmx/bulk/localhost-orcl-DS_UI1/promote
{
  "interactionId":"67080007",
  "rejectInteractionId":"67080008",
  "objects":[
    {
      "type":"Hierarchy",
      "name":"HPerson"
    }
  ]
}
```

Sample API Response

The following sample response shows the response after successfully promoting changes:

```
{
    "interactionId": "68520005",
    "rejectInteractionId": "68520006",
    "type": "hierarchy",
```

```
    "name": "HPerson",
    "objects": [
        {
            "before": [
                {
                    "businessEntity": {
                        "Person": {
                            "link": [
                                {
                                    "href": "http://localhost:8080/cmx/request/hm_icons/
hierarchymanager/Person/Person_Small.png?ors=localhost-mdm103-DS_UI2",
                                    "rel": "icon"
                                }
                            ],
                            "rowidObject": "2568254       ",
                            "creator": "admin",
                            "createDate": "2020-04-01T14:38:06.577-04:00",
                            "updatedBy": "admin",
                            "lastUpdateDate": "2020-04-01T14:38:06.577-04:00",
                            "consolidationInd": 4,
                            "lastRowidSystem": "SFA           ",
                            "hubStateInd": 1,
                            "effectivePeriod": {},
                            "label": "LN,FN",
                            "partyType": "Person",
                            "lastName": "LN",
                            "displayName": "FN LN",
                            "generationSuffixCd": "X",
                            "firstName": "FN"
                        }
                    },
                    "entityLabel": "LN,FN",
                    "entityType": "Person",
                    "depth": 0
                }
            ],
            "after": [
                {
                    "businessEntity": {
                        "Person": {
                            "link": [
                                {
                                    "href": "http://localhost:8080/cmx/request/hm_icons/
hierarchymanager/Person/Person_Small.png?ors=localhost-mdm103-DS_UI2",
                                    "rel": "icon"
                                }
                            ],
                            "rowidObject": "2568253       ",
                            "creator": "admin",
                            "createDate": "2020-04-01T14:38:03.956-04:00",
                            "updatedBy": "admin",
                            "lastUpdateDate": "2020-04-01T14:38:03.956-04:00",
                            "consolidationInd": 4,
                            "lastRowidSystem": "SFA           ",
                            "hubStateInd": 1,
                            "label": "LN,FN",
                            "partyType": "Person",
                            "lastName": "LN",
                            "displayName": "FN LN",
                            "generationSuffixCd": "X",
                            "firstName": "FN"
                        }
                    },
                    "entityLabel": "Person",
                    "entityType": "Person",
                    "depth": 1,
                    "object": {
                        "businessEntity": {
                            "Automobile": {
                                "rowidObject": "1450063       ",
                                "creator": "admin",
```

```
                                    "createDate": "2020-04-01T14:38:09.630-04:00",
                                    "updatedBy": "admin",
                                    "lastUpdateDate": "2020-04-01T14:38:09.630-04:00",
                                    "consolidationInd": 4,
                                    "lastRowidSystem": "SFA           ",
                                    "hubStateInd": 1,
                                    "effectivePeriod": {},
                                    "label": "Subaru",
                                    "model": "MODEL",
                                    "drivetrainCode": "4WD",
                                    "doorsCode": "2DR",
                                    "make": "Subaru"
                                }
                            },
                            "entityLabel": "Subaru",
                            "relationshipLabel": "PersonToAutomobile Relationship",
                            "relationship": {
                                "rowidObject": "1450083       ",
                                "creator": "ds1",
                                "createDate": "2020-04-01T14:40:14.921-04:00",
                                "updatedBy": "admin",
                                "lastUpdateDate": "2020-04-01T14:40:23.812-04:00",
                                "consolidationInd": 4,
                                "lastRowidSystem": "SYS0          ",
                                "interactionId": "68520005",
                                "hubStateInd": 1,
                                "label": "PersonToAutomobile Relationship",
                                "from": {
                                    "rowidObject": "2568253       "
                                },
                                "to": {
                                    "rowidObject": "1450063       "
                                }
                            },
                            "entityType": "Automobile",
                            "relationshipType": "PersonToAutomobile2"
                        }
                    },
                    {
                        "businessEntity": {
                            "Person": {
                                "link": [
                                    {
                                        "href": "http://localhost:8080/cmx/request/hm_icons/
hierarchymanager/Person/Person_Small.png?ors=localhost-mdm103-DS_UI2",
                                        "rel": "icon"
                                    }
                                ],
                                "rowidObject": "2568253       ",
                                "creator": "admin",
                                "createDate": "2020-04-01T14:38:03.956-04:00",
                                "updatedBy": "admin",
                                "lastUpdateDate": "2020-04-01T14:38:03.956-04:00",
                                "consolidationInd": 4,
                                "lastRowidSystem": "SFA           ",
                                "hubStateInd": 1,
                                "label": "LN,FN",
                                "partyType": "Person",
                                "lastName": "LN",
                                "displayName": "FN LN",
                                "generationSuffixCd": "X",
                                "firstName": "FN"
                            }
                        },
                        "entityLabel": "Person",
                        "entityType": "Person",
                        "depth": 1,
                        "object": {
                            "businessEntity": {
                                "Person": {
                                    "link": [
```

```
                                    {
                                        "href": "http://localhost:8080/cmx/request/
hm_icons/hierarchymanager/Person/Person_Small.png?ors=localhost-mdm103-DS_UI2",
                                        "rel": "icon"
                                    }
                                ],
                                "rowidObject": "2568254       ",
                                "creator": "admin",
                                "createDate": "2020-04-01T14:38:06.577-04:00",
                                "updatedBy": "admin",
                                "lastUpdateDate": "2020-04-01T14:38:06.577-04:00",
                                "consolidationInd": 4,
                                "lastRowidSystem": "SFA           ",
                                "hubStateInd": 1,
                                "effectivePeriod": {},
                                "label": "LN,FN",
                                "partyType": "Person",
                                "lastName": "LN",
                                "displayName": "FN LN",
                                "generationSuffixCd": "X",
                                "firstName": "FN"
                            }
                        },
                        "entityLabel": "LN,FN",
                        "relationshipLabel": "Person associate Person",
                        "relationship": {
                            "rowidObject": "1634729       ",
                            "creator": "ds1",
                            "createDate": "2020-04-01T14:40:14.870-04:00",
                            "updatedBy": "admin",
                            "lastUpdateDate": "2020-04-01T14:40:23.872-04:00",
                            "consolidationInd": 4,
                            "lastRowidSystem": "SYS0          ",
                            "interactionId": "68520005",
                            "hubStateInd": 1,
                            "label": "Person associate Person",
                            "rowidRelType": "103           ",
                            "rowidHierarchy": "1             ",
                            "note": "note DS",
                            "from": {
                                "rowidObject": "2568253       "
                            },
                            "to": {
                                "rowidObject": "2568254       "
                            }
                        },
                        "entityType": "Person",
                        "relationshipType": "PersonAssociatePerson2"
                    }
                }
            ],
            "approved": [
                {
                    "businessEntity": {
                        "Person": {
                            "link": [
                                {
                                    "href": "http://localhost:8080/cmx/request/hm_icons/
hierarchymanager/Person/Person_Small.png?ors=localhost-mdm103-DS_UI2",
                                    "rel": "icon"
                                }
                            ],
                            "rowidObject": "2568253       ",
                            "creator": "admin",
                            "createDate": "2020-04-01T14:38:03.956-04:00",
                            "updatedBy": "admin",
                            "lastUpdateDate": "2020-04-01T14:38:03.956-04:00",
                            "consolidationInd": 4,
                            "lastRowidSystem": "SFA           ",
                            "hubStateInd": 1,
                            "label": "LN,FN",
```

```
                        "partyType": "Person",
                        "lastName": "LN",
                        "displayName": "FN LN",
                        "generationSuffixCd": "X",
                        "firstName": "FN"
                    }
                },
                "entityLabel": "Person",
                "entityType": "Person",
                "depth": 1,
                "object": {
                    "businessEntity": {
                        "Automobile": {
                            "rowidObject": "1450063       ",
                            "creator": "admin",
                            "createDate": "2020-04-01T14:38:09.630-04:00",
                            "updatedBy": "admin",
                            "lastUpdateDate": "2020-04-01T14:38:09.630-04:00",
                            "consolidationInd": 4,
                            "lastRowidSystem": "SFA           ",
                            "hubStateInd": 1,
                            "effectivePeriod": {},
                            "label": "Subaru",
                            "model": "MODEL",
                            "drivetrainCode": "4WD",
                            "doorsCode": "2DR",
                            "make": "Subaru"
                        }
                    },
                    "entityLabel": "Subaru",
                    "relationshipLabel": "PersonToAutomobile Relationship",
                    "relationship": {
                        "rowidObject": "1450083       ",
                        "creator": "ds1",
                        "createDate": "2020-04-01T14:40:14.921-04:00",
                        "updatedBy": "ds1",
                        "lastUpdateDate": "2020-04-01T14:40:14.922-04:00",
                        "consolidationInd": 4,
                        "lastRowidSystem": "SYS0          ",
                        "interactionId": "68520005",
                        "hubStateInd": 0,
                        "label": "PersonToAutomobile Relationship",
                        "from": {
                            "rowidObject": "2568253       "
                        },
                        "to": {
                            "rowidObject": "1450063       "
                        }
                    },
                    "entityType": "Automobile",
                    "relationshipType": "PersonToAutomobile2"
                }
            },
            {
                "businessEntity": {
                    "Person": {
                        "link": [
                            {
                                "href": "http://localhost:8080/cmx/request/hm_icons/
hierarchymanager/Person/Person_Small.png?ors=localhost-mdm103-DS_UI2",
                                "rel": "icon"
                            }
                        ],
                        "rowidObject": "2568253       ",
                        "creator": "admin",
                        "createDate": "2020-04-01T14:38:03.956-04:00",
                        "updatedBy": "admin",
                        "lastUpdateDate": "2020-04-01T14:38:03.956-04:00",
                        "consolidationInd": 4,
                        "lastRowidSystem": "SFA           ",
                        "hubStateInd": 1,
```

```
                            "label": "LN,FN",
                            "partyType": "Person",
                            "lastName": "LN",
                            "displayName": "FN LN",
                            "generationSuffixCd": "X",
                            "firstName": "FN"
                        }
                    },
                    "entityLabel": "Person",
                    "entityType": "Person",
                    "depth": 1,
                    "object": {
                        "businessEntity": {
                            "Person": {
                                "link": [
                                    {
                                        "href": "http://localhost:8080/cmx/request/
hm_icons/hierarchymanager/Person/Person_Small.png?ors=localhost-mdm103-DS_UI2",
                                        "rel": "icon"
                                    }
                                ],
                                "rowidObject": "2568254       ",
                                "creator": "admin",
                                "createDate": "2020-04-01T14:38:06.577-04:00",
                                "updatedBy": "admin",
                                "lastUpdateDate": "2020-04-01T14:38:06.577-04:00",
                                "consolidationInd": 4,
                                "lastRowidSystem": "SFA           ",
                                "hubStateInd": 1,
                                "effectivePeriod": {},
                                "label": "LN,FN",
                                "partyType": "Person",
                                "lastName": "LN",
                                "displayName": "FN LN",
                                "generationSuffixCd": "X",
                                "firstName": "FN"
                            }
                        },
                        "entityLabel": "LN,FN",
                        "relationshipLabel": "Person associate Person",
                        "relationship": {
                            "rowidObject": "1634729       ",
                            "creator": "ds1",
                            "createDate": "2020-04-01T14:40:14.870-04:00",
                            "updatedBy": "ds1",
                            "lastUpdateDate": "2020-04-01T14:40:14.871-04:00",
                            "consolidationInd": 4,
                            "lastRowidSystem": "SYS0          ",
                            "interactionId": "68520005",
                            "hubStateInd": 0,
                            "label": "Person associate Person",
                            "rowidRelType": "103           ",
                            "rowidHierarchy": "1             ",
                            "note": "note DS",
                            "from": {
                                "rowidObject": "2568253       "
                            },
                            "to": {
                                "rowidObject": "2568254       "
                            }
                        },
                        "entityType": "Person",
                        "relationshipType": "PersonAssociatePerson2"
                    }
                }
            ]
        }
    ]
}
```

Bulk Reject

The Bulk Reject REST API rejects changes to the network or hierarchy.

The API uses the POST method.

Request URL

The Bulk Reject URL has the following format:

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

Make the following HTTP GET request to the Bulk Reject URL:

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

Sample API Request

The following sample request rejects changes:

```
POST http://localhost:8080/cmx/bulk/localhost-orcl-DS_UI1/reject
{
  "interactionId":"67080007",
  "rejectInteractionId":"67080008",
  "objects":[
    {
      "type":"Hierarchy",
      "name":"HPerson"
    }
  ]
}
```

Sample API Response

The following sample shows the response after successfully rejecting changes:

```
{
    "interactionId": "68520003",
    "rejectInteractionId": "68520004",
    "type": "hierarchy",
    "name": "HPerson",
    "objects": [
        {
            "before": [
                {
                    "businessEntity": {
                        "Person": {
                            "link": [
                                {
                                    "href": "http://localhost:8080/cmx/request/hm_icons/
hierarchymanager/Person/Person_Small.png?ors=localhost-mdm103-DS_UI2",
                                    "rel": "icon"
                                }
                            ],
                            "rowidObject": "2568254       ",
                            "creator": "admin",
                            "createDate": "2020-04-01T14:38:06.577-04:00",
                            "updatedBy": "admin",
                            "lastUpdateDate": "2020-04-01T14:38:06.577-04:00",
                            "consolidationInd": 4,
                            "lastRowidSystem": "SFA           ",
                            "hubStateInd": 1,
                            "effectivePeriod": {},
                            "label": "LN,FN",
                            "partyType": "Person",
                            "lastName": "LN",
                            "displayName": "FN LN",
                            "generationSuffixCd": "X",
```

```
                            "firstName": "FN"
                        }
                    },
                    "entityLabel": "LN,FN",
                    "entityType": "Person",
                    "depth": 0
                }
            ],
            "after": [
                {
                    "businessEntity": {
                        "Person": {
                            "link": [
                                {
                                    "href": "http://localhost:8080/cmx/request/hm_icons/
hierarchymanager/Person/Person_Small.png?ors=localhost-mdm103-DS_UI2",
                                    "rel": "icon"
                                }
                            ],
                            "rowidObject": "2568254       ",
                            "creator": "admin",
                            "createDate": "2020-04-01T14:38:06.577-04:00",
                            "updatedBy": "admin",
                            "lastUpdateDate": "2020-04-01T14:38:06.577-04:00",
                            "consolidationInd": 4,
                            "lastRowidSystem": "SFA           ",
                            "hubStateInd": 1,
                            "effectivePeriod": {},
                            "label": "LN,FN",
                            "partyType": "Person",
                            "lastName": "LN",
                            "displayName": "FN LN",
                            "generationSuffixCd": "X",
                            "firstName": "FN"
                        }
                    },
                    "entityLabel": "LN,FN",
                    "entityType": "Person",
                    "depth": 0
                }
            ],
            "approved": [
                {
                    "businessEntity": {
                        "Person": {
                            "link": [
                                {
                                    "href": "/request/hm_icons/hierarchymanager/Person/
Person_Small.png?ors={ors}",
                                    "rel": "icon"
                                }
                            ],
                            "rowidObject": "2568253       ",
                            "creator": "admin",
                            "createDate": "2020-04-01T14:38:03.956-04:00",
                            "updatedBy": "admin",
                            "lastUpdateDate": "2020-04-01T14:38:03.956-04:00",
                            "consolidationInd": 4,
                            "lastRowidSystem": "SFA           ",
                            "hubStateInd": 1,
                            "label": "LN,FN",
                            "partyType": "Person",
                            "lastName": "LN",
                            "displayName": "FN LN",
                            "generationSuffixCd": "X",
                            "firstName": "FN"
                        }
                    },
                    "entityLabel": "Person",
                    "entityType": "Person",
                    "depth": 1,
```

```
                    "object": {
                        "businessEntity": {
                            "Automobile": {
                                "rowidObject": "1450063       ",
                                "creator": "admin",
                                "createDate": "2020-04-01T14:38:09.630-04:00",
                                "updatedBy": "admin",
                                "lastUpdateDate": "2020-04-01T14:38:09.630-04:00",
                                "consolidationInd": 4,
                                "lastRowidSystem": "SFA           ",
                                "hubStateInd": 1,
                                "effectivePeriod": {},
                                "label": "Subaru",
                                "model": "MODEL",
                                "drivetrainCode": "4WD",
                                "doorsCode": "2DR",
                                "make": "Subaru"
                            }
                        },
                        "entityLabel": "Subaru",
                        "relationshipLabel": "PersonToAutomobile Relationship",
                        "relationship": {
                            "rowidObject": "1450082       ",
                            "creator": "ds1",
                            "createDate": "2020-04-01T14:38:13.916-04:00",
                            "updatedBy": "ds1",
                            "lastUpdateDate": "2020-04-01T14:38:13.916-04:00",
                            "consolidationInd": 4,
                            "lastRowidSystem": "SYS0          ",
                            "interactionId": "68520003",
                            "hubStateInd": 0,
                            "label": "PersonToAutomobile Relationship",
                            "from": {
                                "rowidObject": "2568253       "
                            },
                            "to": {
                                "rowidObject": "1450063       "
                            }
                        },
                        "entityType": "Automobile",
                        "relationshipType": "PersonToAutomobile2"
                    }
                },
                {
                    "businessEntity": {
                        "Person": {
                            "link": [
                                {
                                    "href": "/request/hm_icons/hierarchymanager/Person/
Person_Small.png?ors={ors}",
                                    "rel": "icon"
                                }
                            ],
                            "rowidObject": "2568253       ",
                            "creator": "admin",
                            "createDate": "2020-04-01T14:38:03.956-04:00",
                            "updatedBy": "admin",
                            "lastUpdateDate": "2020-04-01T14:38:03.956-04:00",
                            "consolidationInd": 4,
                            "lastRowidSystem": "SFA           ",
                            "hubStateInd": 1,
                            "label": "LN,FN",
                            "partyType": "Person",
                            "lastName": "LN",
                            "displayName": "FN LN",
                            "generationSuffixCd": "X",
                            "firstName": "FN"
                        }
                    },
                    "entityLabel": "Person",
                    "entityType": "Person",
```

```
                    "depth": 1,
                    "object": {
                        "businessEntity": {
                            "Person": {
                                "link": [
                                    {
                                        "href": "/request/hm_icons/hierarchymanager/
Person/Person_Small.png?ors={ors}",
                                        "rel": "icon"
                                    }
                                ],
                                "rowidObject": "2568254       ",
                                "creator": "admin",
                                "createDate": "2020-04-01T14:38:06.577-04:00",
                                "updatedBy": "admin",
                                "lastUpdateDate": "2020-04-01T14:38:06.577-04:00",
                                "consolidationInd": 4,
                                "lastRowidSystem": "SFA           ",
                                "hubStateInd": 1,
                                "effectivePeriod": {},
                                "label": "LN,FN",
                                "partyType": "Person",
                                "lastName": "LN",
                                "displayName": "FN LN",
                                "generationSuffixCd": "X",
                                "firstName": "FN"
                            }
                        },
                        "entityLabel": "LN,FN",
                        "relationshipLabel": "Person associate Person",
                        "relationship": {
                            "rowidObject": "1634728       ",
                            "creator": "ds1",
                            "createDate": "2020-04-01T14:38:13.871-04:00",
                            "updatedBy": "ds1",
                            "lastUpdateDate": "2020-04-01T14:38:13.871-04:00",
                            "consolidationInd": 4,
                            "lastRowidSystem": "SYS0          ",
                            "interactionId": "68520003",
                            "hubStateInd": 0,
                            "label": "Person associate Person",
                            "rowidRelType": "103           ",
                            "rowidHierarchy": "1             ",
                            "note": "note DS",
                            "from": {
                                "rowidObject": "2568253       "
                            },
                            "to": {
                                "rowidObject": "2568254       "
                            }
                        },
                        "entityType": "Person",
                        "relationshipType": "PersonAssociatePerson2"
                    }
                }
            ]
        }
    ]
}
```


Start Match

The Start Match REST API starts a business entity match job. You must import the file with business entities before you call the Start Match API.

The API uses the POST method.

Request URL

The Start Match URL has the following format:

```
http://<host>:<port>/adhocmatch/<orsId>/job
```

Make the following HTTP POST request to the Start Match URL:

```
POST http://<host>:<port>/adhocmatch/<orsId>/job
```

Parameters

Specify the parameters to start a business entity match job.

The following table describes the parameters:

Parameter	Type	Description
orsId	Path	Operational Reference Store database ID to retrieve the matched business entities.
fileId	Body	Optional. The ID of the file you want to import.
mappingId	Body	Optional. Mapping ID of the file you want to import.
matchRuleSet	Body	Optional. Specify the match rule set to apply to the business entity match job.

Sample API Request

The following sample request returns the business entities imported after performing a match:

```
POST /cmx/jobcontrol/localhost-orcl-MDM_SAMPLE/execute/match HTTP/1.1
Content-Type: application/json
Host: localhost:8080
Content-Length: 64

{
  "tableName" : "C_PRODUCT",
  "matchRuleSetName" : "IDL"
}
```

Sample API Response

The API returns a 200 OK response code after matching business entities and successfully importing a file. The response body is empty.

Read Matched Records

The Read Matched Records REST API returns records that match a specified root record. You can review the list of records to determine which records you can merge with the original root record. You can use the Merge Records API to merge the records.

The API uses the GET method.

Request URL

The Read Matched Records URL has the following format:

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

Make the following HTTP GET request to the Read Matched Records URL:

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

Response Body

The response body contains the number of records that match the specified record, the details of the matching records, and a search token. Use the search token to fetch the subsequent pages of the match result.

Sample API Request

The following sample request searches the business entity for records that match a record:

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/1038245?action=matched
```

Sample API Response

The following sample response shows the details of the record that matches the specified record:

```
{
    "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"
        }
    ]
}
```

Update Matched Records

The Update Matched Records REST API creates or updates a record in the match table. The match table contains the pairs of matched records in a business entity after you run a match process on the business entity. Use the API to add records that qualify for a merge with the specified record.

The API uses the PUT method.

Request URL

The Update Matched Records URL has the following format:

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

Make the following HTTP PUT request to the Update Matched Records URL:

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

Add the Content-Type header to specify the media type of the data you want to send with the request:

```
PUT http://<host>:<port>/<context>/<database ID>/<business entity>/<rowId>?action=matched
Content-Type: application/<json/xml>
```

Request Body

Send the list of records that match the specified record in the request body. Use the row ID or the source system and source key to specify the records.

Sample API Request

The following sample adds a record in the match table:

```
PUT http://localhost:8080/cmx/cs/localhost-ORCL-DS_UI1/Person/1038245?action=matched

{
    keys: [
       {
          rowid: "1038246"
       }
   ]
}
```

Sample API Response

The API returns a 200 OK response on successfully creating a record in the match table. The response body is empty.

Delete Matched Records

The Delete Matched Records REST API deletes matched records associated with a root record from the match table.

The API uses the DELETE method.

Request URL

The Delete Matched Records URL has the following format:

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

Make the following HTTP DELETE request to the Delete Matched Records URL:

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

Add the Content-Type header to specify the media type of the data you want to send with the request:

```
DELETE http://<host>:<port>/<context>/<database ID>/<business entity>/<rowId>?
action=matched
Content-Type: application/<json/xml>
```

Request Body

Send the list of records that you want to delete from the match table in the request body.

Sample API Request

The following sample request deletes a record that matches the specified root record from the match table:

```
DELETE http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/1038245?action=matched
```

```
{
    keys: [
       {
          rowid: "1038246"
       }
   ]
}
```

Sample API Response

The API returns a 200 OK response on successfully deleting a record from the match table. The response body is empty.

Get Match Data in CSV

The Get Match Data in CSV REST API retrieves matched and unmatched business entities in CSV format after the system matches records from the source system and the target fields.

The API uses the GET method.

Request URL

The Get Match Data URL has the following format:

```
http://<host>:<port>/adhocmatch/<orsId>/result/{jobGroupControlId}{?
withMatches,withoutMatches}
```

Make the following HTTP GET request to the Get Match Data Results URL:

```
GET http://<host>:<port>/adhocmatch/<orsId>/result/{jobGroupControlId}{?
withMatches,withoutMatches}
```

Parameters

Specify the parameters to return the matched and unmatched business entities in CSV format.

The following table describes the parameters:

Parameter	Type	Description
jobGroupId	Path	Group ID of the batch job.
orsId	Path	Operational Reference Store database ID to retrieve business entity records.

Parameter	Type	Description
withMatches	Query	Optional. Use the following value: True: To return records that match existing data.
withoutMatches	Query	Optional. Use the following value: False: To return records that do not match existing data.

