Informatica®

Informatica® Multidomain MDM
10.3

비즈니스 항목 서비스 가이드

Informatica Multidomain MDM 비즈니스 항목 서비스 가이드
10.3
2018년9월

발행 날짜: 2019-05-28

목차

서문

*Multidomain MDM 비즈니스 항목 서비스 가이드*를 시작합니다. 이 가이드는 Informatica ® MDM Hub에서 비즈니스 항목에 대한 작업을 수행하는 비즈니스 항목 서비스 호출을 만드는 방법에 대해 설명합니다.

이 가이드는 MDM Hub에 대한 비즈니스 항목 서비스 호출을 만들 수 있는 사용자 지정 사용자 인터페이스 구성을 담당하는 기술 전문가를 대상으로 합니다.

Informatica 리소스

Informatica 네트워크

Informatica 네트워크는 Informatica 글로벌 고객 지원, Informatica 기술 자료 및 기타 제품 리소스를 호스팅합니다. Informatica 네트워크에 액세스하려면 https://network.informatica.com을 방문하십시오.

회원이 되면 다음과 같은 기능을 이용할 수 있습니다.

- 모든 Informatica 리소스를 한 곳에서 액세스
- 기술 자료에서 설명서, FAQ, 모범 사례 등의 제품 리소스를 검색합니다.
- 제품 사용 가능 여부에 대한 정보를 봅니다.
- 지원 사례 검토
- 거주 지역의 Informatica 사용자 그룹 네트워크를 검색하고 동료와 협업 관계 유지

Informatica 기술 자료

Informatica 기술 자료를 사용하면 Informatica 네트워크에서 설명서, 방법 문서, 모범 사례 및 PAM 같은 제품 리소스를 검색할 수 있습니다.

기술 자료에 액세스하려면 https://kb.informatica.com을 방문하십시오. 기술 자료에 대한 질문, 의견 또는 아이디어가 있는 경우 KB_Feedback@informatica.com을 통해 Informatica 기술 자료 팀에 문의해 주시기 바랍니다.

Informatica 설명서

제품에 대한 최신 설명서를 가져오려면 Informatica 기술 자료 (https://kb.informatica.com/_layouts/ProductDocumentation/Page/ProductDocumentSearch.aspx)에서 검색해 보십시오.

이 설명서에 대한 질문, 의견 또는 아이디어가 있는 경우 전자 메일(infa_documentation@informatica.com)을 통해 Informatica 설명서 팀에 문의해 주시기 바랍니다.

Informatica Product Availability Matrix (PAM)

Product Availability Matrix (PAM)은 제품 릴리스에서 지원하는 운영 체제 버전, 데이터베이스 및 기타 데이터 소스 유형과 대상을 나타냅니다. Informatica 네트워크 회원은 https://network.informatica.com/community/informatica-network/product-availability-matrices 을 통해 PAM에 액세스할 수 있습니다.

Informatica Velocity

Informatica Velocity는 Informatica 전문 서비스업에서 개발한 팁과 모범 사례의 컬렉션입니다. 수백 개의 실제 데이터 관리 프로젝트 환경에서 개발된 Informatica Velocity는 성공적인 데이터 관리 솔루션을 계획, 개발, 배포 및 유지 관리하기 위해 전 세계 조직과 작업한 당사 컨설턴트의 총체적 지식을 나타냅니다.

Informatica 네트워크 회원은 http://velocity.informatica.com을 통해 Informatica Velocity 리소스에 액세스할 수 있습니다.

Informatica Velocity에 대한 질문, 주석 또는 아이디어가 있으시면 Informatica 전문 서비스업 (ips@informatica.com)에 문의하십시오.

Informatica Marketplace

Informatica Marketplace는 Informatica 구현을 확장, 확대 또는 개선하기 위한 솔루션을 찾을 수 있는 포럼입니다. Informatica 개발자와 파트너가 제공하는 수백 개의 솔루션을 활용하여 생산성을 향상시키고 프로젝트의 구현에 걸리는 시간을 줄일 수 있습니다. https://marketplace.informatica.com에서 Informatica Marketplace에 액세스할 수 있습니다.

Informatica 글로벌 고객 지원 센터

전화 또는 Informatica 네트워크의 온라인 지원을 통해 글로벌 지원 센터에 문의할 수 있습니다.

해당 지역의 Informatica 글로벌 고객 지원 전화 번호는 Informatica 웹 사이트 (http://www.informatica.com/us/services-and-training/support-services/global-support-centers)를 방문하여 찾을 수 있습니다.

Informatica 네트워크 회원인 경우에는 온라인 지원(http://network.informatica.com)을 사용할 수 있습니다.

제 1 장

비즈니스 항목 서비스 소개

이 장에 포함된 항목:

비즈니스 항목 서비스 개요

비즈니스 항목 서비스는 MDM Hub 코드를 실행하여 비즈니스 항목에서 기본 개체 레코드를 작성, 업데이트, 삭제 및 검색하는 작업 집합입니다. Java 코드 또는 JavaScript 코드를 실행하여 비즈니스 항목 서비스 호출을 만드는 사용자 지정 사용자 인터페이스를 개발할 수 있습니다.

예를 들어 비즈니스 항목 서비스를 작성하여 Dun 및 Bradstreet 데이터로 공급자 레코드를 확장할 수 있습니다. 공급자 레코드를 입력으로 가져오고, Dun 및 Bradstreet에서 일부 정보를 검색하여 레코드를 업데이트한 다음 업데이트된 공급자 레코드를 출력하도록 비즈니스 항목 서비스를 구성합니다.

비즈니스 항목의 기본 개체에는 다음과 같은 비즈니스 항목 서비스가 있습니다.

읽기

각 비즈니스 항목에는 읽기 작업을 수행하는 비즈니스 항목 서비스가 있습니다.

쓰기

각 비즈니스 항목에는 쓰기 작업을 수행하는 비즈니스 항목 서비스가 있습니다.

검색

검색 가능한 필드가 있는 모든 비즈니스 항목에는 검색 작업을 수행하는 비즈니스 항목 서비스가 있습니다.

예를 들어 Person 비즈니스 항목에 검색 가능한 필드가 있습니다. MDM Hub는 ReadPerson, WritePerson 및 SearchPerson 비즈니스 항목 서비스를 생성합니다. 읽기, 쓰기 및 검색 비즈니스 항목 서비스 단계를 통해 비즈니스 항목의 레코드를 읽고, 작성하고, 업데이트하고, 삭제하고, 검색할 수 있습니다.

비즈니스 항목 서비스

비즈니스 항목 서비스는 작업을 수행합니다. ReadBE, WriteBE 및 SearchBE 비즈니스 항목 서비스를 사용할 수 있습니다.

비즈니스 항목 서비스에는 서비스 단계가 있습니다. 들어오는 요청은 각 서비스 단계를 통과합니다. 한 단계의 출력은 다음 단계의 입력입니다. 한 단계의 출력은 다음 단계의 입력에 정보를 전달할 수 있습니다. 모든 비즈니스 항목 서비스 단계는 단일 트랜잭션에서 하나의 Enterprise Java Bean 호출로 실행됩니다. MDM Hub가 예외를 처리합니다.

참고: 비즈니스 항목 서비스를 사용하기 전에 연산 참조 저장소의 유효성을 검사하십시오.

ReadBE 비즈니스 항목 서비스

ReadBE 비즈니스 항목 서비스는 비즈니스 항목의 기본 개체 레코드에서 데이터를 읽습니다.

ReadBE 단계에서 페이지 번호 지정 매개 변수를 지정하여 반환할 레코드 수 및 보려는 결과 페이지를 설정할 수 있습니다.

ReadBE 서비스 결과에는 일시 삭제된 레코드가 포함되지 않습니다.

비즈니스 항목 서비스 요청에 EffectiveDate 매개 변수를 전달하지 않으면 MDM Hub가 유효 날짜를 NULL로 간주하고, 비즈니스 항목 서비스가 기본 개체에서 데이터를 읽습니다. EffectiveDate 매개 변수를 전달하면 MDM Hub가 교차 참조 레코드에서 BVT(최선의 진실, Best Version of the Truth)를 계산하고 읽기 비즈니스 항목 서비스가 최신 BVT(최선의 진실, Best Version of the Truth)를 반환합니다.

WriteBE 비즈니스 항목 서비스

WriteBE 비즈니스 항목 서비스는 비즈니스 항목 요소의 데이터를 업데이트하거나, 하위 비즈니스 항목 요소를 생성하거나, 하위 비즈니스 항목 요소를 삭제할 수 있습니다.

참고: WriteBE 비즈니스 항목 서비스는 기존의 트러스트 설정을 사용하여 기본 개체에 대한 트러스트를 계산합니다. 이 서비스로는 트러스트 재정의를 수행할 수 없습니다.

선택적 매개 변수

다음 테이블에는 WriteBE 비즈니스 항목 서비스에서 사용할 수 있는 선택적 매개 변수가 설명되어 있습니다.

매개 변수	설명
recordState	레코드 상태를 ACTIVE, PENDING 또는 DELETED로 설정합니다. **참고:** recordState=ACTIVE를 설정하고 소프트 삭제된 레코드에서 서비스를 실행하면 레코드가 활성 상태로 복원됩니다.
EffectivePeriod	유효 기간을 지정합니다. EffectivePeriod 매개 변수를 전달하지 않는 경우 MDM Hub는 바운드되지 않는 기간을 가정합니다. MDM Hub는 루트 개체와 하위 개체 간 유효 기간에 대한 맞춤이 있는지 확인하지 않습니다. 레코드를 작성하거나 업데이트하는 경우 상위 레코드 및 하위 레코드의 유효 기간이 맞는지 확인하십시오.