Sample API Request

The following sample request returns matched and unmatched business entities in CSV format:

```
GET /cmx/adhocmatch/{ors}/result/{jobId}?withMatches=true&withoutMatches=true
Accept: application/csv
```

```
{
    item: [
      {
        businessEntity: {
            Person: {
                firstName: "John"
            }
        },
        matchedEntity: {
            {
                businessEntity: {Person: {rowidObject: 123}},
            }
            matchRule: "MRS|1",
            matchScore: 0.9
        }
      },
      {
        businessEntity: {
            Person: {
                firstName: "Alex"
            }
        }
      }
   ]
}
```

Sample API Response

The API returns a 200 OK response code and the CSV file with the match results.

Get Match Data in JSON

The Get Match Data in JSON REST API retrieves matched and unmatched business entities in CSV format after the system matches records from the source system and the target fields.

The API uses the GET method.

Request URL

The Match Results URL has the following format:

```
http://<host>:<port>/adhocmatch/<orsId>/result/{jobGroupControlId}{?
withMatches,withoutMatches}
```

Make the following HTTP GET request to the Match Results URL:

```
GET http://<host>:<port>/adhocmatch/<orsId>/result/{jobGroupControlId}{?
withMatches,withoutMatches}
```

Parameters

Specify the parameters to return the matched and unmatched business entities in CSV format.

The following table describes the parameters:

Parameter	Type	Description
jobGroupId	Path	Group ID of the batch job.
orsId	Path	Operational Reference Store database ID to retrieve business entity records.
withMatches	Query	Optional. Use the following value: True: To return records that match existing data.
withoutMatches	Query	Optional. Use the following value: False: To return records that do not match existing data.

Sample API Request

The following sample request returns the list of matched and unmatched business entities after the system performs a match:

```
GET /cmx/adhocmatch/{ors}/result/{jobId}?withMatches=true&withoutMatches=true
Accept: application/json


{
    item: [
      {
        businessEntity: {
            Person: {
                firstName: "John"
            }
        },
        matchedEntity: {
            {
                businessEntity: {Person: {rowidObject: 123}},
            }
            matchRule: "MRS|1",
            matchScore: 0.9
        }
      },
      {
        businessEntity: {
            Person: {
                firstName: "Alex"
            }
        }
      }
   ]
}
```

Sample API Response

The API returns a 200 OK response code and the JSON file with the match results.

Import Matched File

The Import Matched File REST API matches new business entities with existing business entities before importing the file. After the import operation is completed, the matched business entities are updated with new data.

You can import duplicate business entities, unique business entities or duplicate and unique business entities.

The API uses the POST method.

Request URL

The Import Matched File URL has the following format:

```
http://<host>:<port>/<context>/<orsId>/<aftermatch>
```

Make the following HTTP POST request to the Import Matched File URL:

```
POST http://<host>:<port>/<context>/<orsId>/<aftermatch>
```

Parameters

Specify the parameters to match new business entities with existing business before importing the file.

The following table describes the parameters:

Parameter	Type	Description
orsId	Path	Operational Reference Store database ID to import the file after matching existing records.
filter	Query	Optional. Use the following filter values: withMatches: Enter `true` if the records match existing records. withoutMatches: Enter `false` if the records do not match existing records.
jobGroupControlId	Query	Optional. Match job group ID.
systemName	Query	Optional. Source system name.

Sample API Request

The following sample request matches new business entities with existing business entities before importing the file:

```
POST /cmx/beimport/{ors}/aftermatch
{
    jobGroupControlId: "...",
    systemName: "SFA",
    filter: {
      withMatches: true,
      withoutMatches: false
    }
}
```

Sample API Response

The API returns a 200 OK response code after successfully matching new business entities with existing business entities and importing the file. The response body is empty.

Get Source System Metadata

The Get Source System Metadata REST API returns the metadata of the source systems.

The API uses the GET method.

Request URL

The Get Source System Metadata URL has the following format:

```
http://<host>:<port>/<context>/<orsId>/sourceSystem
```

Make the following HTTP GET request to the Get Source System Metadata URL:

```
GET http://<host>:<port>/<context>/<orsId>/sourceSystem
```

Parameters

Specify the source system you want to return in the response.

The following table describes the parameters in the request body:

Parameter	Type	Description
orsId	Path	Operational Reference Store ID of the database to retrieve the source system.
entityId	Path	ID of the business entity.
entityName	Path	Name of the business entity.
action	Query	The action you want API to perform. The following is the default value: `getSourceSystems`
aggregate	Query	Optional. Aggregate data across multiple XREFs. Use the following values: - RECORD - NODE - ENTITY
children	Query	Optional. List of child node names. If specified, the response contains the child node names.
compact	Query	Optional. specify to return a compact result.
contentMetadata	Query	Optional. Metadata of the list of history events. Use the following values: XREF PENDING_XREF DELETED_XREF HISTORY MATCH BVT TRUST

Parameter	Type	Description
contributorsOnly	Query	Optional. Counts all the XREFs or those XREFs that contribute to the best version of the truth (BVT).
depth	Query	Optional. Number of child levels to return.
describe	Query	Optional. Adds the source system description.
fields	Query	Optional. Comma-separated list of business entity fields. If specified, the response only contains listed fields.
filter	Query	Optional. Filter expression.
firstRecord	Query	Optional. Specifies the first row in the result.
order	Query	Optional. Comma-separated list of field names with an optional prefix of + or –. The prefix + indicates to sort the results in ascending order, and the prefix – indicates to sort the results in descending order. Default is +. When you specify more than one parameter, the result set is ordered by the parameter that is first in the list, followed by the next.
recordStates	Query	Optional. Record states returned in the list of history events. Provide a comma-separated list. You can use the following values: - ACTIVE - PENDING - DELETED
recordstoReturn	Query	Optional. Specifies the number of rows to return.
recordsTotal	Query	Optional. If set to `true`, returns the number of records in the result. Default is `false`

Sample API Request

The following sample request returns the available source system metadata:

```
GET /cmx/metadata/localhost-orcl-MDM_SAMPLE/sourceSystem HTTP/1.1
Host: localhost:8080
```

Sample API Response

The following sample response returns the available source system metadata:

```
HTTP/1.1 200 OK
Server: JBoss-EAP/7
X-Powered-By: Undertow/1
Content-Type: application/json;charset=UTF-8
Content-Length: 328

{
  "totalCount" : 7,
  "firstRecord" : 1,
  "pageSize" : 7,
  "sourceSystems" : [ {
    "name" : "Product"
  }, {
    "name" : "Informatica Data Director"
```

```
    }, {
      "name" : "Legacy"
    }, {
      "name" : "SFA"
    }, {
      "name" : "Lookups"
    }, {
      "name" : "OrgDataInc"
    }, {
      "name" : "Admin"
    } ]
}
```

Get Record History Events

The Get Record History Events REST API returns a list of history events, or groups of history events, associated with a record. Send the record ID in the request body.

The API uses the GET method to return the following data for each group of history events:

- Start and end date for the group
- Number of events in the group

The API returns the following data for each history event:

- History event ID
- Date of the change
- User who made the change
- List of history tables that are affected by the change
- List of record nodes that are affected by the change

Request URL

The Get Record History Events URL has the following format:

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

Make the following HTTP GET request to the Get Record History Events URL:

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

Query Parameters

The ID of the record is a required parameter. The API uses the record ID to find all related history events.

The following table lists the query parameters:

Parameter	Description
startDate and endDate	Optional. Date range for which you want to retrieve the data. If you specify a date range, the response contains only events within this range.
granularity	Optional. The level of detail to group history events. If specified, the response groups history events. Otherwise, the response does not group history events. Use one of the following values: - YEAR - QUARTER - MONTH - WEEK - DAY - HOUR - MINUTE - AUTO
recordStates	Optional. Record states returned in the list of history events. Provide a comma-separated list. You can use the following values: - ACTIVE - PENDING - DELETED
contentMetadata	Optional. Metadata for the list of history events. Provide a comma-separated list. You can use the following values: - XREF - PENDING_XREF - DELETED_XREF - HISTORY - MATCH - BVT - TRUST
children	Optional. Comma-separated list of the child node names. If specified, the response contains the child node names.
order	Optional. Comma-separated list of field names with an optional prefix of + or –. The prefix + indicates to sort the results in ascending order, and the prefix – indicates to sort the results in descending order. Default is +. When you specify more than one parameter, the result set is ordered by the parameter that is first in the list, followed by the next.
fields	Optional. Comma-separated list of business entity fields. If specified, the response only contains listed fields.
filter	Optional. Filter expression.
depth	Optional. Number of child levels to return.
recordsToReturn	Optional. Specifies the number of rows to return.
searchToken	Optional. Specifies the search token returned with previous request.

Parameter	Description
returnTotal	Optional. If set to `true`, returns the number of records in the result. Default is `false`.
firstRecord	Optional. Specifies the first row in the result.
changeType	Optional. Specifies the types of change returned in the result. Provide a comma-separated list. You can use the following values: - BO - XREF - BVT - MERGE - MERGE_AS_SOURCE - MERGE_AS_TARGET - UNMERGE_AS_SOURCE - UNMERGE_AS_TARGET

Related Topics:

- "Formats for Dates and Time in UTC" on page 34

Sample API Request

The following sample request returns all merges, grouped by year, for a record since January 1, 2000:

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/123?
action=listHistoryEvents&startDate=2010-01-01&granularity=YEAR&depth=2&changeType=MERGE
```

Sample API Response

The following sample response lists the merges for the specified record since January 1, 2000:

```
{
    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
    ]
}
```

Get Event Details

The Get Event Details REST API returns details of a specific history event associated with a record. The API returns details such as the type of change made, and the values before and after the change. Send the record ID and history event ID in the request body.

The API uses the GET method.

Request URL

The Get Event Details URL has the following format:

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

Make the following HTTP GET request to the Get Event Details URL:

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

Query Parameters

Before you begin, use the Get Record History Events API to list the history events associated with a record. From the results returned, use the history event ID and the record ID as query parameters.

The following table lists the query parameters:

Parameter	Description
eventID	Required. The Get Record History Events API returns the event IDs for history events.
recordStates	Optional. Record states returned in the list of history events. Provide a comma-separated list. You can use the following values: - ACTIVE - PENDING - DELETED
contentMetadata	Optional. Metadata for the list of history events. Provide a comma-separated list. You can use the following values: - XREF - PENDING_XREF - DELETED_XREF - HISTORY - MATCH - BVT - TRUST
children	Optional. Comma-separated list of child node names. If specified, the response contains the child node names.
order	Optional. Comma-separated list of field names with an optional prefix of + or –. The prefix + indicates to sort the results in ascending order, and the prefix – indicates to sort the results in descending order. Default is +. When you specify more than one parameter, the result set is ordered by the parameter that is first in the list, followed by the next.

Parameter	Description
fields	Optional. Comma-separated list of business entity fields. If specified, the response only contains listed fields.
filter	Optional. Filter expression.
depth	Optional. Number of child levels to return.
recordsToReturn	Optional. Specifies the number of rows to return.
searchToken	Optional. Specifies the search token returned with previous request.
returnTotal	Optional. If set to `true`, returns the number of records in the result. Default is `false`.
firstRecord	Optional. Specifies the first row in the result.

Sample API Request

The following sample request returns information for a history event:

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/123?
action=getHistoryEventDetails&eventId=2016-07-14T02%3A01%3A24.529%2B0000
```

Sample API Response

The following sample response shows the details of the specified event:

```
{
  "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": "SYS0          ",
      "hubStateInd": 1,
      "label": "BE,AC",
      "partyType": "Person",
      "lastName": "BE",
      "displayName": "AC BE",
      "firstName": "AC"
    }
  }
}
```

List Reports

The List Reports REST API returns a list of registered reports and their configuration.

The API uses the GET method.

Request URL

The List Reports REST URL has the following format:

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

Make the following HTTP GET request to the URL:

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

Query Parameters

You can use the `show` parameter to specify whether you want to list all reports or only root reports.

To return all reports, including root reports and drilldown reports, use the following query parameter:

```
list?show=all
```

To return only the root reports, use the following query parameter:

```
list?show=root
```

Note: If you do not specify the `show` parameter, the API returns root reports.

Response Body

The response body contains the list of reports, including the details of each report.

The following table describes the report configuration parameters in the response body:

Parameter	Description
ROWID_RPT_CONFIG	ID of report.
DIMENSION_NAME_1	The name of a dimension of data in the report. For example, a dimension might be User Name, Review task type, or business entity type.
DIMENSION_NAME_2	Optional. The name of a dimension of data in the report. For example, a dimension might be User Name, Review task type, or business entity type.
TIMEPERIOD_NAME	Optional. Name of the period of time to which the data applies. For example, you might specify `Month`.
RPT_NAME	Name of the report.
METRIC_NAME	Label for the type of data collected by the report. By default, the value specified for this parameter appears as the name of the y-axis in the chart. For example, you might have a report that collects information about closed tasks by user. You might use the metric name `Number of closed tasks`.
RPT_DESC	Description of the report.
RPT_TYPE	Indicates whether the report is a drilldown report. If the value is `null`, then the report is a root report. If the value is anything other than `null`, then the report is a drilldown report.

Sample API Request

The following sample request returns all registered reports, including root reports and drilldown reports:

```
GET http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/list?show=all
```

Sample API Response

The following sample response shows a list of registered reports:

```
{
  "ROWID_RPT_CONFIG": "7 ",
  "DIMENSION_NAME_1": "Country",
  "DIMENSION_NAME_2": "Users",
  "TIMEPERIOD_NAME": "Month",
  "RPT_NAME": "Top 10 Countries",
  "METRIC_NAME": "Number Of Customers",
  "RPT_DESC": "Top 10 Countries",
  "RPT_TYPE": "null"
},
{
  "metadata": {
  "ROWID_RPT_CONFIG": "12 ",
  "DIMENSION_NAME_1": "City code",
  "DIMENSION_NAME_2": "null",
  "TIMEPERIOD_NAME": "null",
  "RPT_NAME": "Customer by City",
  "METRIC_NAME": "Customer by City",
  "RPT_DESC": "Customer by City",
  "RPT_TYPE": "null"
}
```

Get Report Configuration and Data

The Get Report Configuration and Data REST API returns the report configuration and data.

The API uses the GET method.

Request URL

The Get Report Configuration and Data REST URL has the following format:

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

Make the following HTTP GET request to the URL:

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

Query Parameters

The report ID is a required URL parameter. Use the report ID to specify the report for which you want the configuration and data.

Response Body

The response includes the report configuration and data.

The following table describes the parameters in the response body:

Parameter	Description
metadata	Includes the report configuration.
ROWID_RPT_CONFIG	ID of report.
DIMENSION_NAME_1	The name of a dimension of data in the report. For example, a dimension might be User Name, Review task type, or business entity type.

Parameter	Description
DIMENSION_NAME_2	Optional. The name of a dimension of data in the report. For example, a dimension might be User Name, Review task type, or business entity type.
TIMEPERIOD_NAME	Optional. Name of the period of time to which the data applies. For example, you might specify `Month`.
RPT_NAME	Name of the report.
METRIC_NAME	Label for the type of data collected by the report. By default, the value specified for this parameter appears as the name of the y-axis in the chart. For example, you might have a report that collects information about closed tasks by user. You might use the metric name `Number of closed tasks`.
RPT_DESC	Description of the report.
RPT_TYPE	Indicates whether the report is a drilldown report. If the value is `null`, then the report is a root report. If the value is anything other than `null`, then the report is a drilldown report.
fieldsMetadata	Lists the fields in a report data array. Values include `DIMENSION_VALUE_1`, `DIMENSION_VALUE_2`, `TIMEPERIOD_VALUE`, `METRIC_VALUE`, and `DRILLDOWN_RPT_ID`.
data	Lists the values for each field in a report data array.

Sample API Request

The following sample request returns the report configuration and data for a report:

```
GET http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/data/MDM.RPT.1
```

Sample API Response

The following sample response shows the report configuration and data for a report:

```
{
  "metadata": {
    "ROWID_RPT_CONFIG": "11 ",
    "DIMENSION_NAME_1": "Country code",
    "DIMENSION_NAME_2": "null",
    "TIMEPERIOD_NAME": "null",
    "RPT_NAME": "Customer by Region",
    "METRIC_NAME": "Customer by Region",
    "RPT_DESC": "Customer by Region",
    "RPT_TYPE": "null",
    "fieldsMetadata": ["DIMENSION_VALUE_1", "DIMENSION_VALUE_2", "TIMEPERIOD_VALUE",
"METRIC_VALUE", "DRILLDOWN_RPT_ID"]
  },
  "data": [
    [
      "CA ",
      "null",
      "null",
      "9",
      "123"
  ],
    [
      "IN ",
      "null",
      "null",
      "19",
      "456"
```

```
  ],
    [
      "US ",
      "United States",
      "null",
      "9",
      "678"
  ],
    [
      "IN ",
      "India ",
      "null",
      "19",
      null
  ]
  ]
}
```

Get Report Configuration and Drilldown Reports

The Get Report Configuration and Drilldown Reports REST API returns the report configuration and associated drilldown reports.

The API uses the GET method.

Request URL

The Get Report Configuration and Drilldown Reports REST URL has the following format:

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

Make the following HTTP GET request to the URL:

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

Query Parameters

The report ID is a required URL parameter. Use the report ID to specify the report for which you want the configuration and associated drilldown reports.

Response Body

The response body includes the following parameters:

Parameter	Description
ROWID_RPT_CONFIG	ID of report.
DIMENSION_NAME_1	The name of a dimension of data in the report. For example, a dimension might be User Name, Review task type, or business entity type.
DIMENSION_NAME_2	Optional. The name of a dimension of data in the report. For example, a dimension might be User Name, Review task type, or business entity type.
TIMEPERIOD_NAME	Optional. Name of the period of time to which the data applies. For example, you might specify `Month`.
RPT_NAME	Name of the report.

Parameter	Description
METRIC_NAME	Label for the type of data collected by the report. By default, the value specified for this parameter appears as the name of the y-axis in the chart. For example, you might have a report that collects information about closed tasks by user. You might use the metric name `Number of closed tasks`.
RPT_DESC	Description of the report.
RPT_TYPE	Indicates whether the report is a drilldown report. If the value is `null`, then the report is a root report. If the value is anything other than `null`, then the report is a drilldown report.
DRILLDOWN	Lists drilldown reports. The order of the listed reports reflects the level of the report relative to the root report.
RPT_NAME	Name of the drilldown report.
DETAILED_RPT_IDS	Lists the IDs of the report details. Includes report detail IDs of the drilldown reports for the drilldown level.
CONFIG_RPT_IDS	Lists the IDs of the drilldown reports for the drilldown level.

Sample API Request

The following sample request returns the report configuration and associated drilldown reports:

```
GET http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/meta/SVR1.2X9N0
```

Sample API Response

The following sample response shows the report configuration and associated drilldown reports:

```
{
  "ROWID_RPT_CONFIG": "7 ",
  "DIMENSION_NAME_1": "Country",
  "DIMENSION_NAME_2": "Users",
  "TIMEPERIOD_NAME": "Month",
  "RPT_NAME": "Top 10 Countries",
  "METRIC_NAME": "Number Of Customers",
  "RPT_DESC": "Top 10 Countries",
  "RPT_TYPE": "null",
  "DRILLDOWN":
  [
    {
      "RPT_NAME": "Customer by region - First level drilldown",
      "DETAILED_RPT_IDS":["1", "2", "3"],
      "CONFIG_RPT_IDS":["1"]
    },
    {

      "RPT_NAME": "Customer by city - Second level drilldown",
      "DETAILED_RPT_IDS":["5", "6", "7", "8", "9", "10"],
      "CONFIG_RPT_IDS":["2"]
    }
  ]
}
```

Register Report

The Register Report REST API registers custom reports. The API returns the report ID for the report, which you can use in other APIs.

The API uses the POST method.

For information about registering the out of the box reports, see "Managing the Out of the Box Reports" on page 342.

Request URL

The Register Report REST URL has the following format:

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

Make the following HTTP POST request to the URL:

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

Request Body

Specify the report configuration.

The following table describes the parameters for the report details in the request body:

Parameter	Description
ROWID_RPT_CONFIG	ID of report.
DIMENSION_NAME_1	The name of a dimension of data in the report. For example, a dimension might be User Name, Review task type, or business entity type.
DIMENSION_NAME_2	Optional. The name of a dimension of data in the report. For example, a dimension might be User Name, Review task type, or business entity type.
TIMEPERIOD_NAME	Optional. Name of the period of time to which the data applies. For example, you might specify `Month`.
RPT_NAME	Name of the report.
METRIC_NAME	Label for the type of data collected by the report. By default, the value specified for this parameter appears as the name of the y-axis in the chart. For example, you might have a report that collects information about closed tasks by user. You might use the metric name `Number of closed tasks`.
RPT_DESC	Description of the report.
RPT_TYPE	Indicates whether the report is a drilldown report. If the value is `null`, then the report is a root report. If the value is anything other than `null`, then the report is a drilldown report.

Sample API Request

The following sample request registers a report:

```
POST http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/list
{
  "DIMENSION_NAME_1": "Country",
  "DIMENSION_NAME_2": "Users",
  "TIMEPERIOD_NAME": "null",
```

```
  "RPT_NAME": "Top 10 Countries",
  "METRIC_NAME": "Number Of Customers",
  "RPT_DESC": "Top 10 Countries"
 }
```

Sample API Response

The following sample shows the response on successfully registering a report:

```
{
  "ROWID_RPT_CONFIG": "SVR1.2X9N0",
  "DIMENSION_NAME_1": "Subject area",
  "DIMENSION_NAME_2": "Source System",
  "TIMEPERIOD_NAME": "Month",
  "RPT_NAME": "randomname750",
  "RPT_DESC": "Source system Metrics",
  "METRIC_NAME": "Number Of Records",
  "RPT_TYPE": null
}
```

The API returns the report ID in the ROWID_RPT_CONFIG parameter.

Update Report Configuration

The Update Report Configuration REST API updates the configuration of a report.

The API uses the POST method.

Request URL

The Update Report Configuration REST URL has the following format:

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

Make the following HTTP POST request to the URL:

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

Query Parameters

The report ID is a required URL parameter. Use the report ID to specify the report for which you want to update the configuration.

Request Body

Specify the report configuration that you want to update.

The following table describes the parameters in the request body:

Parameter	Description
DIMENSION_NAME_1	The name of a dimension of data in the report. For example, a dimension might be User Name, Review task type, or business entity type.
DIMENSION_NAME_2	Optional. The name of a dimension of data in the report. For example, a dimension might be User Name, Review task type, or business entity type.
TIMEPERIOD_NAME	Optional. Name of the period of time to which the data applies. For example, you might specify `Month`.

Parameter	Description
RPT_NAME	Name of the report.
METRIC_NAME	Label for the type of data collected by the report. By default, the value specified for this parameter appears as the name of the y-axis in the chart. For example, you might have a report that collects information about closed tasks by user. You might use the metric name `Number of closed tasks`.
RPT_DESC	Description of the report.

Sample API Request

The following sample request updates the configuration of a report:

```
POST http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/list/SVR1.2X9N0
{
  "DIMENSION_NAME_1": "Country",
  "DIMENSION_NAME_2": "Users",
  "TIMEPERIOD_NAME": "null",
  "RPT_NAME": "Top 10 Countries",
  "METRIC_NAME": "Number Of Customers",
  "RPT_DESC": "Top 10 Countries"
}
```

Sample API Response

The API does not return a response when the report configuration is successfully updated.

Add or Update Report Data

The Add or Update Report Data REST API adds or updates data entries in a report.

Note: If the report contains report data, the API overrides the report data.

The API uses the POST method.

Request URL

The Add or Update Report Data REST URL has the following format:

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

Make the following HTTP POST request to the URL:

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

Query Parameters

The report ID is a required URL parameter. Use the report ID to specify the report for which you want to add or update data entries.

Request Body

Specify the report for which you want to add or update data entries.

The following table describes the parameters for the report data in the request body:

Parameter	Description
DIMENSION_NAME_1	The name of a dimension of data in the report. For example, a dimension might be User Name, Review task type, or business entity type.
DIMENSION_NAME_2	Optional. The name of a dimension of data in the report. For example, a dimension might be User Name, Review task type, or business entity type.
TIMEPERIOD_NAME	Optional. Name of the period of time to which the data applies. For example, you might specify `Month`.
METRIC_VALUE	Number that represents the dimensions of data. For example, you might have a report that collects information about closed tasks by user. One dimension of the report is the user John Smith, and another dimension is the Review task type. The metric value might be `5`. This means that John Smith closed five tasks of the Review task type.
DRILLDOWN_RPT_ID	ID of a drilldown report. If the value is `null`, then there is no drilldown report configured.

Sample API Request

The following sample request adds or updates data entries in a report:

```
POST http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/data/SVR1.2X9N0
[
  {
    "DIMENSION_VALUE_1": "11:Org,ex.Phone",
    "DIMENSION_VALUE_2": "Cross Source Matches",
    "TIMEPERIOD_VALUE": "null",
    "METRIC_VALUE": "5",
    "DRILLDOWN_RPT_ID": null
 },
  {
    "DIMENSION_VALUE_1": "9:Org Name,City,Zip",
    "DIMENSION_VALUE_2": "Cross Source Matches",
    "TIMEPERIOD_VALUE": "null",
    "METRIC_VALUE": "40",
    "DRILLDOWN_RPT_ID": null
 },
 {
    "DIMENSION_VALUE_1": "12:Org,ex.Email",
    "DIMENSION_VALUE_2": "CRM Matches",
    "TIMEPERIOD_VALUE": "null",
    "METRIC_VALUE": "7",
    "DRILLDOWN_RPT_ID": null
 }
]
```

Sample API Response

The API does not return a response when the data entries in the report are successfully added or updated.

Delete Report

The Delete Report REST API deletes a report.

The API uses the DELETE method.

Request URL

The Delete Report REST URL has the following format:

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

Make the following HTTP DELETE request to the URL:

```
DELETE http://<host>:<port>/<context>/<database ID>/data/<report ID>
```

Query Parameters

The report ID is a required URL parameter. Use the report ID to specify the report that you want to delete.

Sample API Request

The following sample request deletes a report:

```
DELETE http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/data/SVR1.2X9N0
```

Sample API Response

The API does not return a response when the report is successfully deleted.

Run Report Update Job

The Run Report Update Job REST API starts the batch job associated with the out of the box report.

The API uses the GET method.

Request URL

The Run Report Update Job REST URL has the following format:

```
http://<host>:<port>/<context>/<database ID>/data/collect/<report ID>
```

Make the following HTTP GET request to the URL:

```
GET http://<host>:<port>/<context>/<database ID>/data/collect/<report ID>
```

Query Parameters

The report ID is a required URL parameter. Use the report ID to specify the report for which you want to run an update job.

Response Body

The response might contain one of the following values in the `status` parameter:

Value	Description
PROCESSING	Job was triggered. When the job completes, the job ID is returned.
COMPLETED_SUCCESSFULLY	Report data was calculated immediately without running a job.

Value	Description
DONE	Job was not started and a calculation was performed in a synchronous mode.
FAILED	Job was not started because of an error. The `errorMessage` and `errorCode` attributes explain the error.

Sample API Request

The following sample request runs an update job for a report:

```
GET http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/data/collect/SVR1.2X9N0
```

Sample API Response

The following sample shows the response on successfully running the update job for a report:

```
{
  "status": "PROCESSING"
}
```

Get Status of Report Update Job

The Get Status of Report Update Job REST API returns the status of a report update job.

The API uses the GET method.

Request URL

The Get Status of Report Update Job REST URL has the following format:

```
http://<host>:<port>/<context>/<database ID>/data/collect/<report ID>/status
```

Make the following HTTP GET request to the URL:

```
GET http://<host>:<port>/<context>/<database ID>/data/collect/<report ID>/status
```

Query Parameters

The ID of the report is a required URL parameter. Use the `report ID` parameter to specify the report for which you want to get the status of the update job.

Response Body

The response contains the following parameters:

Parameter	Description
status	Status of the report update job. Values are `COMPLETED_SUCCESSFULLY`, `PROCESSING`, `FAILED`, or `ERROR`.
lastUpdateDate	Last update date for the report. Defined for report update jobs with the `COMPLETED_SUCCESSFULLY` status.

Parameter	Description
errorCode	Optional. Error code for error with report update job. Defined for report update jobs with the `ERROR` status.
errorMessage	Optional. Error message for error with report update job. Defined for report update jobs with the `ERROR` status.

Sample API Request

The following sample request returns the status of a report update job:

```
GET http://<host>:<port>/<context>/<database ID>/data/collect/SVR1.2X9N1/status
```

Sample API Response

The following sample shows the response on checking the status of a report update job:

```
{
  "status": "COMPLETED_SUCCESSFULLY",
  "jobId": "SVR1.2X9N1 ",
  "lastUpdateDate": "2019-10-31T14:37:11.846-04:00",
  "startRunDate": "2019-10-31T14:37:09.120-04:00"
}
```

List Job Groups

The List Job Groups REST API returns the list of existing job groups and the status. You can filter the jobs by the job group type. For example, you can view all import jobs only.

The API uses the GET method.

Request URL

The Get List of Jobs URL has the following format:

```
http://<host>:<port>/<jobcontrol>/<orsId>/<group?jobGroupType>
```

Make the following HTTP GET request to the Get List of Jobs URL:

```
GET http://<host>:<port>/<jobcontrol>/<orsId>/<group?jobGroupType>
```

Parameters

Specify the parameters to return the list of existing job groups and the status.

The following table describes the parameters:

Parameter	Path	Description
jobGroupControlId	Path	ID of the job group.
orsId	Path	Operational Reference Store database ID to return the list of existing job groups and the status.

Parameter	Path	Description
detailed	Query	Optional. Use one of the following values: True: To view a detailed list of existing job groups and the status. False: To view a brief list of existing job groups and the status.
firstRecord	Query	Optional. The first record in the list of existing job groups.
jobGroupType	Query	Optional. List jobs by the type of job groups. Use one of the following values: Import Match Import_After_Match Find_Replace
groupBy	Query	Optional. Group jobs based on the business entity.
recordsToReturn	Query	Optional. The number of records you want to return.
returnTotal	Query	Optional. Set to True. the total number of records to return.

Sample API Request

The following sample request returns the list of existing job groups and the status:

```
GET /cmx/jobcontrol/localhost-orcl-MDM_SAMPLE/group/SVR1.1G7UY?groupBy=owningTable
HTTP/1.1
Host: localhost:8080
```

Sample API Response

The following sample response returns the list of existing job groups and the status:

```
HTTP/1.1 200 OK
Server: JBoss-EAP/7
Content-Disposition: inline;filename=f.txt
X-Powered-By: Undertow/1
Content-Type: application/json;charset=UTF-8
Content-Length: 1551

{
  "jobGroupControlId" : "SVR1.1G7UY",
  "jobGroup" : {
    "type" : "IMPORT",
    "name" : "BE_Import_RH_LE_PersonView[1]_16f9ffd55b5"
  },
  "startDate" : "2020-01-13T13:06:26.649-05:00",
  "endDate" : "2020-01-13T13:06:40.842-05:00",
  "status" : "COMPLETED_WITH_ERRORS",
  "statusMessage" : "Completed with errors/warnings",
  "errorMessages" : [ "Completed with errors/warnings", "Completed with errors/
warnings", "Completed with errors/warnings", "Completed with errors/warnings",
"Completed with errors/warnings", "Completed with errors/warnings" ],
  "jobItems" : [ {
    "startDate" : "2020-01-13T13:06:36.150-05:00",
    "endDate" : "2020-01-13T13:06:40.808-05:00",
    "status" : "COMPLETED_WITH_ERRORS",
    "owningTable" : "PersonView",
    "errorMessages" : [ "Completed with errors/warnings", "Completed with errors/
```

```
warnings", "Completed with errors/warnings", "Completed with errors/warnings",
"Completed with errors/warnings", "Completed with errors/warnings" ],
    "progress" : 100,
    "metrics" : {
      "200" : 6,
      "300" : 5,
      "201" : 5,
      "301" : 4,
      "202" : 1,
      "302" : 1
    }
  } ],
  "parameters" : {
    "mappingId" : "SVR1.1G7UX",
    "fileName" : "my.pdf",
    "systemName" : "Admin",
    "fileId" : "DB_SVR1.1G7UW"
  },
  "metricLabels" : {
    "200" : "Total lines in file",
    "201" : "Converted lines",
    "300" : "Total records",
    "202" : "Rejected lines",
    "301" : "Loaded records",
    "302" : "Rejected records"
  },
  "progress" : 100
}
```


List a Job Group

The List a Job Group REST API returns data for each job in a batch group.

The API uses the GET method.

Request URL

The List a Job Group URL has the following format:

```
http://<host>:<port>/<jobcontrol>/<orsId>/<groupBy=>
```

Make the following HTTP GET request to the List a Job Group URL:

```
GET http://<host>:<port>/<jobcontrol>/<orsId>/<groupBy=>
```

Parameters

Specify the parameters to return data for each job in a batch group.

The following table describes the parameters in the request body:

Parameter	Description
orsID	Operational reference store ID of the database to retrieve the mapping from.
groupBy	Optional. Groups jobs by the type of job requested.
jobGroupControlId	Specify the batch job group control ID.

Sample API Request

The following sample request returns data for each job in a batch group:

```
GET /cmx/jobcontrol/localhost-orcl-MDM_SAMPLE/group?jobGroupType=IMPORT HTTP/1.1
Host: localhost:8080
```

Sample API Response

The following sample response shows data for each job in a batch group:

```
HTTP/1.1 200 OK
Server: JBoss-EAP/7
X-Powered-By: Undertow/1
Content-Type: application/json;charset=UTF-8
Content-Length: 564

{
  "records" : [ {
    "jobGroupControlId" : "SVR1.1G7UY",
    "startDate" : "2020-01-13T13:06:26.649-05:00",
    "status" : "PENDING",
    "parameters" : { },
    "metricLabels" : { }
  }, {
    "jobGroupControlId" : "SVR1.1G61Y",
    "startDate" : "2020-01-13T12:39:42.152-05:00",
    "endDate" : "2020-01-13T12:40:03.221-05:00",
    "status" : "COMPLETED_WITH_ERRORS",
    "statusMessage" : "Completed with errors/warnings",
    "parameters" : { },
    "metricLabels" : { }
  } ],
  "totalCount" : 2,
  "firstRecord" : 1,
  "pageSize" : 10
}
```

Get DaaS Metadata

The Get DaaS Metadata REST API returns information about a DaaS provider, such as the name, the type, the business entity it works with, and the list of required fields.

The API uses the GET method.

Request URL

The Get DaaS Metadata URL has the following format:

```
http://<host>:<port>/<context>/<database ID>/meta/daas/<providerName>
```

Make the following HTTP GET request to the Get DaaS Metadata URL:

```
GET http://<host>:<port>/<context>/<database ID>/meta/daas/<providerName>
```

Query Parameter

The `providerName` parameter is a required parameter. The parameter is the name of the configured DaaS provider.

Sample API Request

The following sample request returns the metadata information of the `dcp` DaaS provider:

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/meta/daas/dcp
```

The following sample request returns the metadata information of the `ondemand` DaaS provider:

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/meta/daas/ondemand
```

Sample API Response

The following example shows the metadata information of the `dcp` DaaS provider in JSON format:

```
{
                    providerName: "dcp",
                    providerType: "READ",
                    businessEntity: "Organization",
                    systemName: "SFA",
                    requiredFields: [
                                    "dunsNumber"
                    ]
}
```

The following example shows the metadata information of the `ondemand` DaaS provider in JSON format:

```
{
                    providerName: "ondemand",
                    providerType: "SEARCH",
                    businessEntity: "Organization",
                    systemName: "SFA",
                    requiredFields: [
                                "displayName"
                    ]
}
```

DaaS Search

The DaaS Search REST API uses some input fields of a business entity to call an external DaaS service and transforms the response into a list of records.

The API uses the POST method.

Request URL

The DaaS Search URL has the following format:

```
http://<host>:<port>/<context>/<database ID>/daas/search/<daas provider>
```

Make the following HTTP POST request to the DaaS Search URL:

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

Note: In the request body, provide the details of the business entity with the required field.

Request Body

The request body must contain an XML element or a JSON element of the type `Organization` or `OrganizationView` from the `urn:co-ors.informatica.mdm` namespace.

Sample API Request

The following example is a request to the DaaS provider `ondemand` to search for the `organization` business entity with the display name `INFA`:

```
POST http://localhost:8080/cmx/cs/localhost-orcl-MDM_SAMPLE/daas/search/ondemand
{
  "Organization": {
    "displayname": "INFA"
  }
}
```

Sample API Response

The following sample response shows the results of a search that the DaaS provider returns for the organization business entity with the display name `INFA`:


```
{
  "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 Read

The DaaS Read REST API uses some fields from a business entity to request an external DaaS service and transforms the response into a full record.

The API uses the POST method. Specify the required field in the request to the DaaS provider.

Request URL

The DaaS Read URL has the following format:

```
http://<host>:<port>/<context>/<database ID>/daas/read/<daas provider>
```

Make the following POST request to the URL:

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

Note: In the request body, provide the details of the record with the required field.

Query Parameter

The following table lists the query parameter that you can use:

Parameter	Description
logChanges	Optional. If set to true, the resulting record includes the (Service Data Object) SDO change summary which is passed to the Write Business Entity service. Default is false.

Request Body

The request body must contain an XML element or a JSON element of the type `OrganizationView` from the `urn:coors.informatica.mdm` namespace.

Sample API Request

The following example is a request to the DaaS provider `ondemand` to search for the `organization` business entity with the display name `INFA`:

```
POST http://localhost:8080/cmx/cs/localhost-orcl-MDM_SAMPLE/daas/search/ondemand
{
  "Organization": {
    "displayname": "INFA"
  }
}
```

Sample API Response

The following sample response shows the details that the DaaS provider returned for the organization whose D-U-N-S number is `001352574`:

```
{
  "Organization": {
    "displayName": "Infa Lab Inc",
    "dunsNumber": "001352574",
    "TelephoneNumbers": {
      "link": [],
      "item": [
        {
          "phoneNum": "09736250550"
        }
      ]
    },
    "Addresses": {
      "link": [],
      "item": [
        {
          "Address": {
            "cityName": "Rockaway",
            "addressLine1": "11 WALL ST"
          }
        }
      ]
    }
  }
}
```

WriteMerge

The WriteMerge REST API accepts a list of records retrieved from a DaaS provider, persists them in separate XREFs with the appropriate source system, and merges them into a single record. All XREFs belong to the same record.

The API uses the POST method.

Request URL

The WriteMerge URL has the following format:

```
http://<host>:<port>/<context>/<database ID>/daas/write-merge/<business entity name>
```

Make the following HTTP POST request to the WriteMerge URL:

```
POST http://<host>:<port>/<context>/<database ID>/daas/write-merge/<business entity name>
```

Request Body

The request body should contain an XML or JSON element of the type `DaaSEntity.Pager` from the `urn:cobase.informatica.mdm` namespace.

Response Header

When the response is successful, the API returns the interactionId and the processId in the response header.

Sample API Request

The following sample request uses the result of a DaaS search as input to create a record of the Organization business entity type:

```
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"
    }
  ]
}
```

Sample API Response

The following sample response shows the response header and body after successfully merging the list of records into a single record:

```
{
  "Organization": {
    "key": {
      "rowid": "121921",
      "sourceKey": "140000000000"
    },
    "rowidObject": "121921",
    "TelephoneNumbers": {
      "link": [],
      "item": [
        {
          "key": {
            "rowid": "21721",
            "sourceKey": "140000001000"
          },
          "rowidObject": "21721"
        }
      ]
    }
  }
}
```

DaaS Import

The DaaS Import REST API accepts data in an XML format and converts it into a record.

The API uses the POST method.

Request URL

The Import DaaS Data URL has the following format:

```
http://<host>:<port>/<context>/<database ID>/daas/import/
FamilyTreeMemberOrganizationToOrgView
```

Make the following HTTP POST request to the Import DaaS Data URL:

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

Query Parameters

The name of the source system is a required parameter.

The following table lists the parameters that you can use in the request:

Parameter	Description
systemName	Required. Name of the source system which performs the data change.
interactionId	Optional. Interaction ID to assign to all the changes. Usually, the Hub generates the ID when it creates a pending change as the result of a call.
effectivePeriod	Optional. Contains the start date and end date. Specifies the period of time for which the record is effective. Provide these parameters for a timeline-enabled record.
validateOnly	Optional. If set to TRUE, only validation rules are applied to the modified record and the changes are not persisted.
recordState	Optional. Hub state for the created or the updated records. The supported record states are ACTIVE and PENDING.
processId	Optional. ID of the workflow process that contains the task. When a workflow is started as result of the service call, the parameter contains the identifier of the started workflow process.

Related Topics:

- "Formats for Dates and Time in UTC" on page 34

Request Body

The request body must contain an XML or a JSON element of the type `DaaSChangeFamilyTreeMemberOrganizationToOrgView` from the `urn:co-ors.informatica.mdm` namespace.

Response Header

When the response is successful, the API returns the interactionId and the processId in the response header and the record details in the response body.

If the process generates an interaction ID and uses it to create the record, the API returns the interaction ID. If the process starts a workflow instead of directly saving the record to the database, the API returns the process ID which is the ID of the workflow process.

The following example shows a response header with an interaction ID and a process ID:

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

Sample API Request

The request is always in the XML format.

The following sample request shows XML data of the type `ChildAssociation` from the linkage namespace:

```
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>
```

Sample API Response

The following sample response shows the response header and body after successfully importing and creating an Organization business entity:

```
BES-interactionId: 72202100242034
BES-processId: 156048
{
  "Organization": {
    "key": {
      "rowid": "101921",
      "sourceKey": "697557825"
    },
    "rowidObject": "101921"
  }
}
```

DaaS Update

The DaaS Update REST API accepts the data in an XML format before and after a change. The API applies the changes to the record.

The API uses the POST method.

Request URL

The DaaS Update URL has the following format:

```
http://<host>:<port>/<context>/<database ID>/daas/update/
FamilyTreeMemberOrganizationToOrgView
```

Make the following HTTP POST request to the DaaS Update URL:

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

Query Parameters

The name of the source system is a required parameter.

The following table lists the parameters that you can use in the request:

Parameter	Description
systemName	Required. Name of the source system which performs the data change.
interactionId	Optional. Interaction ID to assign to all the changes. Usually, the Hub generates the ID when it creates a pending change as the result of a call.
effectivePeriod	Optional. Contains the start date and end date. Specifies the period of time for which the record is effective. Provide these parameters for a timeline-enabled record.
validateOnly	Optional. If set to TRUE, only validation rules are applied to the modified record and the changes are not persisted.
recordState	Optional. Hub state for the created or the updated records. The supported record states are ACTIVE and PENDING.
processId	Optional. ID of the workflow process that contains the task. When workflow is started as result of the service call, the parameter contains the identifier of the started workflow process.

Related Topics:

- "Formats for Dates and Time in UTC" on page 34

Request Body

The request body must contain an XML or a JSON element of the type `DaaSChangeFamilyTreeMemberOrganizationToOrgView` from the `urn:co-ors.informatica.mdm` namespace.

Response Header

When the response is successful, the API returns the interactionId and the processId in the response header and the record details in the response body.

If the process generates an interaction ID and uses it to create the record, the API returns the interaction ID. If the process starts a workflow instead of directly saving the record to the database, the API returns the process ID which is the ID of the workflow process.

The following example shows a response header with an interaction ID and a process ID:

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

Sample API Request

The API accepts two responses in XML format, one response is before a change and the other response is after the change. In the following request, a new phone number is added to the organization. The `before` XML data does not have the phone number, the `after` XML data has the phone number.

The following request contains the newly added phone number:

```
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>
```

Sample API Response

The following example shows the rowId of the newly created telephone number of the organization:

```
{
  "Organization": {
    "key": {
      "rowid": "101921",
```

```
      "sourceKey": "697557825"
    },
    "rowidObject": "101921",
    "TelephoneNumbers": {
      "link": [],
      "item": [
        {
          "key": {
            "rowid": "1722",
            "sourceKey": "09736250550"
          },
          "rowidObject": "1722"
        }
      ]
    }
  }
}
```

CHAPTER 4

REST APIs for Data Director

You can use REST APIs to rebrand Data Director and configure the login page legal disclaimer.

REST APIs for Rebranding Data Director

You can use REST APIs to rebrand Data Director and customize the look and feel of the application. You can change the default color theme, login background, and logo of the Data Director application.

The following table describes the REST APIs for rebranding Data Director:

REST API	Description
Upload Logo File	Uploads the logo image file. Supports the POST method.
Delete Logo File	Deletes the logo image file. Supports the DELETE method.
Upload Login BG File	Uploads the login page background image file. Supports the POST method.
Delete Login BG File	Deletes the login page background image file. Supports the DELETE method.
Get Variables	Reads the user interface settings, such as the colors of buttons, menus, and text in the Data Director application. Supports the GET method.
Update Variables	Updates the user interface settings, such as the colors of buttons, menus, and text in the Data Director application. Supports the POST method.
Delete Color Schema	Restores the default user interface settings. Supports the DELETE method.

Upload Login BG File

The Upload Login BG File REST API uploads a background image for the Data Director application login page. To specify the file name, add it to the X-File-Name header of the POST request. Send the image data in the REST request body. The Content-Type header value must be `application/octet-stream`.

The API uses the POST method.

The file size of the image must not exceed the value of the `cmx.file.max_file_size_mb` property specified in the `cmxserver.properties` file. The API supports the image file types that the browser supports. For information about the supported browser versions, see the Product Availability Matrix (PAM) at https://network.informatica.com/community/informatica-network/product-availability-matrices.

Request URL

The Upload Login BG File URL has the following format:

```
http://<host>:<port>/<context>/<path>
```

Make the following HTTP POST request to the Update Login BG URL:

```
POST http://<host>:<port>/cmx/ui/theme/login_bg_image
```

Sample API Request

The following sample request uploads the login page background image file:

```
POST http://localhost:8080/cmx/ui/theme/login_bg_image
```

Sample API Response

The API returns a 200 OK response code after successfully uploading a new background image file for the Data Director login page. The response body is empty.

Delete Login BG File

The Delete Login BG File REST API deletes the custom background image from the Data Director application login page.

The API uses the DELETE method.

After you delete the custom background image, the login page displays the default background image.

Request URL

The Delete Login BG File URL has the following format:

```
http://<host>:<port>/<context>/<path>
```

Make the following HTTP DELETE request to the Delete Login BG URL:

```
DELETE http://<host>:<port>/cmx/ui/theme/login_bg_image
```

Sample API Request

The following sample request deletes the background image from the Data Director application login page:

```
DELETE http://localhost:8080/cmx/ui/theme/login_bg_image
```

Sample API Response

The API returns a 200 OK response code after successfully deleting the background image from the Data Director application login page. The response body is empty.

Upload Logo File

The Upload Logo File REST API uploads a new logo image for the Data Director application. To specify the file name, add it to the X-File-Name header of the POST request. Send the image data in the REST request body. The Content-Type header value must be `application/octet-stream`.

The API uses the POST method.

The file size of the image cannot exceed 130 pixels in width and 33 pixels in height. The API supports the image file types that the browser supports. For information about the supported browser versions, see the Product Availability Matrix (PAM) at https://network.informatica.com/community/informatica-network/product-availability-matrices.

Request URL

The Upload Logo File URL has the following format:

```
http://<host>:<port>/<context>/<path>
```

Make the following HTTP POST request to the Upload Logo File URL:

```
POST http://localhost:8080/cmx/ui/theme/logo
```

Sample API Request

The following sample request uploads the application logo image:

```
POST http://localhost:8080/cmx/ui/theme/logo HTTP/1.1
Host: localhost:8080
X-File-Name: logo_dockers.jpg
Content-Type: application/octet-stream
```

Sample API Response

The API returns a 200 OK response code after successfully uploading the logo image. The response body is empty.

Delete Logo File

The Delete Logo File REST API deletes the custom logo image file from the application.

The API uses the DELETE method.

Note: After you delete the custom logo image, the login page displays the default logo image.

Request URL

The Delete Logo File URL has the following format:

```
http://<host>:<port>/<context>/<path>
```

Make the following HTTP DELETE request to the Delete Logo File URL:

```
DELETE http://<host>:<port>/cmx/ui/theme/logo
```

Sample API Request URL

The following sample request deletes the Data Director logo file:

```
DELETE http://localhost:8080/cmx/ui/theme/logo
```

Sample API Response

The API returns a 200 OK response code after successfully deleting the logo image file. The response body is empty.

Get Variables

The Get Variables REST API returns the current color schema properties of Data Director.

The API uses the GET method.

Request URL

The Get Variables URL has the following format:

```
http://<host>:<port>/<context>/<path>
```

Make the following HTTP GET request to the Get Variables URL:

```
GET http://<host>:<port>/cmx/ui/theme/variables
```

Sample API Request

The following request returns the properties of the current color theme.

```
GET http://localhost:8080/cmx/ui/theme/variables
```

Sample API Response

The API returns a 200 OK response code after successfully retrieving the color schema properties.