SearchBE 비즈니스 항목 서비스

SearchBE 비즈니스 항목 서비스는 비즈니스 항목에서 루트 레코드를 검색하는 데 사용합니다.

스마트 검색을 위해 비즈니스 항목을 구성하는 데 대한 자세한 내용은 *Multidomain MDM 구성 가이드*를 참조하십시오.

비즈니스 항목 서비스 끝점

EJB(Enterprise JavaBean) 끝점, REST(Representational State Transfer) 끝점 또는 SOAP(Simple Object Access Protocol) 끝점을 통해 비즈니스 항목 서비스에 액세스할 수 있습니다.

REST 끝점은 EJB 끝점을 기반으로 작성됩니다. REST 비즈니스 항목 서비스 구성은 REST URL이 EJB 비즈니스 항목 서비스 호출에 매핑되는 방식을 정의합니다.

비즈니스 항목 서비스에 대한 Enterprise JavaBean 끝점

EJB(Enterprise JavaBean) 끝점은 모든 유형의 비즈니스 항목 서비스 호출의 기본 끝점입니다. 다른 모든 끝점은 EJB 끝점에 매핑됩니다.

비즈니스 항목 서비스는 상태 비저장 EJB로 노출됩니다. 상태 비저장 EJB 컨테이너는 인스턴스를 풀링하고, 인스턴스를 할당하고, 로드 균형 조정 전략을 적용하여 도메인 내 여러 서버에서 로드를 분산시킬 수 있습니다.

EJB 끝점은 인증을 위해 사용자 이름과 암호를 수락합니다.

비즈니스 항목 서비스에 대한 REST 끝점

REST(Representational State Transfer Endpoint) 호출은 비즈니스 항목 서비스를 웹 서비스로 사용 가능하게 만듭니다.

WADL(Web Application Description Language) 파일에는 REST 웹 서비스의 XML 설명, 모든 REST URL 및 모든 REST 매개 변수가 포함되어 있습니다. MDM Hub는 각 연산 참조 저장소에 대한 WADL 파일을 생성합니다.

다음·위치에서·각·연산 참조 저장소에 대한 WADL 파일을 다운로드할 수 있습니다.

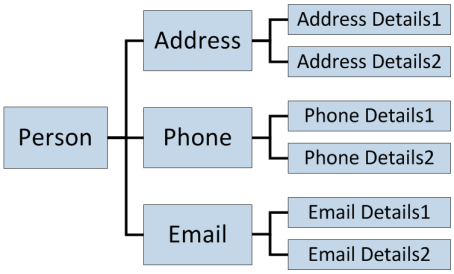
```
http://<호스트>:<포트>/cmx/csfiles
```

REST 및 EJB 비즈니스 항목 서비스 호출

비즈니스 항목 서비스 호출을 만드는 경우 전체 비즈니스 항목을 요청하는 대신 특정 하위 분기를 지정할 수 있습니다.

예를 들어 Person 루트 노드와 여러 하위 분기가 있는 비즈니스 항목에서 읽기 작업을 수행하려고 합니다. Person 기본 개체에는 Address, Phone 및 Email 하위 기본 개체가 있습니다. 각 하위 기본 개체에는 두 수준 아래 기본 개체가 두 개 있습니다.

다음 이미지는 여러 분기가 있는 비즈니스 항목의 구조를 보여 줍니다.



단일 요청을 통해 수준이 서로 다른 여러 하위 분기에서 데이터를 읽을 수 있습니다. 예를 들어 단일 요청의 Person, Phone, Phone Details1, Phone Details 2, Email 및 Email Details 2를 읽을 수 있습니다.

다음 URL 샘플은 행 ID가 1242인 Person 레코드는 물론 Address Details 1과 Email 하위 레코드도 가져오는 REST 읽기 요청을 만드는 방법을 보여 줍니다.

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

비즈니스 항목 서비스에 대한 SOAP 끝점

SOAP(Simple Object Access Protocol) 끝점 호출은 비즈니스 항목 서비스를 웹 서비스로 사용 가능하게 만듭니다.

WSDL(Web Services Description Language) 파일에는 웹 서비스의 XML 설명, SOAP 요청과 응답의 형식 및 모든 매개 변수가 포함되어 있습니다. MDM Hub는 각 연산 참조 저장소에 대한 WSDL 파일을 생성합니다.

루트 레코드 식별

다음 접근 방식 중 하나를 사용하여 루트 레코드를 식별할 수 있습니다.

- rowid. 레코드의 ROWID_OBJECT 열에 있는 값입니다.
- systemName 및 sourceKey. systemName은 레코드가 속해 있는 시스템의 이름입니다. sourceKey는 레코드의 PKEY_SRC_OBJECT 열에 있는 값입니다.
- 개체의 GBID(글로벌 비즈니스 식별자). GBID는 복합 값일 수 있으며, 이 경우 모든 값을 전달해야 합니다.

  참고: GBID 접근 방식은 ReadBE 서비스에서만 작동합니다.

다음 샘플 코드는 systemName 및 sourceKey를 사용하여 레코드를 식별합니다.

```
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);
```

보안 및 데이터 필터

기본 개체와 리소스에 사용자 역할 권한이 있는 경우 비즈니스 항목은 이러한 권한을 상속합니다. 비즈니스 항목 레코드에 액세스하려면 사용자 역할에 비즈니스 항목의 루트 기본 개체와 다른 리소스에 대한 적절한 권한이 있어야 합니다.

비즈니스 항목 서비스는 비즈니스 항목 필드에 설정된 데이터 필드도 상속합니다.

보안 및 데이터 필터에 대한 자세한 내용은 *Multidomain MDM 프로비저닝 도구 가이드*를 참조하십시오.

제 2 장

Enterprise Java Bean 비즈니스 항목 서비스 호출

이 장에 포함된 항목:

Enterprise Java Bean 비즈니스 항목 서비스 호출 개요

EJB(Enterprise Java Bean) 비즈니스 항목 서비스 호출을 만들어 비즈니스 항목의 기본 개체 레코드를 작성, 업데이트, 삭제 및 검색할 수 있습니다. EJB 비즈니스 항목 서비스 호출을 실행할 Java 코드를 작성할 수 있습니다.

표준 SDO(서비스 데이터 개체) 클래스를 기반으로 Java 코드를 작성하거나, MDM Hub가 비즈니스 항목 및 비즈니스 항목 서비스 구성에 따라 생성하는 java 클래스를 기반으로 Java 코드를 작성할 수 있습니다.

표준 SDO 클래스가 포함된 Java 코드 예제

이 예는 표준 SDO(서비스 데이터 개체) 클래스를 기반으로 EJB(Enterprise Java Bean) 호출을 실행하는 Java 코드를 보여 줍니다.

이 예는 리소스 키트의 다음 파일에 있습니다. C:\<MDM Hub 설치 디렉터리>\hub\resourcekit\samples\COS\source\java\com\informatica\mdm\sample\cs\DynamicSDO.java

다음 Java 코드는 표준 SDO 클래스를 기반으로 하며, Person 기본 개체 레코드를 작성하고, 여러 하위 레코드를 추가하고, 하나의 하위 레코드를 삭제한 다음 person 레코드와 그 모든 하위 레코드를 삭제하는 EJB 비즈니스 항목 서비스 호출을 실행합니다.

```
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();
        }

    }
```

생성된 SDO 클래스가 포함된 Java 코드 예

이 예에서는 MDM Hub가 비즈니스 항목 및 비즈니스 항목 서비스 구성에 따라 생성하는 Java 클래스를 기반으로 EJB(Enterprise Java Bean) 호출을 실행하는 Java 코드를 보여 줍니다.

이 예는 리소스 키트의 다음 파일에 있습니다. C:\<MDM Hub 설치 디렉터리>\hub\resourcekit\samples\COS\source\java\com\informatica\mdm\sample\cs\GeneratedSDO.java

다음 Java 코드는 생성된 클래스를 기반으로 하며, person 기본 개체 레코드를 작성하고, 여러 하위 레코드를 추가하고, 하나의 하위 레코드를 삭제한 다음 person 레코드와 그 모든 하위 레코드를 삭제하는 EJB 비즈니스 항목 서비스 호출을 실행합니다.

```
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();
    }
}
```

제 3 장

Representational State Transfer 비즈니스 항목 서비스 호출

이 장에 포함된 항목:

비즈니스 항목 서비스에 대한 REST API 개요

REST 끝점 호출은 모든 비즈니스 항목 서비스를 웹 서비스로 사용 가능하게 만듭니다.

비즈니스 항목의 기본 개체 레코드 및 관련 하위 레코드를 작성, 업데이트, 삭제 및 검색하는 REST 호출을 만들 수 있습니다. 레코드 병합, 병합 해제 및 일치 등의 작업을 수행할 수 있습니다. REST 호출을 수행해 태스크를 생성, 업데이트 및 검색하고 태스크를 수행할 수 있습니다. 태스크 또는 레코드의 첨부 파일 같은 파일을 생성, 업데이트 및 삭제하는 REST 호출을 수행할 수도 있습니다.

REST 비즈니스 항목 서비스 호출은 URL(Uniform Resource Locator) 형식의 웹 서비스 요청입니다. MDM Hub는 비즈니스 항목의 각 기본 개체에 대해 고유한 URL을 할당합니다. 고유한 URL을 사용하여 업데이트하거나 삭제할 기본 개체를 식별할 수 있습니다.

참고: REST API를 사용하여 비즈니스 항목 서비스를 호출하기 전에 연산 참조 저장소의 유효성을 검사하십시오.

지원되는 REST 메서드

비즈니스 항목 서비스에 대한 REST API는 표준 HTTP 메서드를 사용하여 레코드, 태스크 및 파일 같은 리소스에 대한 작업을 수행합니다.