```
    "variables": {
        "menu_text_color": "black",
        "dropdown_menu_item_checked_hover_bg_color": "Gold",
        "button_default_selected_border_color": "blue",
        "dropdown_menu_item_checked_hover_text_color": "Coral",
        "button_success_text_color": "white",
        "login_form_text_color": "green",
        "menu_hover_text_color": "#07fc03",
        "dropdown_menu_item_checked_text_color": "LightSalmon",
        "button_danger_text_color": "black",
        "menu_hover_bg_color": "#00349D",
        "header_search_text_color": "#36270a",
        "button_default_border_color": "red",
        "dropdown_menu_item_text_color": "#daf97b",
        "button_danger_bg_color": "pink",
        "button_default_bg_color": "yellow",
        "menu_sel_text_color": "#26cfe0",
        "button_success_border_color": "olive",
        "dropdown_menu_text_color": "#945d29",
        "dropdown_menu_bg_color": "#a4c7f5",
        "primary_button_text_color": "white",
        "header_search_bg_color": "#a4c7f5",
        "dropdown_menu_item_checked_bg_color": "LightGoldenrodYellow",
        "button_default_selected_bg_color": "orange",
        "primary_button_bg_color": "green",
        "entity_view_label_font_wight": "bold",
        "primary_button_selected_bg_color": "pink",
        "primary_button_selected_border_color": "red",
        "header_text_color": "#36270a",
        "button_danger_selected_text_color": "green",
        "primary_button_border_color": "white",
        "menu_sel_bg_color": "#636260",
        "button_success_selected_border_color": "red",
        "login_form_background_color": "#fcd483",
        "dropdown_menu_item_hover_text_color": "OrangeRed",
        "header_background_color": "#c4922d",
        "button_success_selected_bg_color": "PaleVioletRed",
        "button_danger_border_color": "purple",
        "button_default_text_color": "red",
        "button_success_bg_color": "orange",
        "button_danger_selected_bg_color": "yellow",
        "button_danger_selected_border_color": "red",
        "primary_button_selected_text_color": "red",
        "workspace_bg": "#cadee0",
        "button_default_selected_text_color": "blue",
        "dropdown_menu_item_hover_bg_color": "RoyalBlue",
        "menu_background_color": "#cdc1bb",
        "button_success_selected_text_color": "black"
    }

}
```


For more information about the color schema properties of Data Director, see " Default Color Schema" on page 258.

Update Variables

The Update Variables REST API updates the current color properties of Data Director. Use the JSON format to send data in the request body.

The API uses the POST method.
Note: The API request might not update the color properties consistently across the application. For example, after you submit the request to update the color properties of buttons and menus, the Save button might differ on different screens and some menus might change only their background color.

Request URL

The Update Variables URL has the following format:

```
http://<host>:<port>/<context>/<path>
```

Make the following HTTP POST request to the Update Variables URL:

```
POST http://<host>:<port>/cmx/ui/theme/variables
```

Sample API Requests

The following request updates all the color properties of the Data Director application:

```
POST http://localhost:8080/cmx/ui/theme/variables

{
  "variables": {
    "menu_text_color": "#FFF",
    "button_default_selected_border_color": "orange",
    "button_success_text_color": "white",
    "login_form_text_color": "red",
    "button_danger_text_color": "black",
    "menu_hover_bg_color": "#00349D",
    "button_default_border_color": "yellow",
    "dropdown_menu_item_text_color": "white",
    "button_danger_bg_color": "pink",
    "button_default_bg_color": "yellow",
    "button_success_border_color": "olive",
    "dropdown_menu_text_color": "white",
    "dropdown_menu_bg_color": "olive",
    "primary_button_text_color": "white",
    "button_default_selected_bg_color": "orange",
    "primary_button_bg_color": "green",
    "entity_view_label_font_wight": "normal",
    "primary_button_selected_bg_color": "silver",
    "primary_button_selected_border_color": "black",
    "header_text_color": "#444444",
    "button_danger_selected_text_color": "black",
    "primary_button_border_color": "green",
    "menu_sel_bg_color": "#001E60",
    "button_success_selected_border_color": "black",
    "login_form_background_color": "#E5EDF8",
    "header_background_color": "#ffd900",
    "button_success_selected_bg_color": "silver",
    "button_danger_border_color": "pink",
    "button_default_text_color": "black",
    "button_success_bg_color": "olive",
    "button_danger_selected_bg_color": "orange",
    "button_danger_selected_border_color": "red",
    "primary_button_selected_text_color": "black",
    "workspace_bg": "#E5EDF8",
    "button_default_selected_text_color": "black",
    "dropdown_menu_item_hover_bg_color": "navy",
    "menu_background_color": "#004fb6",
    "button_success_selected_text_color": "black"
  }
}
```

The following request updates the color properties of the primary buttons in the Data Director application.

```
{
  "variables":{
     "primary_button_text_color": "white",
        "primary_button_bg_color": "green",
        "primary_button_border_color": "white",
        "primary_button_selected_bg_color": "pink",
        "primary_button_selected_border_color": "red",
        "primary_button_selected_text_color": "red"
    }
}
```

Sample API Response

The API returns a 200 OK response code after successfully updating the specified color properties. The response body is empty.

Default Color Schema

Depending on the needs of your organization, you can change the default color properties of the Data Director application. Some properties affect a specific Data Director user interface component. Other properties, such as the button properties, affect multiple user interface components.

Login Page

The following table lists the properties you can change for the login page:

Property	Description	Default
login_form_text_color	Text color of the field labels on the login page.	#333333
login_form_background_color	Background color of the login page.	#fff

Page Header

The following table lists the properties you can change for the Data Director page header:

Property	Description	Default
header_text_color	Text color of the page header.	#fff
header_search_text_color	Color of the business entity names that appear in the page header to the left of the search field.	#000
header_background_color	Background color of the page header.	#000

Navigation Bar

The following table lists the properties you can change for the left navigation bar:

Property	Description	Default
menu_text_color	Text color of a navigation bar item that the user did not select or hover over.	#fff
menu_sel_text_color	Text color of a selected navigation bar item.	#fff
menu_hover_text_color	Text color of a navigation bar item that the user hovers over.	#fff
menu_background_color	Background text color of a navigation bar item that the user did not select or hover over.	#333
menu_hover_bg_color	Background color of navigation bar item that the user hovers over.	#000
menu_sel_bg_color	Background color of a selected navigation bar item.	#000

Workspaces

The following table lists the properties you can change for a workspace or the business entity record view:

Property	Description	Default
workspace_bg	Background color of the workspace that appears to the right of the navigation bar.	#fff
entity_view_label_font_weight	Font weight of the field labels in the read-only view.	Normal

Drop-down Lists

The following table lists the properties you can change for the drop-down lists in the application:

Property	Description	Default
dropdown_menu_text_color	Text color of the list items.	#000
dropdown_menu_bg_color	Background color of a drop-down list.	#fff

Property	Description	Default
dropdown_menu_item_hover_text_color	Text color of a list item that the user selects or hovers over.	#2f3342
dropdown_menu_item_hover_bg_color	Background color of a list item that the user selects or hovers over.	#fafbfc

Application Buttons

The buttons in the application belong to the following categories:

- 1. Buttons that might cause loss of data when you cancel an operation. For example, the **Cancel Edit** button in the business entity record views.
- 2. Buttons that submit or save data. For example, the **Edit** or **Save** button in the business entity record views.
- 3. Buttons that perform the primary action in the user interface. For example, the **Back to Form** button.
- 4. Default buttons. Buttons that you find in most of the pages and dialog boxes. For example, the **OK** button.

The following table lists the button styles and properties you can change:

Button Style	Property	Description	Default
1	button_danger_text_color	Text color on a button.	#3df
1	button_danger_bg_color	Background color of a button.	#ddd
1	button_danger_border_color	Border color of a button.	#2373e4
1	button_danger_selected_border_color	Border color of a button that the user selects or hovers over.	#2373e4
1	button_danger_selected_text_color	Text color on a button that the user selects or hovers over.	#3df
1	button_danger_selected_bg_color	Background color of a button that the user selects or hovers over.	#ddd
2	button_success_text_color	Text color on a button.	#3df
2	button_success_bg_color	Background color of a button.	#ddd
2	button_success_border_color	Border color of a button.	#2373e4
2	button_success_selected_text_color	Text color on a button that the user selects or hovers over.	#3df
2	button_success_selected_bg_color	Background color of a button that the user selects or hovers over.	#ddd
2	button_success_selected_border_color	Border color of a button that the user selects or hovers over.	#2373e4

Button Style	Property	Description	Default
3	primary_button_text_color	Text color on a button.	#fff
3	primary_button_bg_color	Background color of a button.	#4b91ea
3	primary_button_border_color	Border color of a button.	#3483e7
3	primary_button_selected_text_color	Text color on a button that the user selects or hovers over.	#fff
3	primary_button_selected_bg_color	Background color of a button that the user selects or hovers over.	#4b91ea
3	primary_button_selected_border_color	Border color of a button when the user does not select or hover over it.	#3483e7
4	button_default_text_color	Text color on a button.	#333
4	button_default_bg_color	Background color of a button.	#fff
4	button_default_border_color	Border color of a button.	#eaeaea
4	button_default_selected_text_color	Text color on a button that the user selects or hovers over.	#333
4	button_default_selected_bg_color	Background color of a button that the user selects or hovers over.	#fff
4	button_default_selected_border_color	Border color of a button that the user selects or hovers over.	#eaeaea

Delete Color Schema

The Delete Color Schema REST API deletes the custom color properties and restores the default color properties in the Data Director application.

The API uses the DELETE method.

Request URL

The Delete Color Schema URL has the following format:

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

Make the following HTTP DELETE request to the Delete Color Schema URL:

```
DELETE http://<host>:<port>/cmx/ui/theme
```

Sample API Request

The following sample request deletes the custom color properties and restores the default properties:

```
DELETE http://localhost:8080/cmx/ui/theme
```

The request body is empty.

Sample API Response

The API returns a 200 OK response code after successfully deleting the current color properties from the Data Director application and restoring the default properties. The response body is empty.

REST APIs for Configuring a Legal Disclaimer

You can use REST APIs to configure the login page legal disclaimer for Data Director, the Provisioning tool, and the Hub Console. For example, you can describe permitted uses of the applications and the security policies enforced by your organization. The user must acknowledge the legal disclaimer before logging in.

The following table lists the REST APIs for configuring the login page legal disclaimer:

REST API	Description
Read Legal Message Configuration	Reads the legal disclaimer file information. For example, retrieves the current legal disclaimer. Supports the GET method.
Set Legal Message Configuration	Creates or updates a legal disclaimer file. For example, updates the current legal disclaimer. Supports the POST method.
Delete Legal Message Configuration	Deletes a legal disclaimer file. For example, deletes the current legal disclaimer. Supports the DELETE method.

Legal Disclaimer Parameters

You can specify the legal disclaimer text, title, and whether the disclaimer appears each time the user tries to log in. You can also specify the applications that display the legal disclaimer on the login page.

The following table lists the legal disclaimer parameters that you can specify:

Parameter	Description
messageContent	The text of the legal disclaimer.
messageTitle	The title of the legal disclaimer.
acceptLabel	The text on the button that the user clicks to accept the legal disclaimer.
rejectLabel	The text on the button that the user clicks to reject the legal disclaimer. The user cannot log in without accepting the disclaimer. If the user rejects the legal disclaimer, the Login screen reappears.
showOnce	Specifies that the legal disclaimer appears when the user logs in for the first time. Default is `true`. Set the value to `false` if you want the legal disclaimer to appear each time the user tries to log in.
enabledApplications	The applications that display the legal disclaimer when the user tries to log in. You can specify the following values: - e360. Specifies the Data Director application. - provisioning. Specifies the Provisioning tool. - hubconsole. Specifies the Hub Console.

Read Legal Message Configuration

The Read Legal Message Configuration REST API returns the current login page legal disclaimer.

The API uses the GET method.

Request URL

The Read Legal Message Configuration URL has the following format:

```
http://<host>:<port>/<context>/<path>
```

Make the following HTTP GET request to the Read Legal Message Configuration URL:

```
GET http://localhost:8080/cmx/ui/legal/config
```

Sample API Request

The following request returns the properties of the current color theme.

```
GET http://localhost:8080/cmx/ui/legal/config
Content-Type: application/json
```

Sample API Response

The API returns a 200 OK response code after successfully retrieving the current login page legal disclaimer.

For example, you might receive the following response:

```
{
   "messageContent":"The use of this systems may be monitored and recorded. If you
access this system, you expressly consent to such monitoring and the use of the
security policies enforced by the company.",
   "messageTitle":"Legal Disclaimer",
   "acceptLabel":"Accept",
   "rejectLabel":"Reject",
   "showOnce":true,
   "enabledApplications":[
      "e360",
      "provisioning",
      "hubconsole"
   ]
}
```

Set Legal Message Configuration

The Set Legal Message Configuration REST API creates or updates the login page legal disclaimer. Use the JSON format to send data in the request body.

The API uses the POST method.

Request URL

The Set Legal Message Configuration URL has the following format:

```
http://<host>:<port>/<context>/<path>
```

Make the following HTTP POST request to the Set Legal Message Configuration URL:

```
POST http://localhost:8080/cmx/ui/legal/config
```

Sample API Request

The following request body contains a sample legal disclaimer:

```
{
    "messageContent": "The use of this systems may be monitored and recorded. If you
access this system, you expressly consent to such monitoring and the use of the
security policies enforced by the company.",
    "messageTitle": "Legal Disclaimer",
    "acceptLabel": "Accept",
    "rejectLabel": "Reject",
    "showOnce": true,
```

```
    "enabledApplications": [
        "e360",
        "provisioning",
        "hubconsole"
    ]
}
```

Sample API Responses

The API returns a 200 OK response code after successfully creating or updating the login page legal disclaimer. The response body is empty.

Delete Legal Message Configuration

The Delete Legal Message Configuration REST API deletes the configuration settings of the login page legal disclaimer.

The API uses the DELETE method.

Request URL

The Delete Legal Message Configuration URL has the following format:

```
http://<host>:<port>/<context>/<path>
```

Make the following HTTP DELETE request to the Delete Legal Message Configuration URL:

```
DELETE http://<host>:<port>/cmx/ui/legal/config
```

Sample API Request

The following sample request deletes the legal disclaimer configuration for the login page:

```
DELETE http://localhost:8080/cmx/ui/legal/config
```

Sample API Response

The user can log in using the standard login procedure. The API returns a 200 OK response code after successfully deleting the legal disclaimer configuration for the login page. The response body is empty.

CHAPTER 5

SOAP Business Entity Service Calls

This chapter includes the following topics:

SOAP Calls for Business Entity Services Overview

Simple Object Access Protocol (SOAP) endpoint calls make all business entity services available as web services. You can make SOAP calls to create, delete, update, and search for records in a business entity. You can perform operations, such as merge, unmerge, and match records. You can also make SOAP calls to create, update, and search for tasks and perform tasks.

The SOAP endpoints for business entity services use the Web Services Security (WS-Security) UsernameToken to authenticate users.

Note: Before you use the SOAP APIs to call the business entity services, validate the Operational Reference Store.

Authentication method

All SOAP calls to the business entity services require user authentication. Provide the user name and password in the SOAP message header of the web service request.

The SOAP header element Security contains the security-related data. The Security element contains the UsernameToken element which has the following child elements:

username

User name associated with the token.

password

Password for the user name associated with the token.

Send the user name and password in the UsernameToken element.

The following example shows the Security header element in the SOAP message:

```
<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>
```

Authentication Cookies for Login from Third-Party Applications

Use authentication cookies to authenticate MDM Hub users and call business entity services from third-party applications, such as Postman. With the use of authentication cookies, you need not hard-code the user name and password.

The process for using authentication cookies includes the following steps:

1. Use the credentials of an authenticated user to get an authentication cookie.
2. Save the authentication cookie.
3. Use the authentication cookie to call SOAP APIs.

All MDM sessions use an Informatica CSRF Token (ICT). An ICT appears in the HTTP header.

When a user requests a session ID, the response returns an ICT as part of the HTTP header. For external applications, there is a URL for authentication that the external applications can use. This URL returns an authentication cookie and an ICT that is in the header. The authentication cookie is an MDM session ID.

For any subsequent business entity services calls, the call must include the authentication cookie and the ICT. Subsequent requests can use the ICT and MDM session ID until the session expires.

Get the Authentication Cookie and Informatica CSRF Token

The following example shows how to get the authentication cookie and the Informatica CSRF Token (ICT):

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

Authorization:   No Auth

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

The following example shows how the ICT appears in the header:

```
ICT:  0ffa3b95c67e111b9c14c140a70273a15db266993d0c763d555135de4c4d6110
```

The following example shows how the authentication cookie appears in the response:

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

SOAP Example with Authentication Cookie and Informatica CSRF Token

The following example shows how to use an authentication cookie and Informatica CSRF Token (ICT) in your API request header:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:cs-ors.informatica.mdm">
  <soapenv:Header/>
   <soapenv:Body>
      <urn:listAssignableUsers>
         <!--Optional:-->
         <urn:parameters>
            <!--Optional:-->
            <urn:taskType>Update</urn:taskType>
            <!--Optional:-->
            <urn:businessEntity>Person</urn:businessEntity>
         </urn:parameters>
      </urn:listAssignableUsers>
   </soapenv:Body>
</soapenv:Envelope>

Headers:
Cookie:  mdmsessionid: vppOTOhJLKMjYJf7Diil
ICT:  0ffa3b95c67e111b9c14c140a70273a15db266993d0c763d555135de4c4d6110
```

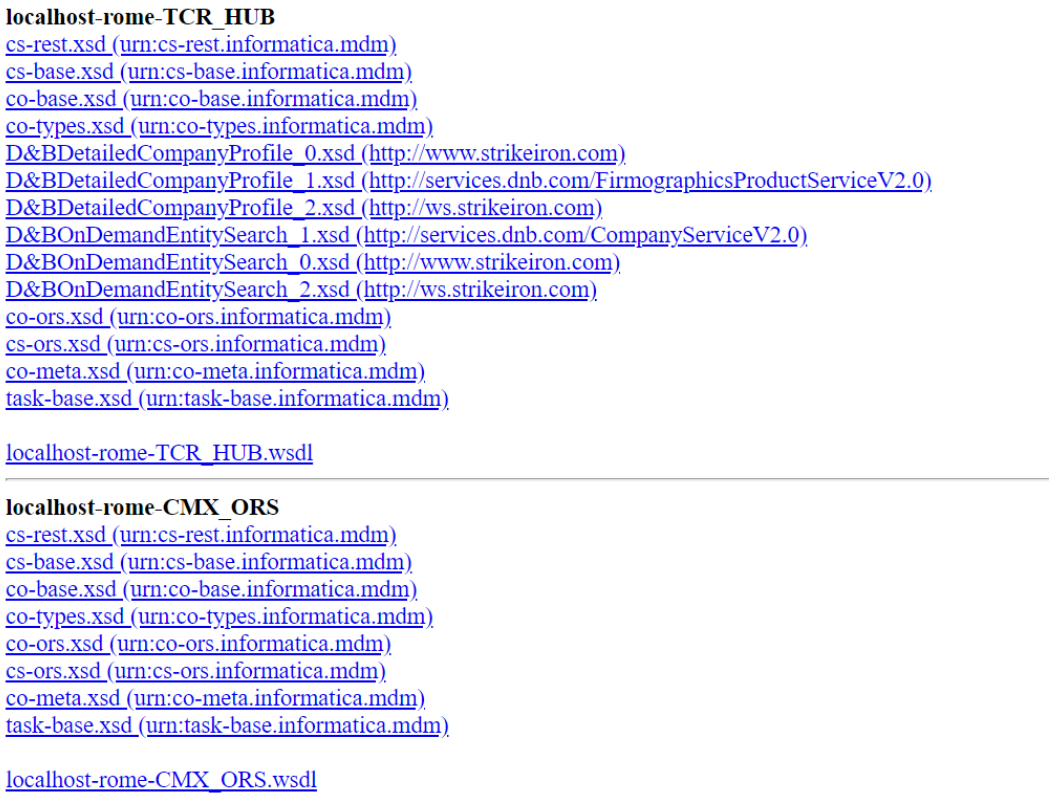
Web Services Description Language File

Web Services Description Language (WSDL) files contain the XML descriptions of the web services, formats of the SOAP requests and responses, and all parameters. The MDM Hub generates a WSDL file for each Operational Reference Store.

The WSDL files for each Operational Reference Store are in the following location:

```
http:://<host>:<port>/cmx/csfiles
```

The following image shows the location where you can download the WSDL file for the Operational Reference Stores:



Click the link to download the WSDL file for the available Operational Reference Stores.

SOAP URL

A SOAP URL is the address you use to connect to the SOAP server.

A SOAP URL has the following syntax:

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

The URL has the following fields:

host

The host that runs the database.

port

Port number that the database listener uses.

context

The context is always `cmx/services/BEServices`.

database ID

ID of the ORS as registered in the Databases tool in the Hub Console.

The following example shows a SOAP URL:

```
http://localhost:8080/cmx/services/BEServices/localhost-orcl-DS_UI1
```

SOAP Requests and Responses

Use the SOAP XML message format to send requests through a SOAP client to the business entity service and to receive responses from the business entity service to the client. The SOAP request and response format is the same.

A SOAP message contains the following elements:

Envelope

Defines the start and the end of the message.

Header

Optional. Contains additional attributes, such as authentication details for processing the message. If the Header element is present, it must be the first child element of the Envelope element.

Body

Contains the XML data that the client or the web service processes.

A SOAP message has the following format:

```
<?xml version="1.0"?>
<env:Envelope  xmlns:env="http://schemas.xmlsoap.org/soap/envelope/" >

<env:Header>
</env:Header>

<env:Body>
</env:Body>

</env:Envelope>
```

A SOAP request has the following format:

```
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>
```

A SOAP response has the following format:

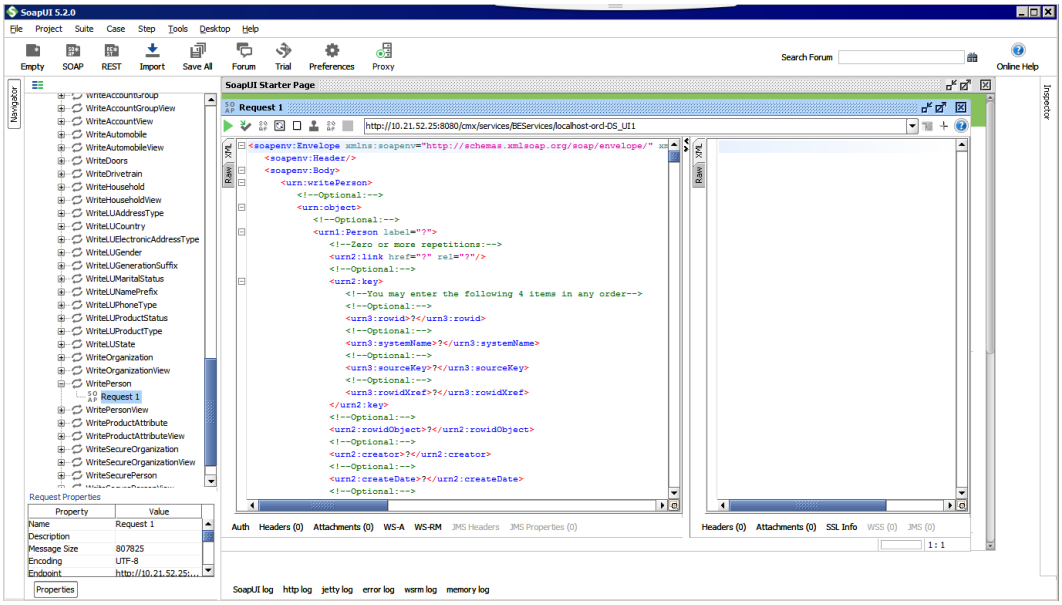
```
HTTP/1.0 200 OK
Content-Type: text/xml; charset=utf-8

<?xml version="1.0"?>
<env:Envelope  xmlns:env="http://schemas.xmlsoap.org/soap/envelope/" >

<env:Header>
</env:Header>

<env:Body>
</env:Body>

</env:Envelope>
```

Viewing Input and Output Parameters

You can use a functional testing tool, such as SoapUI, to view SOAP API input and output parameters.

Create a SOAP project and import the WSDL file into the project. The operations you can perform using the business entity services appear as nodes in SoapUI. Each operation has a request and a response message format. SoapUI creates a sample request for each operation when you import the WSDL file.

Open the project and double-click a request to open the request editor. The following image shows the input parameters in SoapUI for the WritePerson SOAP API:



SOAP API Reference

The SOAP API reference for business entity services lists the SOAP APIs and provides a description for each API. Also refer to the WSDL file for descriptions of the business entity services.

Use the following SOAP APIs to perform operations on business entities:

Get Metadata

Returns the data structure of a business entity.

List Record

Returns the list of lookup values or foreign key values.

Read Record

Returns the details of a root record in the business entity.

Create Record

Creates a record in the specified business entity.

Update Record

Updates the specified root record and its child records.

Delete Record

Deletes a root record in a business entity.

Search Record

Searches a string value in a searchable root business entity and returns the root records that match the search criteria.

Preview Promote

Returns a preview of a resulting record if you promote pending changes based on the interaction ID of the change request.

Promote

Promotes all pending changes made to a record based on the interaction ID of the change request.

Delete Promote

Deletes all pending changes that you make to a record based on the interaction ID of the change request.

Preview Merge

Returns a preview of a consolidated root record if you merge two or more root records.

Merge Records

Merges two or more root records to create a single consolidated record.

Unmerge Records

Unmerges a root record into individual root records that existed before the records were merged.

Get Related Records

Returns a list of related records based on the relationships that you configure in the Hierarchy Manager.

Read Matched Records

Returns records that match a specified root record.

Update Matched Records

Creates or updates a record in the match table.

Delete Matched Records

Deletes matched records from the match table.

Get BPM Metadata

Returns the task types and two indicators that specify whether the workflow adapter can perform the Get Task Lineage service and the administration services.

List Tasks

Returns a list of workflow tasks.

Read Task

Returns the details of a task.

Create Task

Creates a task and starts a workflow.

Update Task

Updates a single task.

Execute Task Action

Performs a task action and sets the task back to the workflow for further processing.

List Assignable Users

Returns a list of users to whom you can assign the tasks that belong to a task type.

Task Complete

Closes a task workflow after you complete all the tasks in the workflow.

Sample SOAP Requests and Responses

To understand SOAP APIs, review the following sample SOAP requests and responses.

List Assignable Users Sample

The following sample SOAP request retrieves a list of assignable users:

```
<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>
```

The following sample SOAP response includes a list of assignable users:

```
<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>
```

Read Business Entity Sample

The following sample SOAP request retrieves information for a Person business entity:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:cs-ors.informatica.mdm" xmlns:urn1="urn:cs-base.informatica.mdm">
   <soapenv:Header/>
   <soapenv:Body>
      <urn:readEntity>
         <urn:parameters>
            <urn:coFilter>
               <urn1:object name="Person" depth="3" suppress="false">
                  <urn1:key>
                     <urn1:rowid>931</urn1:rowid>
                  </urn1:key>
                  <urn1:fields>firstName,lastName,middleName,gender</urn1:fields>
                  <urn1:recordState>ACTIVE</urn1:recordState>
                  <urn1:recordState>PENDING</urn1:recordState>
                  <urn1:contentMetadata>XREF</urn1:contentMetadata>
               </urn1:object>
            </urn:coFilter>

            <urn:outputFilter>
               <urn1:object name="PersonView" depth="2" suppress="false">
               </urn1:object>
            </urn:outputFilter>
         </urn:parameters>
      </urn:readEntity>
   </soapenv:Body>
</soapenv:Envelope>
```

Create Business Entity Sample

The following sample SOAP request creates a Person business entity:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:cs-ors.informatica.mdm" xmlns:urn1="urn:co-base.informatica.mdm"
xmlns:urn2="urn:cs-base.informatica.mdm" xmlns:urn3="urn:co-ors.informatica.mdm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Header/>
    <soapenv:Body>
        <urn:writeEntity>
            <urn:object xmlns:urn3="urn:co-ors.informatica.mdm" xsi:type="urn3:Person">
                <urn3:Person label="Person">
                    <urn3:partyType>Person</urn3:partyType>
                    <urn3:lastName>test1</urn3:lastName>
                    <urn3:middleName>test1</urn3:middleName>
                    <urn3:firstName>test1</urn3:firstName>
                </urn3:Person>
                <urn3:changeSummary create="#/urn:writeEntity/urn:object/urn1:Person"
logging="false" xmlns:sdo="commonj.sdo">
                    <urn:object sdo:ref="#/urn:writeEntity/urn:object"
```

```
sdo:unset="Person"/>
                </urn3:changeSummary>
            </urn:object>
            <urn:parameters>
                <urn:systemName>Admin</urn:systemName>
            </urn:parameters>
        </urn:writeEntity>
    </soapenv:Body>
</soapenv:Envelope>
```

CHAPTER 6

Cross-reference Records and BVT Calculations Services

This chapter includes the following topics:

Cross-reference Records and BVT Calculations Services Overview

You can use the services for cross-reference records and best version of the truth (BVT) calculations to learn how the source data forms the master record.

You can use these services to perform the following tasks:

- gather information about the source data
- determine how the best version of the truth was determined
- override the BVT calculations to ensure the master records contain the best version of the truth

Getting Cross-reference Data and Investigating BVT Calculations

Master records in the MDM Hub maintain the best version of the truth (BVT). The MDM Hub consolidates the most trustworthy data from several source systems into each master record to achieve the best version of the truth. The MDM Hub stores source data in cross-reference records. You can use business entity services to read data from the cross-reference records and determine how the BVT was calculated.

Get Cross-reference Records

You can use a business entity service to get the cross-reference records for a particular master record.

The REST API URL to get cross-reference records has the following format:

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<row ID>?
contentMetadata=XREF
```

The following sample request retrieves cross-reference records for a Person business entity record with a row ID of 123:

```
GET http://localhost:8080/cmx/cs/localhost-orcl-ORS/Person/123?contentMetadata=XREF
```

.

Get Cross-reference Records Response

The following example shows the cross-reference records that are returned for the Person record with a row ID of 123:

```
GET /Person/123?contentMetadata=XREF

{
  "firstName": "Joe",
  "lastName": "Smith",
  "XREF": {
    "item": [
      {
        "rowidXref": 111,
        "firstName": "Joe",
        "lastName": "Smith",
      },
      {
        "rowidXref": 222,
        "firstName": "John",
        "lastName": "Smith"
      }
    ]
  }
}
```

Determine Contributors to the Master Record

You can use business entity services to see which cross-reference record field contributes to the master record. The contributing record to each field is identified by the row ID of the cross-reference record.

The REST API URL to determine contributors to the master record has the following format:

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<row ID>?
contentMetadata=BVT
```

The following sample request retrieves BVT information for a Person record with a row ID of 123:

```
GET http://localhost:8080/cmx/cs/localhost-orcl-ORS/Person/123?contentMetadata=BVT
```

Determine Contributors to the Master Record Response

The following example shows which cross-reference record contributed to each field in the master record:

```
{
  "firstName": "Joe",
  "lastName": "Smith",
  "BVT": {
    "firstName": {
      "rowidXref": 111
    },
```

```
    "lastName": {
      "rowidXref": 222
    }
  },
}
```

Get the Trust Scores of Contributing Cross-reference Record Fields

You can use business entity services to get the trust scores of cross-reference record fields that contribute to the master record.

The REST API URL to determine contributors and get the trust scores has the following format:

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<row ID>?
contentMetadata=TRUST
```

The following sample request provides trust scores for a Person record with a row ID of 123:

```
GET http://localhost:8080/cmx/cs/ors/Person/123?contentMetadata=TRUST
```

Get the Trust Scores of Contributing Cross-reference Record Fields Response

The following response example provides trust scores for each field in a Person business entity record:

```
{
  "firstName": "John",
  "lastName": "Smith",
  "TRUST": {
    "firstName": {
      "score": 75.0,
      "valid": true,
      "trustSetting" :{
          // custom settings
      }
    },
  },
}
```

Getting the Trust Scores of All Cross-reference Record Fields

Use the REST API with the contentMetadata parameter set to XREF_TRUST to get the trust scores and downgrade percentages of all cross-reference record fields.

The request URL for the Read REST API to determine contributors and get the trust scores:

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<row ID>?
contentMetadata=XREF_TRUST
```

The following sample request retrieves cross-reference data for a Person record with a row ID of 123:

```
GET http://localhost:8080/cmx/cs/localhost-orcl-ORS/Person/123?contentMetadata=XREF_TRUST
```

Get the Trust Scores of All Cross-reference Record Fields Response

The following example shows the trust scores and downgrade percentages of all cross-reference record fields for a Person business entity:

```
{
    "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
                }
            }
        }]
    }
}
```

Get Information about Source Systems

You can get information about which source systems the cross-reference data comes from, and how many cross-reference records the source systems contribute for the whole record, for each node, or for each record.

The following parameters can be specified in the request:

describe

Set to `true` to return the description of the source system. Can be `true` or `false`. Default is `false`.

aggregate

Defines for which level to return source system information. Can be `ENTITY`, `NODE`, and `RECORD`. Default is `ENTITY`.

recordStates

Specifies the record state for which to return records. Can be `ACTIVE`, `PENDING`, or `DELETED`. Default is `ACTIVE`.

compact

When set to `no`, specifies that the entity level data is returned when the aggregate parameter is specified with `ENTITY` and other aggregate levels. Can be `yes` or `no`. For REST API requests only. Default is `yes`.

Get Information about Source Systems Example

The following sample request gets entity-level and node-level source system information for the Person business entity with a row ID of 123:

```
GET http://localhost:8080/cmx/cs/ors/Person/123?
action=getSourceSystems&aggregate=ENTITY,NODE&compact=no
```

The following sample request gets record-level source system information and the source system descriptions for the Person business entity with a row ID of 456:

```
GET http://localhost:8080/cmx/cs/ors/Person/123/Address/456?
action=getSourceSystems&aggregate=ENTITY,NODE&compact=no
```

Get Information about Source Systems Response

The following example shows entity-level and node-level information for a Person business entity:

```
{
      "name": "Admin",
      "xrefCount": 120
    }
  ],
  Person: {
    "rowidObject": "456",
    "sourceSystem":
      {
        "name": "Admin",
        "xrefCount": 30
      }
    ]
  }
}
```

Filtering and Paginating Responses

You can select the fields to return in the response, filter results by several criteria, and paginate results.

Filtering Request Examples

The following table shows sample requests for the Person Business entity with a variety of filters applied and a description of the results that are returned in the response:

Request	Description of returned results
/Person/123	All user-defined fields
/Person/123?readSystemFields=true	All user-defined fields and all system fields
/Person/123?fields=firstName	One user-defined field
/Person/123?fields=updatedBy	One system field
Person/123?fields=firstName,updatedBy	One user-defined field and one system field
/Person/123?fields=firstName&readsystemFields=true	One user-defined field and all system fields

Establish the Best Version of the Truth

After a data steward investigates the source data in the cross-reference records, they can then make adjustments to how source data is consolidated to ensure that the master record represents the best version of the truth.

You can use business entity services to perform the following actions to establish the best version of the truth:

- Update the trust settings
- Remove mismatched source data
- Select the correct contributing field
- Write the correct value to the master record

Select the Correct Contributing Field

If the field with the highest trust score does not contain the best version of the truth, a data steward can select the field that does contain the correct data to contribute the data to the master record.

The URL and request body to select the correct contributing field based on the system name and source key has the following format:

```
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>"
        }
    }
}
```

The URL and request body to select the correct contributing field based on the cross-reference record ID has the following format:

```
POST http://<host>:<port>/<context>/<database ID>/<business entity>/<row ID>?
systemName=<source system name>
{
    BVT: {
        <field name>: {
            rowidXref: "<row ID>"
        }
    }
}
```

Select the Correct Contributing Field Example

The following URL and request body selects the first name field of the cross-reference record from the Sales source system with a source key of 0001 to contribute to the master record:

```
POST http://localhost:8080/cmx/cs/localhost-orcl-ORS/Person/123?systemName=Admin
{
    BVT: {
        firstName: {
            systemName: "Sales",
            sourceKey: "0001"
        }
    }
}
```

The following URL and request body selects the first name field of the cross-reference record with a row ID of 789 to contribute to the master record:

```
POST http://localhost:8080/cmx/cs/localhost-orcl-ORS/Person/123?systemName=Admin
{
    BVT: {
        firstName: {
            rowidXref: "789"
        }
    }
}
```

Write the Correct Value to the Master Record

When you use a business entity service call to write a correct value to a master record, you also can establish the trust settings for the value. If you do not specify trust settings, the MDM Hub uses the administrator system settings.

The URL and request body to write the correct value with the administrator trust setting has the following format:

```
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
            }
        }
    }
}
```

The URL and request body to write the correct value with defined trust settings has the following format:

```
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>"
            }
        }
    }
}
```

Trust Parameters

You can define the following trust parameters:

minimumTrust

Trust level that a data value has when it is old (after the decay period has elapsed). This value must be less than or equal to the maximum trust.

Note: If the maximum and minimum trust are equal, then the decay curve is a flat line, and the decay period and decay type have no effect.

maximumTrust

Trust level that a data value has if it has just been changed. For example, if source system X changes a phone number field from 555-1234 to 555-4321, the new value gets system X's maximum trust level for the phone number field. By setting the maximum trust level relatively high, you can ensure that changes in the source systems are applied to the base object.

timeUnit

Specifies the units used in calculating the decay period—day, week, month, quarter, or year.

maximumTimeUnits

Specifies the number — of days, weeks, months, quarters, or years — used in calculating the decay period.

graphType

Decay follows a pattern in which the trust level decreases during the decay period. The graph types can be one of the following decay patterns:

Graph Type Parameter	Description
LINEAR	Simplest decay. Decay follows a straight line from the maximum trust to the minimum trust.
RISL	Most of the decrease occurs toward the beginning of the decay period. Decay follows a concave curve. If a source system has this graph type, then a new value from the system will probably be trusted, but this value might be overridden.
SIRL	Most of the decrease occurs toward the end of the decay period. Decay follows a convex curve. If a source system has this graph type, it will be relatively unlikely for any other system to override the value in the master record until the value is near the end of its decay period.

Write the Correct Value to the Master Record Example

Example 1

You want to change the name in the master record from Sam Brown to John Smith. The change is attributed to the Sales source system. The trust settings are set to the administrator trust settings.

The following code shows the URL and command for Example 1.

```
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
            }
        }
    }
}
```

Example 2

You want to change the name in the master record from Sam Brown to John Smith. The change is attributed to the SFA source system. The trust settings are set to a minimum trust of 60 and a maximum trust of 90, and the trust decays linearly over a decay period of three months.

The following code shows the URL and command for Example 2.

```
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"
            }
        }
    }
}
```

Remove Mismatched Source Data

If a cross-reference record is incorrectly associated with a particular master record, a data steward can unmerge the cross-reference record. A new master record is created from the unmerged cross-reference record.

Only one cross-reference record can be unmerged in an unmerge call. If several cross-reference records need to be unmerged, make an unmerge call for each cross-reference record.

If a trigger is configured for an unmerge event, then an unmerge task is created. Otherwise, the cross-reference record is unmerged.

The URL and command to unmerge a record based on the system name and source key has the following format:

```
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>" }
}
```

The URL and command to unmerge a record based on the cross-reference record ID has the following format:

```
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>"}
}
```

Remove Mismatched Source Data Example

REST API Example

The following code shows the URL and command to unmerge the cross-reference record at the child level from an Address record:

```
POST http://localhost:8080/cmx/cs/localhost-orcl-MDM_SAMPLE/Person/181921?
action=unmerge&systemName=Admin
{
"name":"Person.Address",
"key":{
"rowid":"41721 ",
"rowidXref":41722
}
}
```

Where:

- the cross-reference record to unmerge has a row ID of 41722
- the row ID of the master record to unmerge the cross-reference record from is 41721
- the row ID of the root record is 181921

SOAP/EJB Example

The following code shows the URL and command to unmerge the cross-reference record at the child level from an Address record:

```
<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>
```

Where:

- the cross-reference record to unmerge has a row ID of 41722
- the row ID of the master record to unmerge the cross-reference record from is 41721
- the row ID of the root record is 181921

Unmerge Response

The unmerge response contains the row ID of the base object that is created from the unmerged cross-reference record.

Response example 1

The following example shows the response when you a cross-reference record from a Person root node.

```
{
    Person: {
        rowidObject: "7777"
    }
}
```

Response example 2

The following example shows the response when you a cross-reference record from an Address child node.

```
{
    Person: {
        Address: {
            item: [
                rowidObject: "55555"
            ]
        }
    }
}
```

CHAPTER 7

Supporting Corporate Linkage Service

This chapter includes the following topics:

Overview

The corporate linkage service from Duns & Bradstreet (D&B) returns the parent of a requested organization and all its related entities. You can use the linkage service from D&B through which you can get information of all branches and divisions of an organization. You can create and update records with the data from the linkage service.

You can import the corporate linkage data into the MDM Hub. You must develop a custom application that can use the linkage service from the DaaS provider custom component in an Entity View.

You require a business entity service to import the data from the D&B service and create a record with the data. When there is a change in the data in the external storage, you must be able to make the corresponding changes in your record. D&B provides a monitoring service that notifies you of changes in data. You require a service that can accept data before and after a change, and apply the changes to the corresponding record.

Business Entity Services for DaaS Import and Update

The DaaS Import business entity service accepts data from the linkage service in XML format and converts it into a record. The DaaS Update business entity service accepts data from the external service in the form of

two XML files. The two XML files correspond to data before and after a change. The Update service applies the changes to the corresponding record.

Related Topics:

- "DaaS Import" on page 247
- "DaaS Update" on page 250

Configuring Linkage Support

To use the linkage service from D&B to create and update records, you must add configuration in the Provisioning tool and create a custom application to split the response from the linkage service.

Perform the following tasks to configure support for the linkage service of D&B:

1. Use the Provisioning tool to upload the WSDL for the linkage service.
2. Use the Provisioning tool to configure an XML document to business entity transformation and expose it as a service. When you expose the transformation as a service, the process creates the DaaS Import and Update business entity services.
3. Create a custom application that can request data from the linkage service and split the response into record details and linkage details.
4. Develop a user interface that sends the request to the custom application.

Note: For more information about how to upload the WSDL and configure an XML to business entity transformation, see the Integrating Data as a Service chapter in the *Multidomain MDM Provisioning Tool Guide*.

Custom Application for Linkage Data Splitting

To use the linkage services from D&B, you must design a custom application that can split the linkage information into record details and linkage details.

The custom application must perform the following functions:

1. Accept request for a linkage service from the Entity View.
2. Send the request to D&B and receive the response.
3. Convert the response to XML.
4. Split the response into record details and linkage details.
5. Send the XML information to business entity services to save as a record in the database.
6. Monitor changes to the data and call the List Change Notice function of the external service.

CHAPTER 8

External Calls to Cleanse, Analyze, and Transform Data

This chapter includes the following topics:

Overview

External providers provide web services to cleanse, analyze, and transform record data. Use the external web services for custom validation, such as checking if the address field is empty when you add a record. Use the external web services for custom logic to transform record data. For example, when you merge two records, you can merge addresses, but not telephone numbers.

An external web service exposes one or more operations that the business entity services can call. Each operation has a request and response type. Business entity services send the record data with the required service parameters to external services. You can configure calls to the external web services for certain steps in the execution logic. Based on the logic that you implement, requests go from Data Director to update the record data. The external services might modify the data, if required.

In the Provisioning tool, configure the business entity and the events for which you want to call the external service. In the Provisioning tool, upload the WSDL file for the external service and register the SOAP service and operation. Bind the service to specific business entities and events.

Use the WSDL file in the Resource Kit to understand the service, operations, methods, and the data types that the service methods exchange. The `custom-logic-service.wsdl` file for the external web services is in the following Resource Kit location: `C:\<infamdm installation directory>\hub\resourcekit\samples\BESExternalCall\source\resources\webapp\WEB-INF\wsdl\`

The Resource Kit includes sample code that implements custom logic and validation. When you install the Resource Kit, the `bes-external-call.ear` file for the sample custom logic and validation is deployed on the application server.

Service Phases

A business entity service consists of multiple service phases. You can configure a service phase to invoke an external call and trigger a supported event.

Service phases trigger in the following order:

1. BeforeEverything. Invokes an external call and triggers a supported event before performing the logic of the service, such as validation, localization, and projection. A sample use case might involve you configuring this service phase to change the default merge behavior to prevent a merge of an attribute of two business entities. For example, during a merge of two or more Person records, you can prevent the merge of the child telephone number records of a parent Person record.

   Note: Projection is the process of adding cross reference information to specific business entity service responses. You can consider projection as a possible additional step that occurs before or after a particular service phase.
2. BeforeValidate. Invokes an external call and triggers a supported event before validating the input data. A sample use case might involve you configuring this service phase to validate logic before validating the input data when persisting a business entity. For example, address or phone number validation that occurs against an external third-party service before persisting a Person record.
3. AfterValidate. Invokes an external call and triggers a supported event immediately after validating the input data. A sample use case might involve you configuring this service phase to validate logic after validating the input data when persisting a business entity. For example, address or phone number validation that occurs against an external third-party service after persisting a Person record.

   Note: When the AfterValidate service triggers, the input data might be different because the previous service phase modified or processed the data.
4. AfterEverything. Invokes an external call and triggers a supported event after performing the logic of the service, such as validation, localization, and projection. A sample use case might involve you configuring this service phase to trigger notification after the service performs all the logic. For example, sending an email notification after a successful merge of two or more Person records.

   Note: When the AfterEverything service phase triggers, an external call has access to the data from the service response and not the input data.

Services that Support Service Phases

You can use the following internal services that support service phases:

WriteCO

Persists and updates business entities. The WriteCO service supports the following service phases:

- BeforeEverything
- BeforeValidate
- AfterValidate
- AfterEverything

These service phases trigger before, during, and after a business entity persists.

WriteView

Persists and updates business entity views. The WriteView service supports the following service phases:

- BeforeEverything
- BeforeValidate
- AfterValidate
- AfterEverything

These service phases trigger before, during, and after a business entity persists.

The WriteView service triggers all supported WriteCO service phases in the following order:

1. WriteView.BeforeEverything
2. WriteView.BeforeValidate
3. WriteView.AfterValidate
4. WriteCO.BeforeEverything
5. WriteCO.BeforeValidate
6. WriteCO.AfterValidate
7. WriteCO.AfterEverything
8. WriteView.AfterEverything

ReadCO

Retrieves business entities. The ReadCO service supports the following service phases:

- BeforeEverything
- AfterEverything

These service phases trigger before and after a read of a business entity.

ReadView

Retrieves business entity views. The ReadView service supports the following service phases:

- BeforeEverything
- AfterEverything

These service phases trigger before and after a read of a business entity view.

Note: The ReadView service triggers all supported ReadCO service phases in the following order:

1. ReadView.BeforeEverything
2. ReadCO.BeforeEverything
3. ReadCO.AfterEverything
4. ReadView.AfterEverything

PreviewMergeCO

Retrieves the merge preview result of multiple business entities. The PreviewMergeCO service supports the following service phases:

- BeforeEverything
- AfterEverything

These service phases trigger before and after the generation of the preview of a merged business entity.

MergeCO

Merges multiple business entities. The MergeCO service supports the following service phases:

- BeforeEverything
- AfterEverything

These service phases trigger before and after the merge of a business entity.

Process Method Parameters

When you implement the business logic for an external call, you can use process method parameters.

Use the following parameters in a custom logic Java interface:

helperContext

Defines the helper execution context that the Service Data Objects (SDO) operation requires.

inputSdo

Specifies the Service Data Object that represents the business entity, including parent and child records.

inParams

Specifies input properties from the previous request and directly depends on the defined service phase within the external call. The inParams parameter can also include the outParams parameters from a previous call.

For example, validateOnly is an input property, that indicates to perform validation at a parent record, child record, or business entity level.

The validateOnly property can have the following values:

- true. Performs validation for a parent or child record. For example, you can specify this property to validate 10 telephone numbers individually.
- null or false. Performs validation at a business entity level. For example, you can specify this property to perform validation for the business entity when a user saves the record.

Another example is the servicePhase property that specifies the name of the external call.

outParams

Adds or overwrites the supported inputSdo and the inParams parameters. For example, an external call that includes the Merge_CO_Before_Everything event might update the keysAndOverrides parameter to add more business entity keys or remove keys for a merge.

In the return value, when there is a data change in the inputSdo, the external call returns one of the following values:

- null. When there is no data change in the inputSdo parameter.
- The modified inputSdo parameter. When there is a data change in the inputSdo parameter and you want to reflect the change in the business entity.

The following sample code shows a custom logic java interface that includes the process method parameters:

```
public class ValidateOrgWriteBeforeEverything implements CustomLogic {
    @Override
    public DataObject process(HelperContext helperContext, DataObject inputSdo,
            Map<String, Object> inParams,
            Map<String, Object> outParams) throws StepException {
```

```
                    // your implementation
                    return null;
                              //or
                    return inputSdo;
            }
    }
```

External Call Best Practices

Consider the following best practices when implementing external calls:

- Implement an external call when an existing function does not exist to cleanse, analyze, or transform data.
- Be cautious when you implement an external call because the call can impact the performance of the MDM Hub and business entity service call response.
- Identify when you want to use the external call. For example, run an external call when a user navigates from one child record to another or when the user clicks **Save**.
- Define a separate class for each external call.
- If you include the inParams process method parameter with the validateOnly property in an external call, ensure that you set the validateOnly property to one of the following values:
  - false or null. Runs the external call after the user clicks **Save**.
  - true. Runs the external call after the user updates a parent or child record and then navigates to another child record.
- If you include custom properties in subsequent external calls up to the AfterEverything event, add the custom properties in the outParams process method parameter.
- The final WriteCO.AfterEverything event during a save operation returns row IDs of parent and child records. If you require other data, call read business entity services.
- Do not implement external calls for read operations because read operations occur during the following operations:
  - Search
  - Read before write
  - Read after write
- Validate records through cleanse functions.
- If you implement additional validation, perform the additional validation at the WriteCO.AfterEverything event when the validateOnly property is set to one of the following values:
  - false or null. For a base object.
  - true. For a child record.
- Make a business entity services call from an external call similar to any external application that uses business entity services calls to read or write data.