비즈니스 항목 서비스에 대한 REST API는 다음 HTTP 요청 메서드를 지원합니다.

메서드	설명
GET	레코드, 태스크 또는 파일에 대한 정보를 검색합니다.
POST	태스크, 루트 레코드, 하위 레코드 또는 파일을 생성합니다. **참고:** POST 요청에서 ORS(연산 참조 저장소) 이름은 대/소문자를 구분합니다. ORS 이름의 대/소문자가 MDM Hub에서의 이름과 일치하지 않으면 오류가 발생합니다.
PUT	루트 레코드, 하위 레코드, 태스크 또는 파일을 업데이트합니다.
PATCH	부분적으로 태스크를 업데이트합니다.
DELETE	레코드, 하위 레코드 또는 파일을 삭제합니다.

인증 방법

비즈니스 항목 서비스에 대한 REST 끝점은 기본 HTTP 인증 방법을 사용하여 사용자를 인증합니다. 브라우저에서 비즈니스 항목 서비스에 처음으로 연결하는 경우 MDM Hub 사용자 이름과 암호를 제공해야 합니다. 인증이 성공하면 비즈니스 항목 서비스 REST API를 사용하여 작업을 수행할 수 있습니다.

브라우저는 사용자 자격 증명을 캐시하고 비즈니스 항목 서비스에 대한 모든 후속 요청에 해당 자격 증명을 사용합니다.

타사 응용 프로그램에서 로그인하기 위한 인증 쿠키

인증 쿠키를 사용하여 타사 응용 프로그램에서 MDM Hub 사용자를 인증하고 비즈니스 항목 서비스를 호출할 수 있습니다. 인증된 사용자의 자격 증명에 기반하여 쿠키를 가져올 수 있습니다. 쿠키를 저장하여 REST API를 호출하는 데 사용합니다. 사용자 이름과 암호는 하드 코딩하지 않아도 됩니다.

사용자 이름과 암호를 사용하여 Entity 360 View에 로그인하기 위해 다음과 같은 POST 요청을 만드십시오.

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

로그인 작업이 성공하면 서버가 set-cookie 헤더 필드에 인증 쿠키를 반환합니다. 다음 샘플 코드는 응답 헤더의 set-cookie를 보여 줍니다.

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

해시를 저장하여 API 호출의 요청 헤더에 사용합니다. API 호출 시 사용자 이름과 암호를 제공하지 않아도 됩니다.

다음 예에서는 API 요청 헤더에 인증 쿠키를 사용하는 방법을 보여 줍니다.

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

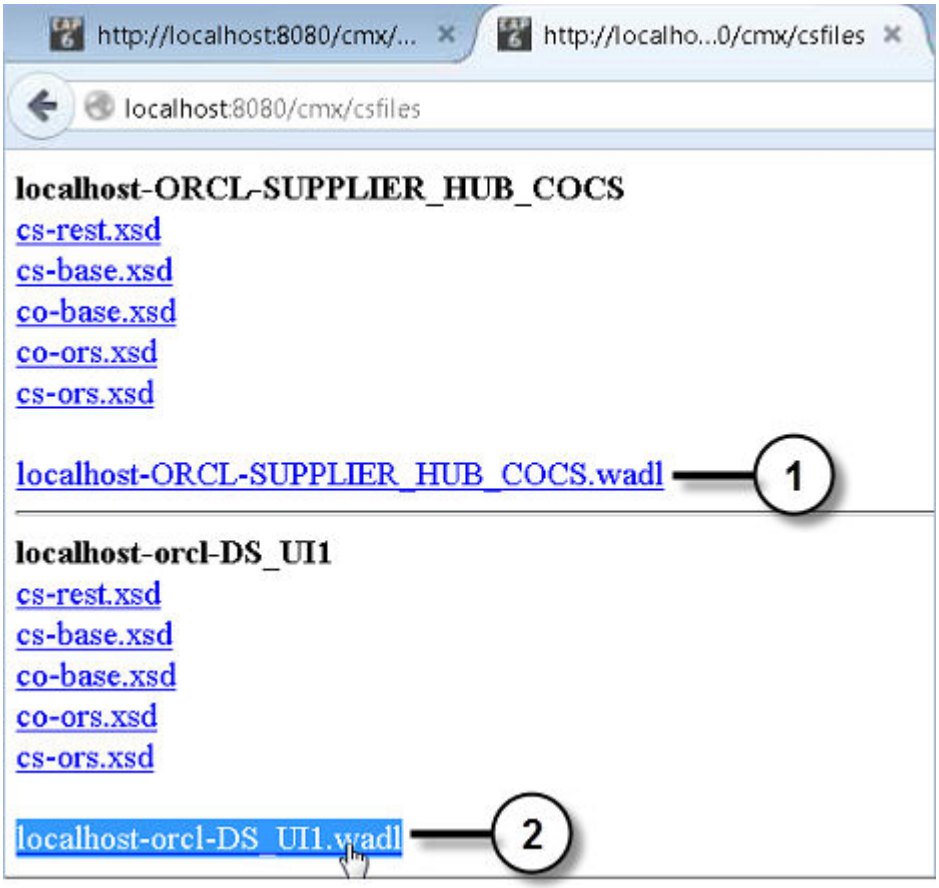
WADL(Web Application Description Language) 파일

WADL(Web Application Description Language) 파일에는 REST 웹 서비스의 XML 설명, 모든 REST URL 및 모든 REST 매개 변수가 포함되어 있습니다. MDM Hub는 각 연산 참조 저장소에 대한 WADL 파일을 생성합니다.

각 연산 참조 저장소에 대한 WADL 파일은 다음 위치에 있습니다.

```
http://<호스트>:<포트>/cmx/csfiles
```

다음 이미지는 연산 참조 저장소에 대한 WADL 파일을 다운로드할 수 있는 위치를 보여 줍니다.



1. SUPPLIER_HUB_COCS 연산 참조 저장소에 대한 WADL 파일을 다운로드하는 링크
2. DS_UI1 연산 참조 저장소에 대한 WADL 파일을 다운로드하는 링크

REST URL(Uniform Resource Locator)

REST URL을 사용하여 비즈니스 항목 서비스에 대한 REST 호출을 만들 수 있습니다.

REST URL에는 다음 구문이 있습니다.

```
http://<호스트>:<포트>/<컨텍스트>/<데이터베이스 ID>/<경로>
```

해당 URL에는 다음 필드가 있습니다.

호스트

데이터베이스가 실행되고 있는 호스트입니다.

포트

데이터베이스 수신기에서 사용하는 포트 번호입니다.

컨텍스트

비즈니스 항목 API 및 태스크 API에 대한 컨텍스트는 cmx/cs입니다.

파일 API에 대한 컨텍스트는 cmx/file입니다.

데이터베이스 ID

Hub 콘솔에서 데이터베이스 도구에 등록된 ORS의 ID입니다.

경로

API를 사용하려는 개체입니다(예 레코드, 태스크 또는 파일).

URL이 루트 레코드용인 경우 경로에는 루트 개체 이름 다음에 고유한 식별자가 나옵니다.

Person 루트 레코드의 경로에 대한 예는 Person/798243.json입니다.

URL이 루트 개체의 직접 하위인 레코드용인 경우 경로에는 하위 레코드 이름과 고유한 식별자도 포함됩니다.

Person 루트 레코드의 하위인 청구 주소 레코드의 경로에 대한 예는 다음과 같습니다.

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

URL이 둘 이상의 깊이에 있는 하위 레코드용인 경우 경로에는 깊이도 포함됩니다.

다음 URL은 깊이가 2인 하위 레코드의 REST URL에 대한 예입니다.

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

참고: 매개 변수는 대/소문자를 구분합니다. REST URL에 있는 매개 변수 이름의 대/소문자가 REST 구성에 있는 매개 변수 이름의 대/소문자와 일치하는지 확인하십시오.

헤더 및 본문 구성

REST 작업은 HTTP 메서드를 리소스에 대한 전체 URL과 결합합니다. 완전한 요청을 위해 REST 작업을 적절한 HTTP 헤더 및 모든 필요한 데이터와 결합하십시오. REST 요청에는 헤더 및 본문 구성 요소가 있습니다. JSON 또는 XML 형식을 사용하여 요청을 정의할 수 있습니다.

요청 헤더

요청 헤더를 사용하여 REST 작업의 메타데이터 또는 작동 매개 변수를 정의하십시오. 이 헤더는 일련의 필드-값 쌍으로 구성됩니다. 해당 API 요청 행에는 메서드 및 URL이 포함됩니다. 요청 행 다음에 헤더 필드를 지정하십시오.

REST API 요청 헤더를 구성하려면 다음 예에 나온 대로<METHOD> <<호스트>:<포트>/<컨텍스트>/<데이터베이스 ID>/<경로> 요청 행 다음에 헤더 필드를 추가하십시오.

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

다음 테이블에서는 일반적으로 사용되는 일부 요청 헤더 필드에 대해 설명합니다.

요청 구성 요소	설명
콘텐츠 유형	요청의 데이터에 대한 미디어 유형입니다. REST 요청에 본문을 포함시키는 경우 콘텐츠 유형 헤더 필드에서 본문의 미디어 유형을 지정해야 합니다. PUT 및 POST 요청에서 콘텐츠 유형 헤더 필드를 포함시키십시오.
허용	응답의 데이터에 대한 미디어 유형입니다. 요청 형식을 지정하려면 헤더의 application/<json/xml>을 사용하거나 .json 또는 .xml을 URL에 추가하십시오. 기본값은 XML입니다.

요청 본문

REST API 요청 본문을 사용하여 요청의 데이터를 보내십시오. 요청 본문은 POST 및 PUT 메서드 등 여기에 연결된 본문이 있을 수 있는 메서드와 함께 사용됩니다. 데이터 본문은 헤더 행 다음에 작성됩니다. 요청 메시지에 본문이 있는 경우 콘텐츠 유형 헤더 필드를 사용하여 요청 헤더의 본문 형식을 지정하십시오.

XSD(XML 스키마 정의) 파일은 사용자가 사용할 수 있는 요소 및 특성에 대해 설명합니다. 요청 본문의 콘텐츠는 사용자가 XSD 파일에서 정의하는 요소 유형에 따라 다릅니다.

XSD 파일은 다음 위치에 있습니다.

```
http://<호스트>:<포트>/cmx/csfiles
```

XML 형식

XML 요청 양식을 사용하는 경우 요청 개체를 태그의 구분 집합으로 정의하십시오.

다음 XML 형식을 사용하여 요청 개체를 정의하십시오.

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

다음 예는 요청 개체의 XML 표현을 보여 줍니다.

```
<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 형식

JSON 요청 형식을 사용하는 경우 유형 특성으로 요청 개체를 정의하십시오.

다음 JSON 형식을 사용하여 요청 개체를 지정하십시오.

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

다음 예는 요청 개체의 JSON 표현을 보여 줍니다.

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

표준 쿼리 매개 변수

비즈니스 항목 서비스 REST API는 표준 쿼리 매개 변수를 사용하여 결과 필터링, 페이지 지정 및 확장을 수행합니다.

물음표(?)를사용하여 쿼리 매개 변수를 다른 매개 변수와 구분하십시오. 쿼리 매개 변수는 등호 기호로 구분된 키-값 쌍입니다. 앰퍼샌드(&)를 사용하여 쿼리 매개 변수의 시퀀스를 구분하십시오.

다음 REST 요청 URL은 쿼리 매개 변수를 사용하는 방법을 보여 줍니다.

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

다음 표준 쿼리 매개 변수를 사용하십시오.

매개 변수	설명
recordsToReturn	반환할 행 수를 지정합니다. 기본값은 10입니다.
firstRecord	결과의 첫 번째 행을 지정합니다. 기본값은 1입니다. 추가 페이지를 읽기 위해 후속 호출에서 사용됩니다.
searchToken	이전 요청과 함께 반환되는 검색 토큰을 지정합니다. 검색 토큰을 사용하여 검색 결과의 후속 페이지를 가져올 수 있습니다. 예를 들어 다음 쿼리는 첫 번째 페이지를 나열합니다. /Person/123/Phone 다음 쿼리는 두 번째 페이지를 반환합니다. /Person/123/Phone?searchToken=SVR1.AZAM5&firstRecord=10
returnTotal	true로 설정된 경우 결과의 레코드 수가 반환됩니다. 기본값은 false입니다.
depth	결과에 포함시킬 하위 수준 수를 지정합니다.

날짜 및 시간 형식(UTC)

요청 및 응답에서 모든 날짜와 시간은 UTC(협정 세계시)로 지정되며, 특정 시간대에 대한 오프셋이 있거나 없을 수 있습니다.

요청 본문에 날짜 및 시간을 지정할 때는 ISO 사양 8601에 대해 Date and Time Formats (NOTE-datetime)에 정의된 형식 중 하나를 사용합니다.

다음은 NOTE-날짜/시간 문서에 나와 있는 지침입니다.

유형	구문	예
날짜: 연도	YYYY	1997
날짜: 연도 및 월	YYYY-MM	1997-07
날짜: 연도, 월 및 일	YYYY-MM-DD	1997-07-16
날짜와 시간 및 분	YYYY-MM-DDThh:mmTZD	1997-07-16T19:20+01:00

유형	구문	예
날짜와 시간, 분 및 초	YYYY-MM-DDThh:mm:ssTZD	1997-07-16T19:20:30+01:00
날짜와 시간, 분, 초 및 소수 초	YYYY-MM-DDThh:mm:ss.sTZD	1997-07-16T19:20:30.45+01:00

설명:

- YYYY = 네 자리 연도
- MM = 두 자리 월(01 - 12)
- DD = 두 자리 일(01 - 31)
- T = 날짜 다음에 시간을 소개하는 리터럴 값
- hh = 두 자리 시간(00 - 23)
- mm = 두 자리 분(00 - 59)
- ss = 두 자리 초(00 - 59)
- s = 초의 소수를 나타내는 하나 이상의 숫자
- TZD = 시간대 지정자(Z, +hh:mm 또는 -hh:mm)
  - Z = UTC 시간
  - +hh:mm = UTC보다 빠른 현지 시간대
  - -hh:mm = UTC보다 느린 현지 시간대

비즈니스 항목 서비스 REST 호출을 실행하도록 WebLogic 구성

비즈니스 항목 서비스 REST 호출은 기본 HTTP 인증을 사용하므로 REST 호출에 대한 WebLogic 서버 인증을 비활성화해야 합니다. 비즈니스 항목 서비스 REST 호출을 실행하도록 WebLogic을 구성하려면 WebLogic `config.xml` 파일을 편집하십시오.

1. 다음 WebLogic 디렉터리로 이동합니다.

   UNIX의 경우.

   ```
   <WebLogic 설치 디렉터리>/user_projects/domains/base_domain/config
   ```

   Windows의 경우.

   ```
   <WebLogic 설치 디렉터리>\user_projects\domains\base_domain\config
   ```

2. 텍스트 편집기에서 다음 파일을 엽니다.

   ```
   config.xml
   ```

3. </security-configuration> 태그를 닫기 전에 다음 XML 코드를 추가합니다.

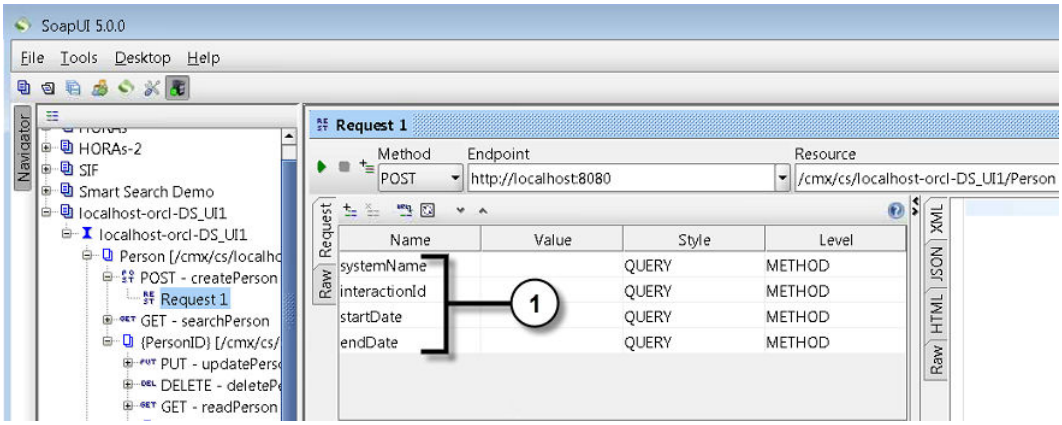
   ```
   <enforce-valid-basic-auth-credentials>
      false
   </enforce-valid-basic-auth-credentials>
   ```

입력 및 출력 매개 변수 보기

SoapUI와 같은 기능 테스트 도구를 사용하여 REST API 입력 및 출력 매개 변수를 볼 수 있습니다.

WADL 파일을 다운로드하고 이 파일을 기능 테스트 도구로 가져와서 REST 프로젝트를 작성하십시오.

다음 이미지는 createPerson REST API에 대한 SoapUI의 입력 매개 변수를 보여 줍니다.



1. createPerson REST API에 대한 입력 매개 변수

참고: WebSphere 환경에서 생성된 WADL 파일은 SoapUI에 제대로 가져오지 못할 수 있습니다. 입력 매개 변수가 SoapUI에 나타나지 않는 경우 WADL 파일을 편집하여 각 param 요소에서 xmlns 특성을 제거한 다음 WADL 파일을 다시 가져오십시오.

JavaScript 템플릿

다음 코드 샘플은 REST 비즈니스 항목 서비스 호출에 대한 JavaScript 코드를 작성하기 위해 수정할 수 있는 기본 템플릿을 보여 줍니다. jQuery java 스크립트 라이브러리가 필요합니다.

```
(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 예

리소스 키트에는 REST 비즈니스 항목 서비스 호출을 만드는 방법을 보여 주는 샘플 Java 소스 코드가 있습니다.

샘플 코드는 다음 파일에 있습니다.

<MDM Hub 설치 디렉터리>\hub\resourcekit\samples\COS\source\resources\webapp\rest-api.html

다음 코드는 person 루트 레코드를 작성하고, 여러 하위 레코드를 추가하고, 하나의 하위 레코드를 삭제한 다음 person 레코드와 그 모든 하위 레코드를 삭제하는 REST API 호출을 보여 줍니다.

```
<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>
```

비즈니스 항목 서비스에 대한 REST API 참조

비즈니스 항목 서비스에 대한 REST API 참조에서는 REST API를 나열하고 각 API에 대한 설명을 제공합니다. 또한 해당 API 참조에는 URL, 쿼리 매개 변수, 샘플 요청및 샘플 응답에 대한 정보가 포함되어 있습니다.

메타데이터 가져오기

메타데이터 가져오기 REST API는 비즈니스 항목 관계 또는 비즈니스 항목의 데이터 구조를 반환합니다.

이 API는 GET 메서드를 사용하여 다음과 같은 비즈니스 항목의 메타데이터를 반환합니다.