Creating External Calls

A business entity service has service steps. An incoming request passes through each service step. You can create external calls to external services for certain steps in the business entity service execution logic. Create an external call when a function does not exist to cleanse, analyze, or transform data.

To create an external call, perform the following steps:

1. Understand and deploy the example Web service.
2. Log in to the Provisioning tool.
3. Upload the WSDL file.
4. Register the SOAP service.
5. Configure the external call.
6. Publish the configurations.

After you create an external call, to ensure that the call works as expected, test the call.

To view a sample external call that includes these steps, see "Sample External Call" on page 293.

Sample External Call

To learn how to create an external call, review the BESExternalCall sample that is available in the Resource Kit. The following sections include information about the required steps to create and test the BESExternalCall sample.

The BESExternalCall sample implements the simplest foundation for a successful external call setup.

The BESExternalCall sample has the following code structure:

- CustomLogic implementations in the ValidatePerson.java and MergePerson.java files
- CustomLogicFactory implementation in the CustomLogicFactorImpl.java file
- WebService implementation in the CustomLogicService.java file

CustomLogic implementations in the ValidatePerson.java and MergePerson.java files

CustomLogic implementations for an external call include the custom business logic, such as additional data validation and manipulation.

CustomLogic implementations must include a com.informatica.mdm.spi.externalcall.CustomLogic interface, which is available in the mdm-spi.jar file. This interface exposes the process method where you can define custom business logic. The following ValidatePerson.java class code is a sample:

```
public class ValidatePerson implements CustomLogic {
    private CompositeServiceClient besClient;
    private CallContext callContext;
    public ValidatePerson(CompositeServiceClient besClient, CallContext callContext) {
        this.besClient = besClient;
        this.callContext = callContext;
    }
    @Override
    public DataObject process(HelperContext helperContext, DataObject inputSdo,
                              Map < String, Object > inParams,
                              Map < String, Object > outParams) throws StepException {
            try {
                List < ValidationError > errorList = new ArrayList < > ();
                DataFactory dataFactory = helperContext.getDataFactory();
```

```
                // Ensure that the Person SDO has at least one Address item
                List<?> list = inputSdo.getList("Person/Addresses/item");
                if ((list == null) || list.isEmpty()) {
                    String personId = inputSdo.getString("Person/rowidObject");

                    if(personId != null) { // Verify if an address already exists
                        DataObject readEntity = dataFactory.create(ReadEntity.class);
                        DataObject personFilter =

readEntity.createDataObject("parameters").createDataObject("coFilter").createDataObject("
object");
                        personFilter.setString("name", "Person");
                        personFilter.createDataObject("key").setString("rowid",
personId);
                        DataObject addressFilter =
personFilter.createDataObject("object");
                        addressFilter.setString("name", "Addresses");

addressFilter.createDataObject("pager").setInt("recordsToReturn", 1);
                        DataObject readEntityReturn = besClient.process(callContext,
readEntity);
                        List existingList = readEntityReturn.getList("object/Person/
Addresses/item");
                        if (existingList == null || existingList.isEmpty()) {
                            errorList.add(createValidationError(dataFactory,
"CUSTOM-00001", "You must enter at least one Address.", "Person.Addresses"));
                        }
```

CustomLogicFactory implementation in the CustomLogicFactorImpl.java file

A CustomLogicFactory implementation provides the CustomLogic instances to run at the expected phase, run for the target business entity, or both.

A CustomLogicFactory implementation must include a com.informatica.mdm.spi.externalcall CustomLogicFactory interface, which is available in the mdm-spi.jar file. This interface exposes the create method, which the external call request invokes at early stages. The logic expected in this method filters and directs the external call request to the expected custom logic implementation. The following CustomLogicFactoryImpl.java code is a sample:

```
public class CustomLogicFactoryImpl implements CustomLogicFactory {

            public static final String PERSON = "Person";

            private static final CustomLogic EMPTY_LOGIC = new EmptyLogic();

            private CompositeServiceClient besClient;

            public CustomLogicFactoryImpl(CompositeServiceClient besClient)
{ this.besClient = besClient;
            }

            @Override
            public CustomLogic create(ExternalCallRequest externalCallRequest,
CallContext callContext) throws StepException {
             // Interest is in the Person business entity. The logic can handle a few
service phases.
                Trigger trigger = externalCallRequest.getTrigger();
                String businessEntity = trigger.getBusinessEntity();
                ServicePhase phase = trigger.getServicePhase();

                switch (phase) {
                    case WRITE_CO_BEFORE_VALIDATE:
                        if (PERSON.equals(businessEntity)) {
                            return new ValidatePerson(besClient, callContext);
                        }
                        break;
                    case PREVIEW_MERGE_CO_BEFORE_EVERYTHING:
                    case MERGE_CO_BEFORE_EVERYTHING:
```

```
                    if (PERSON.equals(businessEntity)) {
                        return new MergePerson();
                    }
                    break;
                default:
                    //
            }
            return EMPTY_LOGIC; // This one does nothing
        }
```

A CustomLogicFactory implementation can include the following create method parameters:

externalCallRequest

An instance of the request that represents the external call as defined in the bes-external-call.xsd file.

callContext

An instance of the composite service context that the CompositeServiceClient requests use in custom logic implementations.

WebService implementation in the CustomLogicService.java file

A WebService implementation exposes the complete service that an external call invokes and is the entry point of the external call flow.

The following CustomLogicService.java sample code implements the web service that the bes-external-call.xsd file defines based on the native WebService API from Java:

```
/**
* Web service implementation.
* It must accept a {urn:bes-external-call.informatica.mdm}ExternalCallRequest as an
operation input
* and return a {urn:bes-external-call.informatica.mdm}ExternalCallResponse as output.
*/
@WebServiceProvider(
        targetNamespace = "http://cs.sample.mdm.informatica.com/",
        serviceName = "CustomLogicService",
        portName = "CustomLogicServicePort",
        wsdlLocation = "WEB-INF/wsdl/custom-logic-service.wsdl"
)
@ServiceMode(Mode.PAYLOAD)
public class CustomLogicService implements Provider<Source> {

    @Override
    public Source invoke(Source request) {

        CompositeServiceClient compositeServiceClient =
createCompositeServiceClient();
        CustomLogicFactory customLogicFactory = new
CustomLogicFactoryImpl(compositeServiceClient);
        String appName = "<trusted_user>"; // replace with proper application user name

        // Create a processor instance and let the instance perform the job.
        // Provide a custom logic factory implementation.
        ExternalCallProcessor externalCallProcessor =
                new ExternalCallProcessor(compositeServiceClient, appName,
customLogicFactory);

 return externalCallProcessor.invoke(request);
}
```

The invoke method of the ExternalCallProcessor instance handles the incoming request.

Note: The ExternalCallProcessor instance relies on a trusted application user. For more information about trusted application users, see the *Multidomain MDM Security Guide*.

BESExternalCall example

The BESExternalCall external call to the web service can perform the following actions:

- When creating or updating a person, the call validates whether the person has at least one address. If the validation fails, the call returns a customized error message.
- When generating a preview of merged records, the call does not merge telephone numbers.
- When merging records, the call does not merge telephone numbers.

The following table describes the files and the file locations that support the BESExternalCall example:

File Name	Description	Location
`bes-external-call.xsd`	Defines the request and response types for all external calls. An external web service operation must use the request and response types for the input and output elements.	`<MDM installation directory>/hub/server/lib/mdm-spi.jar`
`custom-logic-service.wsdl`	WSDL file for the example web service. Defines the external services, operations, methods, and data types that the service methods exchange.	`<MDM installation directory>/hub/resourcekit/samples/BESExternalCall/source/resources/webapp/WEB-INF/wsdl/`
`ValidatePerson.java` and `MergePerson.java`	Examples of custom validate and merge operations that run with the example web service. **Note:** This example uses .java files, but your external web service might require a different coding standard.	`<MDM installation directory>/hub/resourcekit/samples/BESExternalCall/source/java/`
`build.xml`	Builds the `bes-external-call.ear` file.	`<MDM installation directory>/hub/resourcekit/samples/BESExternalCall/`
`bes-external-call.ear`	The `.ear` file that you deploy to the application server to run this example.	`<MDM installation directory>/hub/resourcekit/samples/BESExternalCall/build/`

Step 1. Understand and Deploy the Example Web Service

The BESExternalCall example includes a WSDL file and the Java files with custom logic that the external calls use. Review the files in the example. To use the example, build and deploy the `.ear` file to the application server that runs the MDM Hub.

When you design external calls, first develop the custom logic in the framework of the external service. Then create the WSDL file that defines the external services, operations, methods, and data types that the service methods exchange. Ensure that the WSDL file uses the request and response types that are defined in the `bes-external-call.xsd` file.

1. To understand the requirements for the request and response types, review the `bes-external-call.xsd` file.
2. To understand the implementation of the request and response types for the example web service, review the `custom-logic-service.wsdl` file.
3. To understand the custom logic the example web service uses, review the Java files in the Resource Kit.

4. To build the example `.ear` file, use the `build.xml` file with an ANT build tool.

   For more information about building the EAR file, see the *Multidomain MDM Resource Kit Guide*.
5. Deploy the `bes-external-call.ear` file to the application server that runs the MDM Hub.

   For more information about deploying the EAR file, see the *Multidomain MDM Resource Kit Guide*.

Step 2. Log in to the Provisioning Tool

Configure the web service and external call in the Provisioning tool.

To log in, you need the URL for the Provisioning tool and your user credentials. The URL includes the MDM Hub server host name and port number. If you do not have this information, contact your MDM Hub administrator.

The MDM Hub must run on the same application server as the Provisioning tool. For more information about the Provisioning tool, see the *Multidomain MDM Provisioning Tool Guide*.

1. Open a supported browser.
2. Enter the URL for the Provisioning tool.

   The URL has the following formats:

   - Secure connections. `https://<MDM Hub Server host name>:<MDM Hub Server port number>/provisioning/`
   - Non-secure connections. `http://<MDM Hub Server host name>:<MDM Hub Server port number>/provisioning/`

   The **Log In** page opens.
3. Enter your user name and password.
4. Click **Log In**.
5. When prompted for an ORS, select the sample MDM ORS.

   The Provisioning tool opens and displays the **Home** page.

Step 3. Upload the Sample WSDL file

Upload the WSDL file for the BESExternalCall example web service.

For more information about uploading the WSDL file, see the *Multidomain MDM Provisioning Tool Guide*.

1. In the Provisioning tool, click **Business Entity** > **Extensions**.

   The **Extensions** page opens.
2. Select **WSDL**, and then click **Create**.
3. In the properties pane, specify the following properties:

Property	Description	Example Value
Name	A descriptive name for the example web service.	besexternal
URL	URL of the WSDL file.	`https://<host>:<port>/bes-external-call/customLogicService?wsdl`

4. Click **Apply**.

   The **Tree View** panel displays the details.

Step 4. Register the SOAP Service

Register the SOAP service and operations that the WSDL describes.

For more information about registering the SOAP service, see the *Multidomain MDM Provisioning Tool Guide*.

1. In the **Extensions** page, select **SOAP Services**, and then click **Create**.

   The properties appear in the properties panel.
2. In the properties panel, specify the following properties:

Property	Description	Example Value
Name	Name for the SOAP service.	userexit
WSDL	Name that you specified when you uploaded the example WSDL file. The details of the web service appear in the Namespace, Service Name, Port Name, and Endpoint Address fields.	besexternal
SOAP Header	Any additional information that you require to run the web service, such as user name and password.	In the BESExternalCall example, there are no SOAP headers.

3. Click **Apply**.

   The **Tree View** panel displays the details.

Step 5. Configure the External Call

The external call uses the configured SOAP service and operations, and the call is triggered for three service phases.

For more information about configuring external calls, see the *Multidomain MDM Provisioning Tool Guide*.

1. In the **Extensions** page, select **External Calls**, and then click **Create**.
2. In the **Properties** panel, set the following properties:

Property	Description	Example Value
Name	Name for the external call.	validateperson
SOAP Service	SOAP service that you configured.	userexit
SOAP Operation	SOAP operation to perform.	validate

3. Click **Apply**.

   The **Tree View** panel populates based on the content of the SOAP service and operation.
4. In the **Tree View** panel, select **Business Entities**, click **Create**, and then select **Person**.

5. Select **Service Phase**, click **Create**, and then select the following service phases for the sample code:
   - WriteCO.BeforeValidate
   - PreviewMergeCO.BeforeEverything
   - MergeCO.BeforeEverything

   Note: Based on your custom logic, you can select the service phases.
6. Click **Apply**.

Step 6. Publish the Configurations

The configurations for the WSDL file, SOAP service, and external calls are stored in a cache. Publish the configurations to the MDM Hub.

In the Provisioning tool, click **Publish**.

The publish process saves the configurations to the MDM Hub.

Test the Address Validation

Use the REST APIs to verify that the validation step works. Create a Person record without an address. If the external call setup is correct, a validation error appears that indicates that a Person record must have at least one address.

To use the REST URLs, you need the database ID for the MDM sample ORS and an MDM user name.

1. Find the database ID for the MDM sample ORS:
   a. Log in to the Hub Console.
   b. In the **Configuration** workbench, click **Databases**.
   c. Select the MDM sample database.
   d. In the **Database Properties** panel, copy the value for **Database ID**. Use this value in the REST URL.
2. Use the Create API to create a Person business entity without an address.

```
POST https://<host>:<port>/cmx/cs/<database id>/Person?systemName=<user name>
{
        firstName: "John"
    }
```

   The validation test fails and displays the error. The Person record is not created.
3. Create a Person business entity with an address.

```
POST https://<host>:<port>/cmx/cs/<database id>/Person?systemName=<user name>
{
        firstName: "John",
        Addresses: {
            item: [
                {
                    cityName: "Toronto"
                }
            ]
        }
    }
```

   The validation test passes without errors. The Person record is created.

Test the Merge Logic

To test the merge logic that prevents telephone numbers from merging, use the MergePreview API .

1. To create two Person business entities, use the Create API.

```
POST https://<host>:<port>/cmx/cs/<ors>/Person?systemName=Admin
    {
        firstName: "John",
        Addresses: {
            item: [
                {
                    cityName: "Toronto"
                }
            ]
        },
        TelephoneNumbers: {
            item:[
                {
                    phoneNum: "111-11-11"
                }
            ]
        }
    }
POST https://<host>:<port>/cmx/cs/<ors>/Person?systemName=Admin
    {
        firstName: "John",
        Addresses: {
            item: [
                {
                    cityName: "Ottawa"
                }
            ]
        },
        TelephoneNumbers: {
            item:[
                {
                    phoneNum: "222-22-22"
                }
            ]
        }
    }
```

   The response includes the following sample row IDs:

   - Person: 161923, 161924
   - Addresses: 2123, 2124
   - TelephoneNumbers: 101723, 101724

2. To preview the merged Person business entities, run the PreviewMerge API.

```
POST https://<host>:<port>/cmx/cs/<ors>/Person/161923?action=previewMerge&depth=2
{
        keys: [
            {
                rowid: "161924"
            }
        ]
    }
```

   The response includes two addresses and two telephone numbers.

```
{
  "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": "SYS0          ",
           "hubStateInd": 1,
           "label": "Addresses",
           "Address": {
             "rowidObject": "2122          ",
             "creator": "admin",
             "createDate": "2016-10-20T09:50:54.777-04:00",
             "updatedBy": "admin",
             "lastUpdateDate": "2016-10-20T09:50:54.777-04:00",
             "consolidationInd": 4,
             "lastRowidSystem": "SYS0          ",
             "hubStateInd": 1,
             "label": "Address",
             "cityName": "Ottawa"
           }
         }
       ]
     }
   }
}
```

3. Run the PreviewMerge API with the merge overrides.

   The overrides trigger the external call that prevents the telephone numbers from merging.

```
POST https://<host>:<port>/cmx/cs/<ors>/Person/161923?action=previewMerge&depth=3
{
    keys: [
        { rowid: "161923" }
    ],
    overrides: {
        Person: {
            Addresses: {
                item:[
                    {
                        rowidObject: "2123",
                        MERGE: {
                            item:[{key:{rowid: "2124"}}],
                            $original: {
                                item:[null]
                            }
                        }
                    }
                ]
            },
            TelephoneNumbers: {
                item:[
                    {
                        rowidObject: "101723",
                        MERGE: {
                            item:[{key:{rowid: "101724"}}],
                            $original: {
                                item:[null]
                            }
                        }
                    }
                ]
            }
        }
    }
}
```

The response contains a single address and two telephone numbers:


```
{
  "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": []
    }
  }
}
```

Getting Source Row IDs with the MergeCO.BeforeEverything External Call

During a merge action, source row IDs help trace the source base objects. You can get source row IDs from the inputSDO process method parameter with the MergeCO.BeforeEverything external call.

To get the source row IDs, perform the following steps:

1. Implement the MergeCO.BeforeEverything external call.

   The following sample code from the MergePerson.java file gets the source row IDs with the MergeCO.BeforeEveryting external call:

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

import java.util.List;
import java.util.Map;

import com.informatica.mdm.sdo.co.base.BaseObject;
import com.informatica.mdm.sdo.co.base.CompositeObjectBase;
import com.informatica.mdm.sdo.co.base.KeysAndOverrides;
import com.informatica.mdm.sdo.co.base.MergedRecord;
import com.informatica.mdm.sdo.co.base.MergedRecordPager;
import com.informatica.mdm.sdo.co.base.Pager;
import com.informatica.mdm.sdo.cs.base.BusinessEntityKey;
import com.informatica.mdm.sdo.cs.base.Key;
import com.informatica.mdm.spi.cs.StepException;
import com.informatica.mdm.spi.externalcall.CustomLogic;
import commonj.sdo.ChangeSummary;
import commonj.sdo.DataObject;
import commonj.sdo.Property;
import commonj.sdo.helper.HelperContext;
import org.eclipse.persistence.sdo.SDOChangeSummary;

/**
 * Call this piece of code before the Person business entity record merge or the
previewMerge API call.
 */
public class MergePerson implements CustomLogic {

    public static final String KEYS_AND_OVERRIDES = "keysAndOverrides";
    public static final String OVERRIDES = "overrides";
    public static final String MERGE = "MERGE";

    @Override
    public DataObject process(HelperContext helperContext, DataObject inputSdo,
            Map<String, Object> inParams,
            Map<String, Object> outParams) throws StepException {

        DataObject keysAndOverrides = (DataObject) inParams.get(KEYS_AND_OVERRIDES);

        if (keysAndOverrides != null) {

            explainMerge(inParams);
```

```
            DataObject overrides = keysAndOverrides.getDataObject(OVERRIDES);
            if (overrides != null) {
                // if "overrides" are provided, then user is not simply merging
several records, but
                // tryes to merge some child records or provides some winner override
                List<DataObject> list = overrides.getList("Person/TelephoneNumbers/
item");
                if ((list != null) && !list.isEmpty()) {
                    SDOChangeSummary changeSummary = (SDOChangeSummary)
overrides.getChangeSummary();
                    changeSummary.resumeLogging();
                    // remove "MERGE" element from all "Telephone" items, this will
prevent mering of "Telephone" child records
                    for (DataObject dataObject : list) {
                        if (dataObject.isSet(MERGE)) {
                            dataObject.unset(MERGE);
                        }
                    }
                    // send updated "overrides" back to Hub
                    outParams.put(KEYS_AND_OVERRIDES, keysAndOverrides);
                }
            }
        }
        return inputSdo;
    }

    private void explainMerge(Map<String, Object> inParams) {
        BusinessEntityKey businessEntityKey = (BusinessEntityKey)
inParams.get("businessEntityKey");
        System.out.println(String.format("%s:%s", businessEntityKey.getName(),
businessEntityKey.getKey().getRowid()));
        KeysAndOverrides keysAndOverrides = (KeysAndOverrides)
inParams.get("keysAndOverrides");
        List<Key> keys = keysAndOverrides.getKeys();
        for (Key key : keys) {
            System.out.println(String.format(" | <- %s", key.getRowid()));
        }
        CompositeObjectBase overrides = keysAndOverrides.getOverrides();
        if(overrides != null) {
            DataObject co = (DataObject) overrides;
            ChangeSummary changeSummary = co.getChangeSummary();
            DataObject root = co.getDataObject(co.getType().getName());
            dumpOne("", co.getType().getName(), root, changeSummary);
        }
    }

    private void dumpOne(String prefix, String name, DataObject dataObject,
ChangeSummary changeSummary) {
        String rowidObject = dataObject.getString("rowidObject");
        if(rowidObject != null) {
            System.out.println(String.format("%s%s:%s", prefix, name, rowidObject));
        }
        List<MergedRecord> mergeItems = dataObject.getList("MERGE/item");
        if(mergeItems != null) {
            for (MergedRecord mergeItem : mergeItems) {
                System.out.println(String.format("%s | <- %s", prefix,
mergeItem.getKey().getRowid()));
            }
        }
        List<Property> properties = dataObject.getInstanceProperties();
        for (Property property : properties) {
            Object o = dataObject.get(property);
            if((o instanceof Pager) && !(o instanceof MergedRecordPager)) {
                DataObject pager = (DataObject) o;
                System.out.println(prefix + property.getName());
                List<DataObject> items = pager.getList("item");
                for (int i = 0; i < items.size(); i++) {
                    DataObject item = items.get(i);
```

```
                    dumpOne(prefix + "  " ,"[" + i + "]", item, changeSummary);
                }
            }
            if(o instanceof BaseObject) {
                dumpOne(prefix + "  ", property.getName(), (DataObject) o,
changeSummary);
            }
        }
        if(changeSummary.isModified(dataObject)) {
            List<ChangeSummary.Setting> oldValues =
changeSummary.getOldValues(dataObject);
            for (ChangeSummary.Setting setting : oldValues) {
                Property property = setting.getProperty();
                if(!"MERGE".equals(property.getName())) {
                    Object value = dataObject.get(property);
                    System.out.println(String.format("%s manual override: %s = %s",
prefix, property.getName(), value));
                }
            }
        }
    }
}
```

Note: The inputSDO process method parameter is the merge target. Any keys and overrides are from the merge source.

2. To trigger an external call, run a POST request for the Merge API or Preview Merge API with any standard REST client, such as Postman.

   The process calls the MergeCO.BeforeEverything event.

   The following sample POST request outputs the source row IDs:

```
POST: http://localhost:8080/cmx/cs/localhost-orcl103-MDM_SAMPLE/Person/
{{person1_id}}?action=previewMerge

{
        "keys": [
            {
                "rowid": "{{person2_id}}"
            }
        ],
        "overrides": {
            "Person": {
                "Addresses": {
                    "item":[
                        {
                            "rowidObject": "{{person1_Addresses1_id}}",
                            "MERGE": {
                                "item":[{"key":{"rowid":
"{{person2_Addresses1_id}}"}}],
                                "$original": {
                                    "item":[null]
                                }
                            }
                        }
                    ]
                },
                "TelephoneNumbers": {
                    "item":[
                        {
                            "rowidObject": "{{person1_TelephoneNumbers1_id}}",
                            "phoneNum": "777-77-77",
                            "$original": {
                                "phoneNum":null
                            },
                            "MERGE": {
                                "item":[{"key":{"rowid":
"{{person2_TelephoneNumbers1_id}}"}}],
                                "$original": {
                                    "item":[null]
```

```
                                    }
                                }
                            }
                        ]
                    }
                }
            }
        }
```

The following sample output displays the source row IDs:

```
2020-03-20 11:59:49,631 INFO  [stdout] (default task-55) Person:401924
2020-03-20 11:59:49,631 INFO  [stdout] (default task-55)  | <- 401925
2020-03-20 11:59:49,632 INFO  [stdout] (default task-55)  | <- 401926
2020-03-20 11:59:49,632 INFO  [stdout] (default task-55) TelephoneNumbers
2020-03-20 11:59:49,632 INFO  [stdout] (default task-55)   [0]:1724
2020-03-20 11:59:49,632 INFO  [stdout] (default task-55)    | <- 1725
2020-03-20 11:59:49,633 INFO  [stdout] (default task-55)    | <- 1726
2020-03-20 11:59:49,633 INFO  [stdout] (default task-55)    manual override:
phoneNum = 777-77-77
2020-03-20 11:59:49,633 INFO  [stdout] (default task-55) Addresses
2020-03-20 11:59:49,633 INFO  [stdout] (default task-55)   [0]:2124
2020-03-20 11:59:49,633 INFO  [stdout] (default task-55)    | <- 2125
2020-03-20 11:59:49,634 INFO  [stdout] (default task-55)    | <- 2126
```

Example: Custom Validation and Logic for Business Entity Services

You can test the custom validation and logic when you add and merge the Person records. The custom validation checks if the Person record has an address. The custom logic does not allow you to merge two telephone numbers. Use REST APIs to create and merge the Person records.

1. To check the validation when you create a Person record, perform the following steps:
   a. Use the Create API to create a Person record without an address. You get a validation error.
   b. Use the Create API to create a Person record with an address. The operation is successful.
2. To check the custom logic when you merge records, perform the following steps:
   a. Use the Create API to create two Person records with addresses and telephone numbers.
   b. Use the Preview Merge API to merge the two Person records. Add `overrides` to the preview merge request to merge the addresses and the telephones. The response shows a single address, but two telephone numbers. The custom logic prevents the merging of telephone numbers.

Prerequisites

To check the custom logic and validation, you must upload the WSDL file in the Provisioning tool. You must register the SOAP service and operation. Bind the service to the business entities and events for which you

want to use custom logic and validation. You can test the logic and validation for the specified business entities and events.

Step 1. Test Custom Validation

Use the Create API to create the following Person record without an address:

```
POST http://localhost:8080/cmx/cs/localhost-orcl-mdm_Sample/Person?systemName=Admin
{
        firstName: "John"
    }
```

You get a validation error.

Use the Create API to create the following Person record with an address:

```
POST http://localhost:8080/cmx/cs/localhost-orcl-mdm_Sample/Person?systemName=Admin
{
        firstName: "John",
        Addresses: {
            item: [
                {
                    cityName: "Toronto"
                }
            ]
        }
    }
```

The request creates a Person record.

Step 2. Test Custom Logic

Perform the following steps to test the custom logic:

1. Use the Create API to create two Person records with addresses and telephones:

```
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"
                }
            ]
        }
    }
```

The response contains the following rowIds:

- Person: 161923, 161924
- Addresses: 2123, 2124
- TelephoneNumbers: 101723, 101724

2. Run the PreviewMerge API to merge both the Person records:

```
POST http://localhost:8080/cmx/cs/localhost-orcl-mdm_sample/Person/161923?
action=previewMerge&depth=2
{
        keys: [
            {
                rowid: "161924"
            }
        ]
    }
```

The response is a Person record with two Addresses and two Telephone numbers.