- 비즈니스 항목의 구조
- 필드 목록
- 데이터 유형 및 크기 등 필드 유형
- 생성, 업데이트 및 병합 같은 작업의 보안 설정
- 노드 또는 필드에 대한 지역화된 레이블
- 조회 필드에 대한 코드 및 표시 필드의 이름

API는 비즈니스 항목 관계에 대해 다음과 같은 세부 정보를 반환합니다.

- 관계의 이름 및 레이블.
- 원본 및 대상 비즈니스 항목.
- 관계 방향.

요청 URL

비즈니스 항목에 대한 메타데이터 가져오기 URL의 형식은 다음과 같습니다.

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

관계에 대한 메타데이터 가져오기 URL의 형식은 다음과 같습니다.

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

쿼리 매개 변수 "action=meta"를 사용하여 메타데이터 정보를 검색하십시오. 비즈니스 항목의 이름을 정확하게 지정해야 합니다.

메타데이터 가져오기 URL에 HTTP GET 요청을 만드십시오.

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

HTTP 헤더를 이 요청에 추가할 수 있습니다.

샘플 API 요청

다음 샘플 요청은 Person 비즈니스 항목 및 루트 노드에 대한 메타데이터 정보를 검색합니다.

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

다음 샘플 요청은 JSON 형식으로 Person 비즈니스 항목에 대한 메타데이터 정보를 검색하기 위한 헤더를 포함합니다.

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

다음 샘플 요청은 HouseholdContainsMemberPerson 관계에 대한 메타데이터 정보를 검색합니다.

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

샘플 API 응답

항목 및 관계에 대한 샘플 응답에는 보안 사용 권한이 포함되어 있습니다. 첫 번째 operations 섹션은 가능한 사용 권한을 정의합니다. object 섹션은 전체 비즈니스 항목 또는 관계에 대한 사용 권한을 나열합니다. fields 섹션은 필드 수준에서 사용 권한을 정의합니다.

다음 예제는 JSON 형식의 Person 비즈니스 항목에 대한 일부 데이터 구조를 보여 줍니다.

```
{
  "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
  }
}
```


메타데이터 나열

메타데이터 나열 REST API는 사용자가 정의한 비즈니스 항목 또는 관계의 목록을 반환합니다. 비즈니스 항목에 시간 표시 막대 정보 및 보안 정보가 포함되는 경우 응답에 이러한 정보가 포함됩니다. 이 API를 사용하면 항목에서 시작하거나, 항목에서 끝나거나, 지정된 항목에서 시작하고 끝나는 관계 목록을 검색할 수 있습니다.

해당 API는 GET 메서드를 사용합니다.

메타데이터 나열 URL

항목 메타데이터에 대한 메타데이터 나열 URL의 형식은 다음과 같습니다.

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

관계 메타데이터에 대한 메타데이터 나열 URL의 형식은 다음과 같습니다.

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

메타데이터 나열 URL에 대한 다음 HTTP GET 요청을 만드십시오.

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

쿼리 매개 변수

쿼리 매개 변수를 요청 URL에 추가하여 비즈니스 항목 관계의 검색 결과를 필터링할 수 있습니다. 방향을 지정하고 비즈니스 항목을 기준으로 관계를 검색할 수 있습니다.

다음 테이블에는 쿼리 매개 변수가 나와 있습니다.

매개 변수	설명
start	선택 사항입니다. 관계가 시작되는 항목을 지정합니다. 예를 들어 meta/relationship?start=Organization 쿼리는 Organization 비즈니스 항목에서 시작하는 모든 관계를 반환합니다.
finish	선택 사항입니다. 관계가 끝나는 항목을 지정합니다. 예를 들어 meta/relationship?finish=Person 쿼리는 Person 비즈니스 항목에서 끝나는 모든 관계를 반환합니다.

두 가지 매개 변수 모두 지정하여 비즈니스 항목 두 개 사이의 모든 관계를 검색할 수 있습니다. 예를 들어 meta/relationship?start=Organization&finish=Person 쿼리는 Organization 비즈니스 항목에서 시작하고 Person 비즈니스 항목에서 끝나는 모든 관계를 반환합니다.

샘플 API 요청

다음 샘플 요청은 구성된 비즈니스 항목의 목록을 검색합니다.

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

다음 샘플 요청은 구성된 관계의 목록을 검색합니다.

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

다음 샘플 요청은 Organization 비즈니스 항목에서 시작하여 Person 비즈니스 항목에서 끝나는 관계의 목록을 검색합니다.

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

샘플 API 응답

다음 예제는 구성된 관계 목록의 일부를 보여 줍니다.

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

레코드 읽기

레코드 읽기 REST API는 비즈니스 항목의 루트 레코드에 대한 세부 정보를 반환합니다. 해당 API를 사용하여 루트 레코드의 하위 레코드에 대한 세부 정보를 반환할 수 있습니다. 이 API를 사용하여 레코드의 콘텐츠 메타데이터를 볼 수 있습니다.

해당 API는 GET 메서드를 사용합니다.

결과 집합을 정렬하여 정보를 오름차순 또는 내림차순으로 볼 수 있습니다. 더 많은 수의 복합 매개 변수가 필요한 경우에는 POST 메서드를 사용하십시오. 예를 들어 필드 집합을 기준으로 하위 요소의 순서가 지정된 데이터를 검색할 수 있습니다.

요청 URL

행 ID 또는 소스 시스템 및 소스 키의 이름을 사용하여 요청 URL의 레코드를 지정하십시오.

레코드 읽기 URL의 형식은 다음과 같을 수 있습니다.

rowId가 포함된 URL

행 ID를 지정할 때 다음 URL 형식을 사용하십시오.

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

URL에 대한 다음 HTTP GET 요청을 만드십시오.

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

소스 시스템 이름 및 소스 키가 포함된 URL

소스 시스템 이름 및 소스 키를 지정할 때 다음 URL 형식을 사용하십시오.

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

시스템 이름 및 개체의 GBID(**글로벌 비즈니스 식별자**)**가 포함된** URL

소스 시스템 이름 및 GBID를 지정하는 경우에는 다음 URL 형식을 사용하십시오.

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

GBID**만 포함된** URL

GBID만 지정하는 경우에는 다음 URL 형식을 사용하십시오.

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

GBID**가 두 개 이상 포함된** URL

GBID를 두 개 이상 지정하는 경우에는 다음 URL 형식을 사용하십시오.

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

하위 노드에 대한 세부 정보를 반환하기 위한 URL

하위 노드에 대한 세부 정보를 반환하려면 다음 URL 형식을 사용하십시오.

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

지정된 하위 노드에 대한 세부 정보를 반환하기 위한 URL

지정된 하위 노드에 대한 세부 정보를 반환하려면 다음 URL 형식을 사용하십시오.

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

예를 들어 children= BillAddresses/Address,Email입니다.

특정 노드에 대한 세부 정보를 반환하기 위한 URL

특정 노드에 대한 세부 정보를 반환하려면 다음 URL 형식을 사용하십시오.

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

특정 노드의 하위 항목에 대한 세부 정보를 반환하기 위한 URL

특정 노드의 하위 항목에 대한 세부 정보를 반환하려면 다음 URL 형식을 사용하십시오.

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

레코드의 콘텐츠 메타데이터를 반환하기 위한 URL

레코드의 콘텐츠 메타데이터를 반환하려면 다음 URL 형식을 사용하십시오.

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

예를 들어 다음과 같이 GET 요청을 사용하여 하위 레코드에 대한 일치 항목을 검색할 수 있습니다.

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

필드를 기준으로 하위 요소를 정렬하기 위한 URL

필드를 기준으로 하위 요소를 정렬하려면 다음 URL 형식을 사용하십시오.

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

- 문자를 접미사로 사용하여 내림차순을 지정하십시오.

쿼리 매개 변수

요청 URL에 쿼리 매개 변수를 추가하여 레코드의 세부 정보를 필터링할 수 있습니다.

다음 테이블에는 쿼리 매개 변수가 나와 있습니다.

매개 변수	설명
depth	반환할 하위 수준 수입니다. 루트 노드 및 직접 하위 노드를 반환하려면 2를 지정하고 루트 노드, 직접 하위 노드 및 두 수준 하위 노드를 반환하려면 3을 지정하십시오. 기본값은 1입니다.
effectiveDate	데이터를 검색하려는 날짜입니다.
readSystemFields	결과에 시스템 필드를 반환할지 여부를 나타냅니다. 기본값은 false입니다.

매개 변수	설명
recordStates	레코드의 상태입니다. 상태의 쉼표로 구분된 목록을 제공하십시오. 지원되는 레코드 상태는 ACTIVE, PENDING 및 DELETED입니다. 기본값은 ACTIVE입니다.
contentMetadata	레코드의 메타데이터입니다. 쉼표로 구분된 목록을 제공하십시오. 예: XREF, PENDING_XREF, DELETED_XREF, HISTORY, XREF_HISTORY, 및 MATCH MATCH를 선택하면 _MTCH 테이블에서 검색한 일치된 레코드 목록이 응답에 포함됩니다.
historyDate	기록 데이터를 검색하려는 날짜입니다. _HIST 테이블에서 검색한 지정된 날짜의 레코드 데이터가 응답에 포함됩니다. historyDate를 contentMetadata 매개 변수와 함께 사용하여 기록 메타데이터를 검색할 수 있습니다. contentMetadata를 XREF, BVT 또는 TRUST로 설정합니다. - XREF. _HXRF 테이블의 기록 교차 참조 데이터가 응답에 포함됩니다. - BVT. _HCTL 테이블의 기록 BVT(최선의 진실, Best Version of the Truth)가 응답에 포함됩니다. - TRUST. _HCTL 및 _HVXR 테이블의 기록 트러스트 설정이 응답에 포함됩니다.
children	하위 노드 이름 또는 경로의 쉼표로 구분된 목록입니다.
suppressLinks	API 응답에 상위-하위 링크가 표시되는지 여부를 나타냅니다. 응답에서 모든 상위-하위 링크를 억제하려면 매개 변수를 true로 설정합니다. 기본값은 false입니다. 예를 들어 Person/1242?depth=10&suppressLinks=true 쿼리는 응답에 상위-하위 링크를 표시하지 않은 채로 레코드 세부 정보를 하위 수준 10개까지 표시합니다.
order	필드 이름의 쉼표로 구분된 목록입니다(선택적 접두사 + 또는 -). + 접두사는 결과를 오름차순으로 정렬함을 나타내고 - 접두사는 결과를 내림차순으로 정렬함을 나타냅니다. 기본값은 +입니다. 매개 변수를 두 개 이상 지정하면 결과 집합은 목록의 첫 번째 매개 변수부터 차례대로 순서가 지정됩니다. 예를 들어 Person/1242/Names?order=-name 쿼리는 이름을 내림차순으로 정렬하여 결과를 표시합니다. Person/1242/BillAddresses?order=rowidObject,-effStartDate 쿼리는 rowid를 오름차순으로 정렬한 다음 유효 시작 날짜를 내림차순으로 정렬하여 청구 주소를 표시합니다.

다음 예는 레코드의 세부 정보를 필터링하는 방법을 보여 줍니다.

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

관련 항목:

- "날짜 및 시간 형식(UTC)" 페이지 27

하위 요소의 정렬 순서를 지정하는 POST 요청

POST 요청을 사용하여 여러 필드를 기준으로 결과 집합의 순서를 지정할 수 있습니다. POST 본문에는 매개 변수나 필드를 포함합니다.

다음 샘플 요청에서는 읽기 작업에 POST 요청을 사용하고 여러 필드를 기준으로 데이터를 정렬하는 방법을 보여 줍니다.

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

참고: 비즈니스 항목의 각 수준에서 하위 항목 유형 각각에 대해서는 한 가지 유형의 정렬 순서만 허용됩니다.

정렬 순서 고려 사항

레코드 읽기 API는 각 비즈니스 항목 하위 노드에 대해 하나 이상의 필드를 사용한 정렬 순서 기능을 지원합니다. 다음 섹션에서는 정렬 순서를 지정할 때 알고 있어야 하는 특정 고려 사항을 설명합니다.

- 하위 요소 말고 두 수준 아래 하위 요소에 대해 정렬 순서를 지정하면 지정한 순서대로 두 수준 아래 하위 요소가 정렬됩니다. 이 경우 하위 요소는 두 수준 아래 하위 요소에 대해 지정된 정렬 순서에 따라 정렬되지 않습니다. 다음은 샘플 요청입니다.

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

  이 샘플 요청에서는 두 수준 아래 하위 요소인 PhonePermissions에 대해서는 정렬 순서가 내림차순으로 지정되었지만 하위 요소인 Phone에 대해서는 순서가 지정되지 않았습니다. Phone은 PhonePermissions 정렬 순서를 기준으로 정렬되지 않습니다.

- 두 수준 아래 하위 요소 말고 하위 요소에 대해 정렬 순서를 지정하면 지정된 정렬 순서로 하위 요소가 정렬됩니다. 이 경우 두 수준 아래 하위 요소는 하위 요소에 대해 지정된 정렬 순서에 따라 정렬되지 않습니다. 다음은 샘플 요청입니다.

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

  이 샘플 요청에서는 정렬 순서가 하위 요소 Phone에 대해서만 지정되고 두 수준 아래 하위 요소 PhonePermissions에 대해서는 지정되지 않았습니다. 이 경우 하위 요소 Phone은 지정된 순서대로 정렬됩니다.

- 하위 요소와 두 수준 아래 하위 요소에 대해 정렬 순서를 지정하면 두 요소 모두 정렬 순서에 따라 정렬됩니다. 다음 샘플 요청은 Phone(하위 요소) 및 PhonePermissions(두 수준 아래 하위 요소)에 대해 정렬 순서를 지정합니다.

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

```
    ]}
}}}
```

- 하위 요소는 하위 요소 자체의 열을 기준으로만 정렬할 수 있고, 두 수준 아래 하위 요소는 두 수준 아래 요소에 있는 열을 기준으로 정렬할 수 있습니다. 다음 샘플 요청에서는 PhoneType을 기준으로 Phone을 정렬하고 열 1을 기준으로 PhonePermissions를 정렬합니다. PhoneType은 Phone(하위 요소)의 열이고, 열 1은 PhonePermissions(두 수준 아래 하위 요소)의 열입니다.

  ```
  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
  ```

- 비즈니스 항목의 각 수준에서 하위 항목 유형 각각에 대해서는 한 가지 유형의 정렬 순서만 허용됩니다. 다음 요청에서는 상위 요소가 서로 다른 PhonePermissions 하위 요소에 대해 서로 다른 정렬 순서를 지정했습니다. 하지만 첫 번째 정렬 순서가 내림차순으로 지정되었기 때문에 양쪽 상위 요소(rowid 861 및 rowid 862)의 PhonePermissions 하위 요소가 내림차순으로 정렬됩니다.

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

샘플 API 요청

다음 샘플 요청은 Person 비즈니스 항목의 루트 레코드에 대한 세부 정보를 반환합니다.

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

다음 샘플 요청은 rowId 2가 포함된 하위 레코드의 세부 정보를 반환합니다. 하위 기본 개체는 genderCd이며 하위 레코드는 깊이 2에 있습니다.

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

다음 샘플 요청은 시스템 이름 및 소스 키를 사용하여 루트 레코드의 세부 정보를 반환합니다.

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

다음 샘플 요청은 루트 레코드 및 해당 XREF 레코드의 세부 정보를 반환합니다.

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

다음 샘플 요청은 이름을 내림차순으로 정렬하여 루트 레코드의 세부 정보를 반환합니다.

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

다음 샘플 요청은 rowid를 오름차순으로 정렬한 다음 유효 시작 날짜를 내림차순으로 정렬하여 청구 주소를 반환합니다.

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

샘플 API 응답

다음 예에서는 Person 비즈니스 항목의 루트 레코드에 대한 세부 정보를 보여 줍니다.

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

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


다음 예는 genderCd 기본 개체의 하위 레코드에 대한 세부 정보를 보여 줍니다.


```
    {
        "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"
}
```

레코드 생성

레코드 생성 REST API는 지정된 비즈니스 항목에 레코드를 생성합니다. 요청 본문의 레코드 데이터를 전송합니다. 승격 API를 사용하여 비즈니스 항목에 레코드를 승격시키고 추가합니다.

해당 API는 POST 메서드를 사용하여 레코드를 작성합니다.

요청 URL

레코드 생성 URL의 형식은 다음과 같습니다.

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

참고: 소스 시스템의 이름은 URL의 필수 매개 변수입니다.

레코드 생성 URL에 대한 다음 HTTP POST 요청을 만드십시오.

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

콘텐츠 유형 헤더를 추가하여 요청과 함께 보내려는 데이터의 미디어 유형을 지정하십시오.

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

URL 매개 변수

소스 시스템의 이름은 요청 URL의 필수 매개 변수입니다.

다음 테이블에는 URL에서 사용할 수 있는 매개 변수가 나와 있습니다.

매개 변수	설명
systemName	소스 시스템의 이름입니다.
interactionId	상호 작용의 ID입니다. 여러 요청을 단일 상호 작용에 그룹화할 수 있습니다. 모든 변경 내용이 상호 작용 ID로 수행됩니다.
startDate 및 endDate	레코드가 유효한 기간을 지정합니다. 시간 표시 막대가 사용되는 기본 개체에 대해 이러한 매개 변수를 제공하십시오.
validateOnly	쓰기 비즈니스 항목 서비스가 들어오는 데이터의 유효성을 검사하는지 여부를 나타냅니다. 기본값은 false입니다.
recordState	레코드의 상태입니다. 해당 매개 변수를 사용하여 레코드의 초기 상태를 지정하십시오. ACTIVE 또는 PENDING을 사용하십시오. 기본값은 ACTIVE입니다.
taskComment	API에 의해 트리거되는 워크플로우 태스크에 설명을 추가합니다.
taskAttachments	태스크 첨부 파일이 활성화된 경우 API에 트리거되는 워크플로우 태스크에 파일을 첨부합니다.

관련 항목:

- "날짜 및 시간 형식(UTC)" 페이지 27

요청 본문

레코드의 데이터를 REST 요청 본문에 보냅니다. JSON 형식 또는 XML 형식을 사용하여 데이터를 보내십시오. 메타데이터 가져오기 API를 사용하여 비즈니스 항목의 구조를 가져오고 요청 본문에 필수 매개 변수 값을 제공합니다.

응답 헤더 및 본문

응답이 성공한 경우 해당 API는 응답 헤더의 interactionId와 processId 및 응답 본문의 레코드 세부 정보를 반환합니다.

이 프로세스에서 상호 작용 ID를 생성한 후 이 ID를 사용하여 레코드를 생성하는 경우 해당 API는 상호 작용 ID를 반환합니다. 이 프로세스에서 데이터베이스에 레코드를 직접 저장하는 대신 워크플로우를 시작하는 경우 해당 API는 워크플로우 프로세스의 ID인 프로세스 ID를 반환합니다.

다음 예는 상호 작용 ID 및 프로세스 ID가 포함된 응답 헤더를 보여 줍니다.

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

응답 본문에는 레코드 및 생성된 행 ID가 포함됩니다.

샘플 API 요청

다음 샘플 요청은 Person 비즈니스 항목에 루트 레코드를 생성합니다. 이 요청은 API에 의해 트리거되는 워크플로우 태스크에 설명 및 첨부 파일을 추가합니다.