```
{
  "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": "SYS0          ",
            "hubStateInd": 1,
            "label": "Addresses",
            "Address": {
              "rowidObject": "2122          ",
              "creator": "admin",
              "createDate": "2016-10-20T09:50:54.777-04:00",
              "updatedBy": "admin",
              "lastUpdateDate": "2016-10-20T09:50:54.777-04:00",
              "consolidationInd": 4,
              "lastRowidSystem": "SYS0          ",
              "hubStateInd": 1,
              "label": "Address",
              "cityName": "Ottawa"
            }
          }
        ]
      }
    }
}
```


3. Run the PreviewMerge API to merge both the Person records, with overrides to merge the addresses and telephone numbers:

```
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]
                            }
                        }
                    }
                ]
            }
        }
    }
}
```

The response shows a single address, but two telephone numbers:

```
{
  "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": []
      }
    }
  }
```


Example: Call a Services Integration Framework Call from an External Call

For security reasons, MDM does not pass password and payload information to an external call. The BESExternalCallWithSIF example shows how to call a Services Integration Framework (SIF) call from an external call.

The external call in this example performs the following custom validation:

1. Checks whether a phone number in a Person record has special characters in the phone extension number.
2. If the phone number includes special characters, corrects the phone number.
3. Checks whether only one phone number has the type PRI.

After the external call completes the validation that only one phone number exists with the type PRI, the external call performs an update using the SIF Put call.

Prerequisites

Configure an application user for calls from the external service to the MDM Hub. This example uses e360 as an application user. Replace this application user with the user you configured. For more information, see the *Multidomain MDM Security Guide*.

Calling a SIF Call from an External Call

1. Build and deploy the bes-external-call.ear file.
2. In the Provisioning tool, register the web service as a SOAP service.
   The following table lists the properties that you can configure for the SOAP service:

Property	Example Value
Name	besexternal
URL	`https://<host>:<port>/bes-external-call/CustomLogicService?wsdl`

3. In the Provisioning tool, register the external call.
   The following table lists the properties that you can configure for the external call:

Property	Example Value
SOAP Operation	validate
Business Entity	Person
Service Phase	WriteCO.BeforeValidate

   - Operation: validate
   - Business Entity: Person
   - Service Phases: WriteCO.BeforeValidate

4. Create a Person record with a phone number that has special characters in the phone extension number.
   The following sample POST request includes the special characters @ in the phone number extension:

```
POST /cmx/cs/<ors>/Person?systemName=Admin
    {
        "firstName": "John",
        "TelephoneNumbers": {
            "item": [
                {
                    "phoneExtNum": "e@1",
                    "phoneNum": "1234567",
                    "phoneTypeCd":{
                       "phoneType":"PRI"
                     }
                }
            ]
        }
    }
```

   The following response displays the validation error:

```
<?xml version="1.0" encoding="UTF-8"?>
   <ns1:CsFault xmlns:ns1="urn:cs-base.informatica.mdm" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance">
       <ns1:errorMessage>SIP-50022: Validation failed.</ns1:errorMessage>
       <ns1:errorCode>SIP-50022</ns1:errorCode>
       <ns1:details xsi:type="ns1:ValidationErrors">
           <ns1:error>
               <ns1:code>CUSTOM-00001</ns1:code>
               <ns1:message>The Phone Ext Number has special symbols.</ns1:message>
               <ns1:field>Person.TelephoneNumbers.phoneExtNum</ns1:field>
           </ns1:error>
```

```
        </ns1:details>
    </ns1:CsFault>
```

The following sample is a valid POST request that does not generate a validation error:

```
POST /cmx/cs/<ors>/Person?systemName=Admin
    {
        "firstName": "John",
        "TelephoneNumbers": {
            "item": [
                {
                    "phoneExtNum": "e1",
                    "phoneNum": "1234567",
                    "phoneTypeCd":{
                       "phoneType":"PRI"
                     }
                }
            ]
        }
    }
```

5. Create a Person record with a phone number that includes a letter, such as o.
   When the POST request runs, the number 0 replaces the letter o.

   The following sample POST request includes a phone number with the letter o:

```
POST /cmx/cs/<ors>/Person?systemName=Admin
    {
        "firstName": "John",
        "TelephoneNumbers": {
            "item": [
                {
                    "phoneExtNum": "e1",
                    "phoneNum": "123o4567",
                    "phoneTypeCd":{
                       "phoneType":"PRI"
                     }
                }
            ]
        }
    }
```

6. Consider the following scenarios that demonstrate how the phone type value is set depending on whether a POST request includes one or multiple phone records:

   - Scenario 1. This scenario includes two POST requests that create one phone record each. The following POST request creates the first phone record with the phone type of PRI:

```
POST /cmx/cs/<ors>/Person?systemName=Admin
    {
        "firstName": "John",
        "TelephoneNumbers": {
            "item": [
                {
                    "phoneExtNum": "e1",
                    "phoneNum": "12304567",
                    "phoneTypeCd":{
                       "phoneType":"PRI"
                     }
                }
            ]
        }
    }
```

   When the following post request runs to create the second phone record, the phone type is set to PRI for the second phone record and the phone type from the first post request is changed to OTH.

```
POST /cmx/cs/<ors>/Person/rowidObject?systemName=Admin
    {
        "TelephoneNumbers": {
            "item": [
```

```
                {
                    "phoneExtNum": "e2",
                    "phoneNum": "23045678",
                    "phoneTypeCd":{
                       "phoneType":"PRI"
                     }
                }
            ]
        }
    }
```

- Scenario 2. This scenario includes one POST request that creates two phone records. The following sample POST request includes two phone records with the phone type of PRI:

```
POST /cmx/cs/<ors>/Person?systemName=Admin
    {
        "firstName": "John",
        "TelephoneNumbers": {
            "item": [
                {
                    "phoneExtNum": "e1",
                    "phoneNum": "12304567",
                    "phoneTypeCd":{
                       "phoneType":"PRI"
                     }
                },
                {
                    "phoneExtNum": "e2",
                    "phoneNum": "23045678",
                    "phoneTypeCd":{
                       "phoneType":"PRI"
                     }
                }
            ]
        }
    }
```

  When this POST request runs, the phone type for the first phone record is set to PRI. For the second phone record, the phone type is set to OTH.

APPENDIX A

Using REST APIs to Add Records

This appendix includes the following topics:

Using REST APIs to Add Records Overview

After you create a business entity model and configure the business entity structure, you can use the REST APIs to add the records.

The following sections use the example of the Person business entity to illustrate how you can add records by using REST APIs. The Person business entity contains data for the employees in your company.

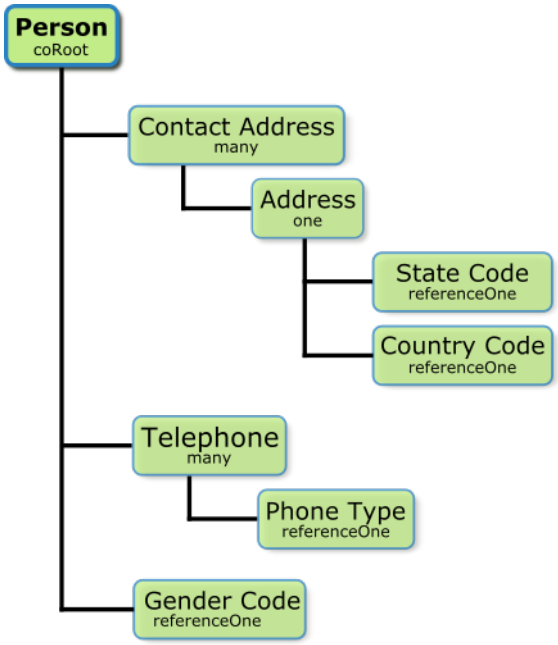
Use the following APIs to add the details of your employees:

1. Get information about the schema. Use the Get Metadata REST API to get information about the data structure of a business entity, including the structure, the list of fields, the field types, and the details of the lookup fields. Or, you can access the XML Schema Definition (XSD) files which describe what elements and attributes you can use. The XSD files are in the `http://<host>:<port>/cmx/csfiles` location.
2. Create a record. Use the Create Record REST API to create the record.
3. Read the data from the record that you have added. Use the Read Record REST API to retrieve the data from the record.

Person Business Entity Structure

We will add a Person record by using REST APIs. The Person root node is the uppermost node in the Person business entity structure. Under the Person root node, there are nodes for the employee details such as gender, address, and phone.

The following image shows the structure of the Person business entity:



Person is the root node of the Person business entity. The node type, listed below the node name, indicates the relationship between the parent node and the child node. There is a one-to-one relationship between the Contact Address and the Address, which indicates that each contact address can only have one address associated with it. There is a one-to-many relationship between the Person and the Telephone, which indicates that a Person record can have many telephone number records associated with it. There is a one-to-one relationship between the Person and the Gender, which indicates that a person record can have only one gender value. The gender values reside in a lookup table. Similarly, the state code and country code values reside in lookup tables.

Step 1. Get Information about the Schema

Use the Get Metadata REST API to get information about a schema. The Get Metadata API returns the data structure of a business entity. The metadata lists the business entity fields, field types, and details of the lookup fields.

The Get Metadata URL has the following format:

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

The following sample request retrieves metadata information for the Person business entity:

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

Get Metadata Response

The following example shows some excerpts of the data structure of the Person business entity:


```
{
    "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
```

```
    }
}
```

Step 2. Create a Record

Use the Create Record REST API to create a record. The name of the business entity and the name of the source system are required parameters. Send data for the record in the request body.

The Create Record URL has the following format:

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

The `systemName` parameter is a required parameter and specifies the name of the source system.

The Person business entity has the Person root node, and the second level address, gender, and phone nodes.

The following sample request creates a Person record:

```
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"
                }
            }
        ]
    }
}
```

The request body specifies the following details of the Person record:

- First name.
- Last name.

- Gender.
- Address with the state code and the country code.
- Phone numbers along with the phone type, such as home phone and mobile phone.

Create Record Response

The following sample response shows the response after successfully creating a Person record:


```
{
    "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              "
                    }
                }
            ]
        }
    }
}
```


Note: The response body contains the record with the generated rowIds.

If you configure a workflow process to start when you create a record, the following things happen:

- Record is created in a pending state.
- Workflow process is started.
- Workflow process ID is returned in the response header.

If you do not configure a workflow process, then by default, the record is created as Active.

The API returns an interaction ID in the response header if you process the request using an interaction ID.

Step 3. Read the Record

Use the Read Record REST API to retrieve the details of a root record that you added. You can use the API to retrieve the details of the child records of a root record.

The Read Record URL has the following format:

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

Use the `depth` parameter to specify the number of child levels to return. Specify 2 to return the root node and its direct children, and 3 to return the root node, direct children, and grandchildren. Use the following URL to return the details of the child records:

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

The following sample request returns the details of the root node, the direct children, and the grandchildren:

```
GET http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/Person/658248?depth=3
```

The request returns the details of an active record.

Note: If a workflow is started when you create a record, the record created is in pending state. By default, the Read Record request reads active records. Use the `recordStates` parameter to specify the pending state of the record.

The following sample request reads the details of a pending record:

```
GET http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/Person/658248?
depth=3&recordStates=PENDING
```

Read the Record Response

The following sample response shows the details of the record that you added:


```
{
    "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"
                }
            }
        ]
    }
}
```

APPENDIX B

Using REST APIs to Upload Files

This appendix includes the following topics:

Using REST APIs to Upload Files Overview

You can use the REST APIs to upload files to a storage type. After you upload a file, you can attach the file to a record or a task, or use the file to localize the Data Director user interface.

Based on how you want to use the files, the combination of REST APIs, the file components, and the storage type you use might differ. For example, to attach files to records or tasks, you create the metadata for the file and upload the file to a temporary storage. After you upload the file, you can attach the file to a record in a database or attach the file to a task in the BPM storage. To localize the Data Director user interface, you download the ZIP file, modify the zipped files, and then upload the modified ZIP bundle to the bundle storage.

REST APIs for Files

You can use a set of general purpose REST APIs to upload and manage files for attachments or localization.

The following table lists the REST APIs for files:

REST API	Description	Supported Storage Type
List File Metadata	Returns list of file metadata in a storage.	BPM or TEMP
Create File Metadata	Creates metadata for a file in a storage.	DB or TEMP

REST API	Description	Supported Storage Type
Get File Metadata	Returns the metadata for a file.	BPM, BUNDLE, DB, or TEMP
Update File Metadata	Updates the metadata for a file.	DB or TEMP
Upload File Content	Uploads the content for a file to a storage.	BUNDLE, DB, or TEMP
Get File Content	Downloads the content for a file.	BPM, BUNDLE, DB, or TEMP
Delete File	Deletes a file in a storage, including components associated to the file such as the file metadata or content.	BUNDLE, DB, or TEMP

File Components

To attach files to records or tasks, create metadata for the file and then upload the file content. To localize the Data Director user interface, you download the resource bundle file and then upload the modified resource bundle file.

File metadata

Information about the file, such as the file name, file type, and content type. Depending on your storage type, you might need to include other parameters, such as creator, create time, and upload date.

File content

The content for the file. For example, the text, image, document, or resource bundle.

Storage Types

Upload and store files in a supported storage implementation. The storage type that you use depends on whether you want to localize the Data Director user interface or attach files to records or tasks.

The following list describes the supported storage types:

BPM

Stores files attached to tasks together with the task data. When you attach a file to a task, the process stores the file to a BPM storage from the TEMP storage.

Files stored in the BPM storage use the following file ID format: `taskId::filename`.

Note: To attach a file to triggered tasks or existing tasks, in the Provisioning tool, enable attachments for task triggers, task types, and task actions. For more information, see the *Multidomain MDM Provisioning Tool Guide*.

BUNDLE

Stores resource bundle files that localize the Data Director user interface.

Files stored in the BUNDLE storage use the following file ID format: `besMetadata`.

DB

Stores file attachments for records in the C_REPOS_ATTACHMENTS table. When you attach a file to a record, the process stores the file to a DB storage from the TEMP storage.

Files stored in the DB storage use the following file ID format: `DB_<RowID>`.

Note: To attach a file to a record, in the Provisioning tool, configure a field with `FileAttachment` as the data type. For more information about configuring the data type, see the *Multidomain MDM Provisioning Tool Guide*.

TEMP

Temporarily stores files in the C_REPOS_ATTACHMENTS table and marks the files as TEMP. Files are deleted from the TEMP storage after they are successfully uploaded to the BPM or DB storage or after a preconfigured expiration time.

Files stored in the TEMP storage use the following file ID format: `TEMP_<ROWID_ATTACHMENT>`.

For more information about configuring the expiration time, see the *Multidomain MDM Configuration Guide*.

Attaching Files to Records

Before you attach a file to a record, create the metadata of the file and then upload the file to the temporary storage.

1. To create the metadata of a file, use the Create File Metadata REST API with TEMP as the storage type. For example, the following request creates the metadata for the `Document_3.pdf` file:

```
POST http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/TEMP
Content-Type: application/json
{
    "fileName": "Document_3.pdf",
    "fileType": "pdf",
    "fileContentType": "application/pdf"
}
```

   Note: Always create the file metadata in the TEMP storage.

   The Create File Metadata REST API returns an ID for the file. The file ID is in the following format: `<Storage Type>_<RowID>`. Where the RowID refers to the row ID of the file that you upload to the storage.

   In the example, the API call returns the following ID for the `Document_3.pdf` file: `TEMP_SVR1.OJU3`

   You can use the file ID to upload, attach, update, download, and delete the file.

2. To upload the file, use the Upload File Content REST API with TEMP as the storage type. For example, the following request uploads the file to the TEMP storage:

```
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)>
```

   Note: After you upload a file, the TEMP storage stores the file for a pre-configured period of 60 minutes. You must attach the file to a record before the pre-configured period expires.

3. To create a record and attach the file to a new record, use the Create Record REST API.

For example, the following request creates a record and attaches the file with the file ID, TEMP_SVR1.OJU3:

```
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"
         }
      ]
   }
}
```

Note: When you attach a file to a record, the process stores the file to the database. The ID of the file changes to DB_<RowID>, where DB indicates that the file is stored in the database.

4. To replace a file attached to a record, use the Upload File Content REST API with DB as the storage type. For example, the following request replaces the attached file with the file ID, DB_SVR1.OJU3, in the database:

```
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)>
```

Note: The storage type in the request URL is DB.

5. To edit the file metadata after you attach a file to a record, use the Update File Metadata REST API with DB as the storage type.
For example, the following request updates the file metadata of a file associated with the file ID, DB_SVR1.OJU3, in the DB storage:

```
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. To download a file attached to a record, use the GET File Content REST API with DB as the storage type. For example, the following request downloads a file associated with the file ID, DB_SVR1.OJU3, from the DB storage:

```
GET http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/DB/DB_SVR1.OJU3/content
```

7. To delete a file attached to a record, use the Delete File REST API with DB as the storage type.
For example, the following request deletes a file associated with the file ID, DB_SVR1.OJU3, from the DB storage:

```
DELETE http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/DB/DB_SVR1.OJU3
```

Attaching Files to Tasks

Create the metadata for a file and then upload the file content to a temporary storage. After you upload the file, attach the file to a triggered task or an existing task.

Note: To attach a file to triggered tasks or existing tasks, in the Provisioning tool, enable attachments for task triggers, task types, and task actions. For more information, see the *Multidomain MDM Provisioning Tool Guide*.

1. To create the metadata of a file, use the Create File Metadata REST API with TEMP as the storage type. For example, the following request creates the metadata for the `file1.txt` file:

```
POST http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/TEMP

{
    "fileName": "file1.txt",
    "fileType": "text",
    "fileContentType": "text/plain"
}
```

   Note: Always create the file metadata in the TEMP storage.

   The Create File Metadata REST API returns an ID for the file. The file ID is in the following format: `<Storage Type>_<RowID>`. Where the RowID refers to the row ID of the file that you upload to the storage.
   In the example, the API call returns the following ID for `file1.txt`: `TEMP_SVR1.1VDVS`

   You can use the file ID to upload, attach, update, and delete the file.

2. To upload the file, use the Upload File Content REST API with TEMP as the storage type.
   For example, the following request uploads the file to the TEMP storage:

```
PUT http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/TEMP/TEMP_SVR1.1VDVS/
content

Test attachment content: file 1
```

   Note: After you upload a file, the TEMP storage stores the file for a pre-configured period of 60 minutes. You must attach the file to a task before the pre-configured period expires.

3. Attach the file to the task that is triggered when you manage records.

   - To attach the file to the task that is triggered when you create a record, use the Create Business Entity REST API with the `taskattachments` parameter.
     For example, the following request creates a record and attaches the file with the file ID `TEMP_SVR1.1VDVS`:

```
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"
               }
          ]
     }
}
```

   - To attach the file to the task that is triggered when you update a record, use the Update Business Entity REST API with the `taskattachments` parameter.

For example, the following request updates a record and attaches the file with the file ID `TEMP_SVR1.1VDVS`:

```
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"
    }
}
```

- To attach the file to the task that is triggered when you merge a record, use the Merge Business Entity REST API with the `taskattachments` parameter.
  For example, the following request merges a record and attaches the file with the file ID `TEMP_SVR1.1VDVS`:

```
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"
    }
  }
}
```

- To attach the file to the task that is triggered when you unmerge a record, use the Unmerge Business Entity REST API with the `taskattachments` parameter.
  For example, the following request unmerges a record and attaches the file with the file ID `TEMP_SVR1.1VDVS`:

```
POST http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/2478248?
action=unmerge&taskAttachments=TEMP_SVR1.1VDVS
{
        rowid: "4880369"
}
```

4. Attach the file to an existing task.

   - To attach the file when you update a task, use the Update Task REST API with `attachments` in the request body.
     For example, the following request updates a task and attaches the file with the file ID `TEMP_SVR1.1VDVS`:

```
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"
            }
        ],
        ...
    }
```

- To attach the file when you execute a task action, use the Execute Task Action REST API with `attachments` in the request body.
  For example, the following request executes a task action and attaches the file with the file ID `TEMP_SVR1.1VDVS`:

```
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"
        }
    ],
    ...
}
```

After you attach a file to a task, the process moves the file from the TEMP storage and stores the file together with the task data in the BPM storage. The ID of the file changes to `taskId::filename`.

Uploading Resource Bundle Files

To localize the Data Director user interface, download the resource bundle ZIP file, modify the files in the ZIP file, and then upload the modified ZIP file to the bundle storage.

1. To download the resource bundle ZIP file, use the Get File Content REST API with BUNDLE as the storage type.
   For example, the following request downloads the resource bundle ZIP file:

```
GET http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/BUNDLE/besMetadata/
content
```

2. Modify the ZIP file by adding language specific bundle files.
   For example, to localize field names, labels, and table names to Russian, add the `besMetadata_ru.properties` file.
3. To upload the modified resource bundle ZIP file, use the Upload File Content REST API with BUNDLE as the storage type.
   For example, the following request uploads the resource bundle ZIP file:

```
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
```

APPENDIX C

Using REST APIs to Manage Reports

This appendix includes the following topics:

Using REST APIs to Manage Reports Overview

You can use REST APIs to manage the out of the box reports and custom reports that collect information about your master data. Later, in the Provisioning tool, you can populate Chart components with the report data. Then in Data Director, users can view the charts to analyze data about their master data.

The out of the box reports collect information about predefined criteria. If you want to collect other information about your master data, use custom reports.

REST APIs for Reports

You can use a set of REST APIs to manage reports.

The following table lists the REST APIs for reports:

REST API	Description
List Reports	Returns a list of registered reports and the configuration of the reports.
Get Report Configuration and Data	Returns the report configuration and data.
Get Report Configuration and Drilldown Reports	Returns the report configuration and associated drilldown reports.
Register Report	Registers custom reports. Returns the report ID for the report, which you can use in other APIs.
Update Report Configuration	Updates the configuration of a report.
Add or Update Report Data	Adds or updates data entries in a report.
Delete Report	Deletes a report.
Run Report Update Job	Starts the batch jobs associated with the out of the box reports.
Get Status of Report Update Job	Returns the status of a report update job.

Report Configuration

The report configuration consists of multiple parameters, such as the report name, description, dimensions, and metric name. The report configuration of the out of the box reports is preconfigured. If you want to use custom reports, you must specify the report configuration of your custom reports.

The following table describes the parameters in a report configuration:

Parameter	Description
ROWID_RPT_CONFIG	ID of report.
DIMENSION_NAME_1	The name of a dimension of data in the report. For example, a dimension might be User Name, Review task type, or business entity type.
DIMENSION_NAME_2	Optional. The name of a dimension of data in the report. For example, a dimension might be User Name, Review task type, or business entity type.
TIMEPERIOD_NAME	Optional. Name of the period of time to which the data applies. For example, you might specify `Month`.
RPT_NAME	Name of the report.

Parameter	Description
METRIC_NAME	Label for the type of data collected by the report. By default, the value specified for this parameter appears as the name of the y-axis in the chart. For example, you might have a report that collects information about closed tasks by user. You might use the metric name `Number of closed tasks`.
RPT_DESC	Description of the report.
RPT_TYPE	Indicates whether the report is a drilldown report. If the value is `null`, then the report is a root report. If the value is anything other than `null`, then the report is a drilldown report.