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

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

샘플 API 응답

다음 샘플 응답은 레코드를 성공적으로 생성한 후의 응답 헤더와 본문을 보여 줍니다.

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

레코드 업데이트

레코드 업데이트 REST API는 지정된 루트 레코드와 그 하위 레코드를 업데이트합니다. 요청 URL의 레코드 ID를 보내십시오. 변경 요약을 요청 본문에 보냅니다.

변경 후 레코드가 보류 중 상태인 경우, 승격 API를 사용하여 변경 내용을 승격합니다. 예를 들어 업데이트가 검토 워크플로우를 트리거하면 검토가 완료되기 전까지 레코드가 보류 중 상태입니다.

API는 POST 메서드를 사용합니다.

참고: 요청 본문에 변경 요약이 아니라 변경된 필드가 포함되는 간단한 PUT 버전을 사용할 수도 있습니다.

요청 URL

레코드 업데이트 URL의 형식은 다음과 같습니다.

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

참고: 소스 시스템의 이름은 URL의 필수 매개 변수입니다.

레코드 업데이트 URL에 대한 다음 HTTP PUT 요청을 만드십시오.

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

콘텐츠 유형 헤더를 추가하여 요청과 함께 보내려는 데이터의 미디어 유형을 지정하십시오.

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

쿼리 매개 변수

소스 시스템의 이름은 필수 쿼리 매개 변수입니다.

요청에 다음 쿼리 매개 변수를 사용할 수 있습니다.

매개 변수	설명
systemName	소스 시스템의 이름입니다.
interactionId	상호 작용의 ID입니다. 여러 요청을 단일 상호 작용에 그룹화할 수 있습니다. 모든 변경 내용이 상호 작용 ID로 수행됩니다.
startdate 및 enddate	레코드가 유효한 기간을 지정합니다. 시간 표시 막대가 사용되는 기본 개체에 대해 이러한 매개 변수를 제공하십시오.
validateOnly	쓰기 비즈니스 항목 서비스가 들어오는 데이터의 유효성을 검사하는지 여부를 나타냅니다. 기본값은 false입니다.

매개 변수	설명
recordState	레코드의 상태를 설정합니다. ACTIVE, PENDING 또는 DELETED를 사용하십시오. 예를 들어 recordState=ACTIVE를 설정하고 소프트 삭제된 레코드에서 요청을 실행하면 레코드가 활성 상태로 복원됩니다.
taskComment	API에 의해 트리거되는 워크플로우 태스크에 설명을 추가합니다.
taskAttachments	태스크 첨부 파일이 활성화된 경우 API에 트리거되는 워크플로우 태스크에 파일을 첨부합니다.

관련 항목:

- "날짜 및 시간 형식(UTC)" 페이지 27

요청 본문

REST 요청 본문에서 업데이트할 데이터를 보내십시오. JSON 형식 또는 XML 형식을 사용하여 데이터를 보내십시오.

새 매개 변수 값을 제공하십시오. $original 매개 변수를 사용하여 업데이트하려는 매개 변수의 이전 값을 나타내십시오.

하위 레코드에 다음 속성을 사용할 수도 있습니다.

속성/요소	유형	설명
MATCH	개체	하위 레코드에 대한 일치 테이블에서 일치 후보를 추가하거나 제거하려면 하위 레코드에 MATCH 개체를 추가합니다.
MERGE	개체	하위 레코드를 병합하거나, 병합에서 후보를 제거하려면 하위 레코드에 MERGE 개체를 추가합니다.

다음 JSON 코드 샘플은 루트 레코드에서 이름을 Bob으로 변경합니다.

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

다음 JSON 코드 샘플은 Address 하위 레코드의 일치 후보를 제거하고, Telephone 하위 레코드 두 개에 대해 병합을 정의합니다.

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

응답 헤더

응답이 성공한 경우 해당 API는 응답 헤더의 interactionId와 processId 및 응답 본문의 레코드 세부 정보를 반환합니다.

이 프로세스에서 상호 작용 ID를 생성한 후 이 ID를 사용하여 레코드를 생성하는 경우 해당 API는 상호 작용 ID를 반환합니다. 이 프로세스에서 데이터베이스에 레코드를 직접 저장하는 대신 워크플로우를 시작하는 경우 해당 API는 워크플로우 프로세스의 ID인 프로세스 ID를 반환합니다.

다음 예는 상호 작용 ID 및 프로세스 ID가 포함된 응답 헤더를 보여 줍니다.

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

응답 본문에는 레코드 및 생성된 rowId가 포함됩니다.

샘플 API 요청

다음 샘플 요청은 비즈니스 항목의 루트 레코드와 그 하위 레코드를 업데이트합니다. Person은 비즈니스 항목이고 Phone은 하위 기본 개체입니다. 이 요청은 API에 의해 트리거되는 워크플로우 태스크에 설명 및 첨부 파일을 추가합니다.

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

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

샘플 API 응답

다음 샘플 응답은 레코드를 성공적으로 업데이트한 후의 응답 헤더와 본문을 보여 줍니다.

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

레코드 삭제

레코드 삭제 REST API는 비즈니스 항목의 루트 레코드를 삭제합니다. 해당 API를 사용하여 루트 레코드의 하위 레코드를 삭제하십시오.

해당 API는 DELETE 메서드를 사용하여 레코드를 삭제합니다.

요청 URL

레코드 삭제 URL의 형식은 다음과 같습니다.

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

참고: 소스 시스템의 이름은 URL의 필수 매개 변수입니다.

레코드 삭제 URL에 대한 다음 HTTP DELETE 요청을 만드십시오.

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

다음 URL 형식을 사용하여 루트 레코드의 하위 레코드를 삭제하십시오.

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

쿼리 매개 변수

소스 시스템의 이름은 필수 URL 매개 변수입니다. systemName 매개 변수를 사용하여 소스 시스템을 지정하십시오.

샘플 API 요청

다음 샘플 요청은 Person 비즈니스 항목의 루트 레코드를 삭제합니다.

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

샘플 API 응답

다음 샘플 응답은 Person 비즈니스 항목에서 루트 레코드를 성공적으로 삭제한 후의 응답을 보여 줍니다.

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

레코드 나열

레코드 나열 REST API는 조회 값 또는 외래 키 값의 목록을 반환합니다. 조회는 참조 데이터를 제공하고 지정된 열에 대한 가능한 값 목록을 제공합니다.

해당 API는 GET 메서드를 사용합니다.

이 API는 조회 코드 값 및 조회 코드 설명을 가져오는 데도 사용할 수 있습니다. 조회의 정렬 순서를 지정할 수 있습니다. 더 많은 수의 복합 매개 변수가 필요한 경우에는 POST 메서드를 사용하십시오.

요청 URL

레코드 나열 REST URL의 형식은 다음과 같습니다.

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

URL에 대한 다음 HTTP GET 요청을 만드십시오.

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

조회 값의 코드를 나열하려면 다음 URL 형식을 사용하십시오.

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

참고: 메타데이터 가져오기 API를 사용하면 조회 값을 나열할 정확한 URL을 가져올 수 있습니다.

쿼리 매개 변수

쿼리 매개 변수를 요청 URL에 추가하여 결과를 필터링할 수 있습니다.

다음 테이블에는 쿼리 매개 변수가 나와 있습니다.

매개 변수	설명
suppressLinks	API 응답에 상위-하위 링크가 표시되는지 여부를 나타냅니다. 응답에서 모든 상위-하위 링크를 억제하려면 매개 변수를 true로 설정합니다. 기본값은 false입니다. 예를 들어 Person/1242?depth=10&suppressLinks=true 쿼리는 응답에 상위-하위 링크를 표시하지 않은 채로 레코드 세부 정보를 하위 수준 10개까지 표시합니다.
order	조회 값을 오름차순 또는 내림차순으로 나열하는 데 사용됩니다. + 문자를 접두사로 사용하여 오름차순을 지정하고 - 문자를 접두사로 사용하여 내림차순을 지정합니다. 기본적으로 접두사를 지정하지 않으면 결과 집합이 오름차순으로 순서가 지정됩니다. 예를 들어 LUNamePrefix?action=list&order=-namePrefixDisp 쿼리는 접두사의 표시 이름을 내림차순으로 정렬하여 이름의 접두사를 나열합니다.

정렬 순서를 지정하는 POST 요청

POST 요청을 사용하여 조회 값의 정렬 순서를 지정할 수 있습니다. POST 본문에는 매개 변수나 필드를 포함합니다.

다음 예제에서는 목록 작업에 POST 요청을 사용하는 방법을 보여 줍니다.

```
http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/ListCO.json
{
 parameters:
 {
   coFilter: {
   object: {
     name:"LUCountry",
     order: "-countryNameDisp"
   }
  }
 }
}
```

샘플 API 요청

다음 샘플 요청은 조회 값을 나열합니다.

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

다음 샘플 요청은 성별 조회 값에 대한 코드를 나열합니다.

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

다음 샘플 요청은 접두사의 표시 이름을 내림차순으로 정렬하여 이름의 접두사를 나열합니다.

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

샘플 API 응답

다음 샘플 응답은 조회 값 목록을 보여 줍니다.

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

다음 샘플 응답은 조회 값에 대한 코드를 보여 줍니다.

```
{
    "item": [
        {
            "id": "F",
            "label": "FEMALE"
        },
        {
            "id": "M",
            "label": "MALE"
        },
        {
            "id": "N",
            "label": "UNKNOWN"
        }
    ],
    "firstRecord": 1,
    "recordCount": 0,
    "pageSize": 2147483647,
    "searchToken": "SVR1.AU5LD"
}
```

검색 레코드

레코드 검색 REST API는 검색 가능한 루트 레코드와 그 모든 하위 레코드에서 인덱싱된 값을 검색합니다. 필터와 패싯을 사용하여 검색 결과의 하위 집합을 볼 수 있습니다. 패싯은 검색 결과를 범주로 그룹화하고, 필터는 검색 결과를 좁힙니다. 해당 API는 검색 가능으로 구성된 모든 필드를 검색하고 검색 조건과 일치하는 레코드를 반환합니다.