Report Data

Add data to a report so that you can populate charts with the report data. For the out of the box reports, you use the report update jobs associated to the reports to collect and add report data. If you use custom reports, you must manually add the data entries in the custom reports.

The following table describes the parameters you must configure for each data entry in a report:

Parameter	Description
DIMENSION_NAME_1	The name of a dimension of data in the report. For example, a dimension might be User Name, Review task type, or business entity type.
DIMENSION_NAME_2	Optional. The name of a dimension of data in the report. For example, a dimension might be User Name, Review task type, or business entity type.
TIMEPERIOD_NAME	Optional. Name of the period of time to which the data applies. For example, you might specify `Month`.
METRIC_VALUE	Number that represents the dimensions of data. For example, you might have a report that collects information about closed tasks by user. One dimension of the report is the user John Smith, and another dimension is the Review task type. The metric value might be `5`. This means that John Smith closed five tasks of the Review task type.
DRILLDOWN_RPT_ID	ID of a drilldown report. If the value is `null`, then there is no drilldown report configured.

The following example shows three data entries in a report:

```
[
  {
    "DIMENSION_VALUE_1": "High",
    "DIMENSION_VALUE_2": "AVOSBeMerge",
    "TIMEPERIOD_VALUE": "null",
    "METRIC_VALUE": "3",
    "DRILLDOWN_RPT_ID": "null"
 },
  {
    "DIMENSION_VALUE_1": "High",
    "DIMENSION_VALUE_2": "AVOSBeReviewNoApprove",
    "TIMEPERIOD_VALUE": "null",
    "METRIC_VALUE": "0",
    "DRILLDOWN_RPT_ID": "null"
 },
```

```
{
   "DIMENSION_VALUE_1": "High",
   "DIMENSION_VALUE_2": "AVOSBeUpdate",
   "TIMEPERIOD_VALUE": "null",
   "METRIC_VALUE": "0",
   "DRILLDOWN_RPT_ID": "null"
}
   ...
]
```


Drilldown Reports

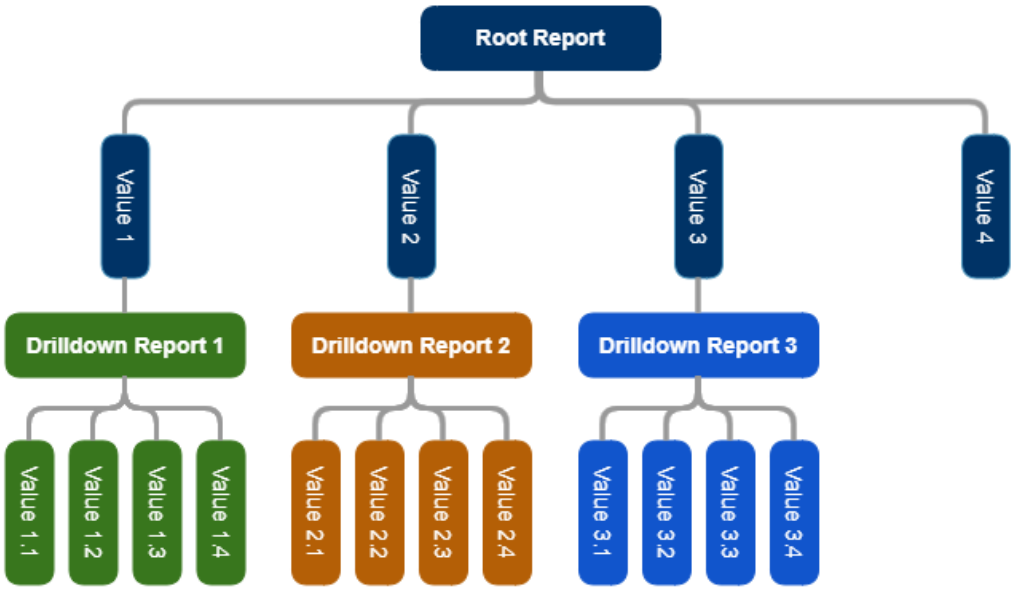
A drilldown report contains another layer of data about a data entry in a report. You can use the drilldown reports associated to the out of the box reports or configure drilldown reports for your custom reports.

When you create custom reports, you register a root report. You can create and associate drilldown reports to data entries in the root report or data entries in other drilldown reports.

For example, you might have a report that collects information about the number tasks per each status. You can associate a drilldown report to the number of Closed tasks. The drilldown report contains the number of Closed tasks for each user. You can also associate a drilldown report to the number of Approved tasks. The drilldown report contains the number of Approved tasks for each user.

In Data Director, users might see a parent chart that shows the number of tasks per each status. Then when they select the number of Closed tasks, they drill down to a drilldown chart that shows the number of Closed tasks for each user.

The following image shows the drilldown report structure:



For information about the drilldown reports associated to the out of the box reports, see "Out of the Box Reports" on page 342.

Out of the Box Reports

The out of the box reports collect information about your master data. Some out of the box reports also have drilldown reports. Drilldown reports contain another layer of data about the report data.

For example, the Tasks Overview by Status report collects information about the number of tasks by status. Drilldown reports associated to the report shows the users assigned tasks for each status.

To use the out of the box reports, you must run the report update job associated to the report that you want to use. The report update job registers the report and collects information about your master data based on the report criteria. Then in the Provisioning tool, you can populate Chart components with the report data. If there are drilldown reports associated to the out of the box report, you can link your chart to a drilldown chart.

The following table describes the out of the box reports and lists the ID of each report:

Report ID	Report	Description
MDM.RPT.1	Tasks Overview by Status	Displays the total number of tasks and breaks down the tasks by status. You can drill down to users assigned for each task status.
MDM.RPT.2	Task Overview by Priority	Displays the total number of tasks and breaks down the tasks by priority. You can drill down to users assigned for each task priority.
MDM.RPT.4	Business Entities Added by Year	Displays the number of records added for each business entity type and breaks down the records by year.
MDM.RPT.5	Customer Onboarding Time	Displays how long it took to qualify records.
MDM.RPT.6	Business Entities by Source Systems	Displays the total number of records for each business entity type that comes from each source system. The detailed report breaks down the records for each pair of business entity type and source system by year.
MDM.RPT.7	Assigned Tasks by Users	Displays the total number of tasks assigned to each user.
MDM.RPT.8	Open Tasks by User Roles	Displays the total number of tasks that are open by user roles.
MDM.RPT.10	Closed Tasks by Users	Displays the total number of tasks that are closed by each user.

Managing the Out of the Box Reports

To use the out of the box reports to collect data about your master data, run the update jobs associated to the out of the box reports. After you run the update jobs, in the Provisioning tool, you can populate Chart components with the report data.

Before you begin, review the out of the box reports and determine the reports that you want to use. For more information, see "Out of the Box Reports" on page 342.

1. To run the update job associated to the out of the box report, in the browser address bar, enter the REST URL from the Run Report Update Job REST API.
   For example, the following REST URL runs the update job associated to the Tasks Overview by Status report:

   ```
   http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/data/collect/MDM.RPT.1
   ```

You registered the report and started collecting data based on the report criteria. The following sample response shows the status of the report update job:

```
{
  "status": "PROCESSING"
}
```

2. To retrieve the status of the report update job, in the browser address bar, enter the REST URL from the Get Status of Report Update Job REST API.
   For example, following REST URL retrieves the status of the update job:

   ```
   http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/data/collect/MDM.RPT.1/status
   ```

   The following sample response shows the status of the report update job:

   ```
   {
     "status": "COMPLETED_SUCCESSFULLY",
     "jobId": "SVR1.2X9N1 ",
     "lastUpdateDate": "2019-10-31T14:37:11.846-04:00",
     "startRunDate": "2019-10-31T14:37:09.120-04:00"
   }
   ```

3. To retrieve the report configuration and data, in the browser address bar, enter the REST URL from Get Report Configuration and Data REST API.
   For example, the following REST URL retrieves the report configuration and data:

   ```
   http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/data/MDM.RPT.1
   ```

   The following sample response shows the report configuration and data:

   ```
   {
     "metadata":{
       "fieldsMetadata":[
         "DIMENSION_VALUE_1",
         "DIMENSION_VALUE_2",
         "TIMEPERIOD_VALUE",
         "METRIC_VALUE",
         "DRILLDOWN_RPT_ID"
       ],
       "ROWID_RPT_CONFIG":"MDM.RPT.1     ",
       "DIMENSION_NAME_1":"Task Status",
       "METRIC_NAME":"Number of tasks",
       "DIMENSION_NAME_2":"Task Type",
       "TIMEPERIOD_NAME":null,
       "RPT_NAME":"Task Status/Type Report",
       "RPT_DESC":"Metrics for task status/type",
       "RPT_TYPE":null
     },
     "data":[
       [
         "Overdue",
         "AVOSBeMerge",
         null,
         "36",
         "SVR1.4FCA2    "
       ],
       [
         "Overdue",
         "AVOSBeReviewNoApprove",
         null,
         "189",
         "SVR1.4FC8E    "
       ],
       [
         "Overdue",
         "AVOSBeUpdate",
         null,
         "1",
         "SVR1.4FCAQ    "
       ],
       [
   ```

```
        "Overdue",
        "AVOSBeFinalReview",
        null,
        "1",
        "SVR1.4FC9W    "
      ]
                  ...
  ]
}
```

4. To retrieve the report configuration and drilldown reports, use the Get Report Configuration and Drilldown Reports REST API.
   For example, the following REST URL retrieves the report configuration and drilldown reports:

   ```
   http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/meta/MDM.RPT.1
   ```

   The following sample response shows the report configuration and drilldown reports:

   ```
   {
     "ROWID_RPT_CONFIG":"MDM.RPT.1",
     "DIMENSION_NAME_1":"Task Status",
     "METRIC_NAME":"Number of tasks",
     "DIMENSION_NAME_2":"Task Type",
     "TIMEPERIOD_NAME":null,
     "RPT_NAME":"Task Status/Type Report",
     "RPT_DESC":"Metrics for task status/type",
     "RPT_TYPE":null,
     "DRILLDOWN":[
       {
         "RPT_NAME":"Task's Owner per Task Status and Task Type",
         "DETAILED_RPT_IDS":[
           "SVR1.4FCA3",
           "SVR1.4FCA4",
           "SVR1.4FCA5",
           "SVR1.4FCA6",
           "SVR1.4FCA7",
           "SVR1.4FC8F",
           "SVR1.4FC8G",
           "SVR1.4FC8H",
           "SVR1.4FC8I",
           "SVR1.4FC8J",
           "SVR1.4FCAR",
           "SVR1.4FCAS",
           "SVR1.4FCAT",
           "SVR1.4FCAU",
           "SVR1.4FCAV",
           "SVR1.4FC9X",
           "SVR1.4FC9Y",
           "SVR1.4FC9Z",
           "SVR1.4FCA0",
           "SVR1.4FCA1",
           "SVR1.4FC9L",
           "SVR1.4FC9M",
           "SVR1.4FC9N",
           "SVR1.4FC9O",
           "SVR1.4FC9P",
           "SVR1.4FC8L",
           "SVR1.4FC8M",
           "SVR1.4FC8N",
           "SVR1.4FC8O"
         ],
         "CONFIG_RPT_IDS":[
           "SVR1.4FCA2",
           "SVR1.4FC8E",
           "SVR1.4FCAQ",
           "SVR1.4FC9W",
           "SVR1.4FC9K",
           "SVR1.4FC8K",
           "SVR1.4FCAW",
           "SVR1.4FC9E",
           "SVR1.4FC88",
   ```

```
            "SVR1.4FC8W",
            "SVR1.4FC82",
            "SVR1.4FC92",
            "SVR1.4FC98",
            "SVR1.4FCAE",
            "SVR1.4FCA8",
            "SVR1.4FCB8",
            "SVR1.4FC9Q",
            "SVR1.4FCAK",
            "SVR1.4FC8Q",
            "SVR1.4FCB2"
          ]
        }
    ]
}
```

Custom Reports

If the out of the box reports do not meet your business requirements, you can configure custom reports to collect other information about your master data. For example, you might want to collect information about the number of closed tasks by priority.

To use custom reports, you must configure and register the reports, and then add data entries to the report. To associate drilldown reports to the report data, you configure and register the drilldown reports. Then you can associate the drilldown report to a data entry in the custom report.

Managing Custom Reports

Register and configure the custom reports that you want to use to collect information about your master data. Then in the Provisioning tool, you can populate the Chart component with the report data.

Before you begin, determine the data you want to collect.

1. To review the registered reports, use the List Reports REST API.
   For example, the following request lists the registered root reports:

   ```
   GET http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/list
   ```

2. To register a custom report, use the Register Report REST API.
   For example, the following request registers a custom report:

   ```
   POST http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/list
   {
     "DIMENSION_NAME_1": "Task Priority",
     "DIMENSION_NAME_2": "Task Type",
     "TIMEPERIOD_NAME": "null",
     "RPT_NAME": "Task Priority/Type Report",
     "RPT_DESC": "Metrics for task status/type",
     "METRIC_NAME": "Number of tasks"
    }
   ```

   The following sample response shows the registered report:

   ```
   {
     "ROWID_RPT_CONFIG": "SVR1.2X9N0",
     "DIMENSION_NAME_1": "Task Priority",
     "DIMENSION_NAME_2": "Task Type",
     "TIMEPERIOD_NAME": "null",
     "RPT_NAME": "Task Priority/Type Report",
     "RPT_DESC": "Metrics for task status/type",
     "METRIC_NAME": "Number of tasks",
   ```

```
  "RPT_TYPE": "null"
}
```

The request returns the report ID in the `ROWID_RPT_CONFIG` parameter.

3. To add data entries in a report, use the Add or Update Report Data REST API.
   For example, the following request adds data entries in a report:

```
POST http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/data/SVR1.2X9N0
[
  {
    "DIMENSION_VALUE_1": "High",
    "DIMENSION_VALUE_2": "AVOSBeMerge",
    "TIMEPERIOD_VALUE": "null",
    "METRIC_VALUE": "3",
    "DRILLDOWN_RPT_ID": "null"
 },
  {
    "DIMENSION_VALUE_1": "High",
    "DIMENSION_VALUE_2": "AVOSBeReviewNoApprove",
    "TIMEPERIOD_VALUE": "null",
    "METRIC_VALUE": "0",
    "DRILLDOWN_RPT_ID": "null"
 },
 {
    "DIMENSION_VALUE_1": "High",
    "DIMENSION_VALUE_2": "AVOSBeUpdate",
    "TIMEPERIOD_VALUE": "null",
    "METRIC_VALUE": "0",
    "DRILLDOWN_RPT_ID": "null"
 }
    ...
]
```

4. To retrieve the report configuration and data, use the Get Report Configuration and Data REST API.
   For example, the following request retrieves the report configuration and data:

```
GET http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/data/SVR1.2X9N0
```

   The following sample response shows the report configuration and the data:

```
{
  "metadata":{
    "fieldsMetadata":[
      "DIMENSION_VALUE_1",
      "DIMENSION_VALUE_2",
      "TIMEPERIOD_VALUE",
      "METRIC_VALUE",
      "DRILLDOWN_RPT_ID"
    ],
    "ROWID_RPT_CONFIG": "MDM.RPT.2",
    "DIMENSION_NAME_1": "Task Priority",
    "METRIC_NAME": "Number of tasks",
    "DIMENSION_NAME_2": "Task Type",
    "TIMEPERIOD_NAME": "null",
    "RPT_NAME": "Task Priority/Type Report",
    "RPT_DESC": "Metrics for task status/type",
    "RPT_TYPE": "null"
  },
  "data":[
    [
      "High",
      "AVOSBeMerge",
      "null",
      "3",
      "SVR1.48P5G"
    ],
    [
      "High",
      "AVOSBeReviewNoApprove",
      "null",
```

```
      "0",
      "null"
    ],
    [
      "High",
      "AVOSBeUpdate",
      null,
      "0",
      "null"
    ],
    [
      "High",
      "AVOSBeFinalReview",
      null,
      "0",
      "null"
    ]
    ...
  ]
}
```

5. To delete a report, use the Delete Report REST API.
   For example, the following request deletes a report:

```
DELETE http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/data/SVR1.2X9N0
```

Managing Custom Reports with Drilldown Reports

Register the custom reports that you want to use and add data entries to your report. Then register the drilldown reports and add data entries to the drilldown reports. When you add or update data entries in your root reports, you can associate your drilldown reports to the data entries. Then in the Provisioning tool, you can populate the Chart component with the report data and link the root chart to a drilldown chart.

Before you begin, determine the data you want to collect. Also, determine the drilldown reports you want to configure and associate to your data entries in your custom report.

1. To review the registered reports, use the List Reports REST API.
   For example, the following request lists the registered root reports and drilldown reports:

```
GET http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/list?show=all
```

2. To register a custom report, use the Register Report REST API.
   For example, the following request registers a custom report:

```
POST http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/list
{
  "DIMENSION_NAME_1": "Task Priority",
  "DIMENSION_NAME_2": "Task Type",
  "TIMEPERIOD_NAME": "null",
  "RPT_NAME": "Task Priority/Type Report",
  "RPT_DESC": "Metrics for task status/type",
  "METRIC_NAME": "Number of tasks"
 }
```

   The following sample response shows the registered report:

```
{
  "ROWID_RPT_CONFIG": "SVR1.2X9N0",
  "DIMENSION_NAME_1": "Task Priority",
  "DIMENSION_NAME_2": "Task Type",
  "TIMEPERIOD_NAME": "null",
  "RPT_NAME": "Task Priority/Type Report",
  "RPT_DESC": "Metrics for task status/type",
  "METRIC_NAME": "Number of tasks",
  "RPT_TYPE": "null"
}
```

   The request returns the report ID in the `ROWID_RPT_CONFIG` parameter.

3. To add data entries in a root report, use the Add or Update Report Data REST API.

   Later, you can associate a drilldown report to a data entry.

   For example, the following request adds data entries in a report:

```
POST http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/data/SVR1.2X9N0
[
  {
    "DIMENSION_VALUE_1": "High",
    "DIMENSION_VALUE_2": "AVOSBeMerge",
    "TIMEPERIOD_VALUE": "null",
    "METRIC_VALUE": "3",
    "DRILLDOWN_RPT_ID": "null"
  },
  {
    "DIMENSION_VALUE_1": "High",
    "DIMENSION_VALUE_2": "AVOSBeReviewNoApprove",
    "TIMEPERIOD_VALUE": "null",
    "METRIC_VALUE": "0",
    "DRILLDOWN_RPT_ID": "null"
 },
 {
    "DIMENSION_VALUE_1": "High",
    "DIMENSION_VALUE_2": "AVOSBeUpdate",
    "TIMEPERIOD_VALUE": "null",
    "METRIC_VALUE": "0",
    "DRILLDOWN_RPT_ID": "null"
 }
 ...
]
```

4. To register a drilldown report, use the Register Report REST API.
   For example, the following request registers the drilldown report:

```
POST http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/list
{
  "DIMENSION_NAME_1": "Task's Owner",
  "DIMENSION_NAME_2": "null",
  "TIMEPERIOD_NAME": "null",
  "RPT_NAME": "Task's Owner per Task Priority and Task Type",
  "RPT_DESC": "Number of tasks for each users for Task Priority/Task Type",
  "METRIC_NAME": "Number of Tasks"
 }
```

   The following sample response shows the registered drilldown report:

```
{
  "ROWID_RPT_CONFIG": "SVR1.48P5G",
  "DIMENSION_NAME_1": "Task's Owner",
  "DIMENSION_NAME_2": "null",
  "TIMEPERIOD_NAME": "null",
  "RPT_NAME": "Task's Owner per Task Priority and Task Type",
  "RPT_DESC": "Number of tasks for each users for Task Priority/Task Type",
  "METRIC_NAME": "Number of Tasks",
  "RPT_TYPE": "High/AVOSBeMerge"
}
```

   The request returns the drilldown report ID in the `ROWID_RPT_CONFIG` parameter.

5. To add data entries in a drilldown report, use the Add or Update Report Data REST API.
   For example, the following request adds data entries in a drilldown report:

```
POST http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/data/SVR1.48P5G
[
  {
    "DIMENSION_VALUE_1": "admin",
    "DIMENSION_VALUE_2": "null",
    "TIMEPERIOD_VALUE": "null",
    "METRIC_VALUE": "0",
    "DRILLDOWN_RPT_ID": "null"
 },
```

```
 {
   "DIMENSION_VALUE_1": "srmgr2",
   "DIMENSION_VALUE_2": "null",
   "TIMEPERIOD_VALUE": "null",
   "METRIC_VALUE": "0",
   "DRILLDOWN_RPT_ID": "null"
 },
 {
   "DIMENSION_VALUE_1": "mgr2",
   "DIMENSION_VALUE_2": "null",
   "TIMEPERIOD_VALUE": "null",
   "METRIC_VALUE": "0",
   "DRILLDOWN_RPT_ID": "null"
 }
]
```

6. To associate a drilldown report to a data entry in a report, use the Add or Update Report Data REST API. Specify the drilldown report ID in `DRILLDOWN_RPT_ID` parameter.
   For example, the following request associates a drilldown report to the data entry:

```
POST http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/data/SVR1.2X9N0
[
  {
    "DIMENSION_VALUE_1": "High",
    "DIMENSION_VALUE_2": "AVOSBeMerge",
    "TIMEPERIOD_VALUE": "null",
    "METRIC_VALUE": "5",
    "DRILLDOWN_RPT_ID": "SVR1.48P5G"
  }
]
```

7. To retrieve the report configuration and data, use the Get Report Configuration and Data REST API.
   For example, the following request retrieves the report configuration and data:

```
GET http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/data/SVR1.2X9N0
```

   The following sample response shows the report configuration and the data:

```
{
  "metadata":{
    "fieldsMetadata":[
      "DIMENSION_VALUE_1",
      "DIMENSION_VALUE_2",
      "TIMEPERIOD_VALUE",
      "METRIC_VALUE",
      "DRILLDOWN_RPT_ID"
    ],
    "ROWID_RPT_CONFIG": "SVR1.2X9N0",
    "DIMENSION_NAME_1": "Task Priority",
    "METRIC_NAME": "Number of tasks",
    "DIMENSION_NAME_2": "Task Type",
    "TIMEPERIOD_NAME": "null",
    "RPT_NAME": "Task Priority/Type Report",
    "RPT_DESC": "Metrics for task status/type",
    "RPT_TYPE": "null"
  },
  "data":[
    [
      "High",
      "AVOSBeMerge",
      "null",
      "3",
      "SVR1.48P5G"
    ],
    [
      "High",
      "AVOSBeReviewNoApprove",
      "null",
      "0",
      "null"
    ],
```

```
      [
        "High",
        "AVOSBeUpdate",
        "null",
        "0",
        "null"
      ],
      [
        "High",
        "AVOSBeFinalReview",
        "null",
        "0",
        "null"
      ]
      ...
    ]
}
```

8. To retrieve the registered reports, including your custom reports and drilldown reports, use the List Reports REST API.
   For example, the following request lists the registered root reports and drilldown reports:

   ```
   GET http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/list?show=all
   ```

   The following sample response shows the registered root reports and drilldown reports:

   ```
   [
     {
       "ROWID_RPT_CONFIG": "SVR1.2X9N0",
       "DIMENSION_NAME_1": "Task Priority",
       "METRIC_NAME": "Number of tasks",
       "DIMENSION_NAME_2": "Task Type",
       "TIMEPERIOD_NAME": "null",
       "RPT_NAME": "Task Priority/Type Report",
       "RPT_DESC": "Metrics for task status/type",
       "RPT_TYPE": "null"
     },
     {
       "ROWID_RPT_CONFIG": "SVR1.48P5G",
       "DIMENSION_NAME_1": "Task's Owner",
       "DIMENSION_NAME_2": "null",
       "TIMEPERIOD_NAME": "null",
       "RPT_NAME": "Task's Owner per Task Priority and Task Type",
       "RPT_DESC": "Number of tasks for each users for Task Priority/Task Type",
       "METRIC_NAME": "Number of Tasks",
       "RPT_TYPE": "High/AVOSBeMerge"
     }
   ]
   ```

9. To delete a report, use the Delete Report REST API.
   For example, the following request deletes a report:

   ```
   DELETE http://localhost:8080/cmx/report/localhost-orcl-DS_UI1/data/SVR1.2X9N0
   ```

Troubleshooting Report APIs

If you encounter issues with the report APIs, use the following information to troubleshoot.

Failed to run the update jobs associated to the out of the box reports.

The report services might not be connected to ActiveVOS. If you edited the ActiveVOS EAR files to customize the JNDI lookup string, you must add the `activevos.jndi` property to the `cmxserver.properties` file and specify the custom JNDI lookup string. For more information, see the *Multidomain MDM Configuration Guide*.

INDEX

S

T

U

W