해당 API는 GET 메서드를 사용하여 검색 가능한 필드의 인덱스를 검색합니다.

요청 URL

레코드 검색 URL의 형식은 다음과 같습니다.

간단한 검색의 URL

간단한 검색의 경우 다음 URL을 사용하십시오.

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

레코드 검색 URL에 대한 다음 HTTP GET 요청을 만드십시오.

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

필드가 지정된 검색의 URL

필드가 지정된 검색의 경우 다음 URL을 사용하십시오.

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

비즈니스 항목 필드에 대해 음수 값(예: -120)을 지정하면 반환되는 레코드의 순위가 영향을 받을 수 있습니다.

패싯 검색의 URL

패싯이 포함된 간단한 검색의 경우 다음 URL을 사용하십시오.

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

패싯이 포함된 필드가 지정된 검색의 경우 다음 URL을 사용하십시오.

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

필터 검색의 URL

필터가 포함된 간단한 검색의 경우 다음 URL을 사용하십시오.

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

검색에 q 또는 fq 매개 변수를 사용하십시오.

URL **인코딩**

URL에는 공백 및 작은따옴표와 같은 문자가 포함되므로 URL 인코딩을 사용해야 합니다.

다음 예에서는 레코드 검색 URL의 URL 인코딩 표현을 보여 줍니다.

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

쿼리 매개 변수

q 또는 fq 쿼리 매개 변수를 사용하여 검색을 위한 문자열 값을 제공합니다. q 및 fq 쿼리 매개 변수는 함께 사용할 수 없습니다. 필드가 지정된 검색에는 fq 매개 변수를 사용하십시오. AND 논리 연산자를 사용하여 조건을 여러 개 지정할 수 있습니다.

다음 테이블에는 URL에서 사용할 수 있는 매개 변수가 나와 있습니다.

매개 변수	설명

q	문자열 값 또는 검색 용어를 지정합니다. 이 쿼리는 레코드의 모든 위치에서 검색 용어를 검색합니다. 간단한 검색에 사용됩니다.
	예를 들어 `Person?q=STEVE` 쿼리는 STEVE라는 용어가 있는 레코드를 검색합니다.
	두 개 이상의 용어를 함께 검색하려면 용어를 큰따옴표로 묶습니다. 검색 결과에 용어를 포함하려면 각 용어 앞에 + 문자를 사용합니다. 필드 값에 공백이 포함된 경우에는 필드 값을 작은따옴표로 묶습니다.
	WILLIAM JOHN LAWSON에 대한 정확한 일치 항목을 검색하려면 다음 쿼리를 사용하십시오.
	`Person?q="WILLIAM JOHN LAWSON"`
	WILLIAM, JOHN 또는 LAWSON을 검색하려면 다음 쿼리를 사용하십시오.
	`Person?q=WILLIAM JOHN LAWSON`
	WILLIAM, JOHN 및 LAWSON을 검색하려면 다음 쿼리를 사용하십시오.
	`Person?q=WILLIAM JOHN LAWSON&queryOperator=AND`

fq	특정 필드의 문자열 값 또는 검색 용어를 지정합니다. 이 쿼리는 레코드의 해당 부분에서만 용어를 검색합니다. 인덱싱된 필드를 기반으로 대상이 지정된 검색에 사용됩니다. 예를 들어 `Person?fq=displayname=STEVE` 쿼리는 표시 이름이 STEVE인 레코드를 검색합니다.
패싯	검색 결과를 그룹화하는 범주 또는 패싯으로 처리해야 하는 필드를 지정합니다. 검색 가능한 필드만 지정합니다. q 및 fq 매개 변수와 함께 사용됩니다. 구문은 `&facets=FieldName1,FieldName2,FieldNameN`입니다. 예를 들어 `Person?q=STEVE&facets=department` 쿼리는 표시 이름이 STEVE인 개인을 검색하고, 검색 결과를 부서별로 그룹화합니다. 이 검색은 표시 이름이 STEVE인 개인의 레코드를 표시하며, 이러한 레코드는 부서별로 그룹화됩니다.
필터	검색 결과를 좁힐 수 있는 필드를 지정합니다. 필터링 가능한 필드만 지정합니다. q 및 fq 매개 변수와 함께 사용됩니다. 예를 들어 `Person?fq=STEVE&filters=birthdate='1980-11-27T08:00:00Z'` 쿼리는 표시 이름이 STEVE인 개인을 검색하고, 생일을 기준으로 검색 결과를 필터링합니다. 이 검색은 표시 이름이 STEVE이고 생일이 1980년 11월 27일인 개인의 레코드를 표시합니다. **참고:** 날짜를 작은따옴표로 묶어서 지정합니다.
depth	반환할 하위 수준 수를 지정합니다. 루트 노드 및 직접 하위 노드를 반환하려면 2를 지정하고 루트 노드, 직접 하위 노드 및 두 수준 하위 노드를 반환하려면 3을 지정하십시오. 루트 노드만 반환하려면 1을 지정합니다. 기본적으로 깊이는 지정되지 않습니다. 깊이를 지정하지 않으면 검색 결과에는 검색 용어와 일치하는 항목이 발견된 루트 노드와 하위 노드가 반환됩니다. 예를 들어 `Person?q=STEVE&depth=2` 쿼리는 STEVE라는 용어가 포함된 레코드를 검색하고, 루트 레코드와 바로 아래의 하위 레코드에 대한 정보를 반환합니다.

매개 변수	설명
queryOperator	검색에서 검색 용어의 임의 문자열을 검색할지 검색 용어의 모든 문자열을 검색할지 여부를 지정합니다. 이 매개 변수는 다음 값 중 하나를 사용합니다. - OR. f 또는 fq 매개 변수에 나열된 임의 문자열을 검색합니다. - AND. f 또는 fq 매개 변수에 나열된 모든 문자열을 검색합니다. 이 매개 변수를 지정하지 않은 경우 기본값은 OR입니다. 예를 들어 Person?q=WILLIAM JOHN LAWSON&queryOperator=AND 쿼리는 WILLIAM, JOHN 및 LAWSON이 포함된 레코드를 검색합니다.
suppressLinks	API 응답에 상위-하위 링크가 표시되는지 여부를 나타냅니다. 응답에서 모든 상위-하위 링크를 억제하려면 매개 변수를 true로 설정합니다. 기본값은 false입니다. 예를 들어 Person?q=STEVE&suppressLinks=true 쿼리는 STEVE라는 용어가 포함된 레코드를 검색한 후 상위-하위 링크가 표시되지 않는 응답을 반환합니다.
readSystemFields	결과에 시스템 필드를 반환할지 여부를 나타냅니다. 기본값은 false입니다.
order	필드 이름의 쉼표로 구분된 목록입니다(선택적 접두사 + 또는 -). + 접두사는 결과를 오름차순으로 정렬함을 나타내고 - 접두사는 결과를 내림차순으로 정렬함을 나타냅니다. 기본값은 +입니다. 하위 필드를 사용하여 결과를 정렬하려면 필드의 전체 이름을 사용합니다. 예: BillAddresses.Address.cityName 매개 변수를 두 개 이상 지정하면 결과 집합은 목록의 첫 번째 매개 변수부터 차례대로 순서가 지정됩니다. 예를 들어 Person?order=displayName,-BillAddresses.Address.cityName 쿼리는 결과를 표시 이름을 기준으로 오름차순으로 정렬한 다음 도시 이름을 기준으로 내림차순으로 정렬합니다.
maxRecordsToSort	정렬할 검색 결과의 최대 수입니다. 기본값은 1000입니다.

filters 매개 변수와 함께 범위 지정:

filters 매개 변수를 사용하여 지정된 범위 내로 검색 결과를 좁힐 수 있습니다. 숫자 및 날짜 데이터 유형의 필터링 가능한 필드에 대해 범위를 지정할 수 있습니다.

정수 데이터 유형의 경우 다음 형식을 사용하십시오.

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

범위는 fromValue부터 toValue까지입니다. fromValue은 toValue보다 낮아야 합니다. 예를 들어 filters=age=[35,45] 쿼리는 검색 결과를 좁혀서 연령대가 35 -45인 레코드를 검색합니다.

날짜 데이터 유형의 경우 다음 형식을 사용하십시오.

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

범위는 fromDate부터 toDate까지입니다. 예를 들어 filters=birthdate=[2000-06-12T12:30:00Z, 2015-06-12T12:30:00Z] 쿼리는 생일을 2000년 6월 12일 - 2015년 6월 12일로 지정합니다.

참고: 정확히 일치하는 날짜 필터를 지정할 경우에는 해당 필터를 작은따옴표로 묶으십시오. 날짜 범위를 지정할 때는 따옴표를 사용하지 마십시오.

관련 항목:

- "날짜 및 시간 형식(UTC)" 페이지 27

샘플 API 요청

q 매개 변수가 포함된 요청

다음 샘플 요청은 이름이 STEVE인 레코드를 Person 비즈니스 항목에서 검색합니다.

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

fq 매개 변수가 포함된 요청

다음 샘플 요청은 표시 이름이 STEVE인 레코드를 Person 비즈니스 항목에서 검색합니다. displayName 필드는 인덱싱된 필드입니다.

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

정렬 옵션이 포함된 요청

다음 샘플 요청은 Person 비즈니스 항목에서 표시 이름이 STEVE인 레코드를 검색하여 도시를 기준으로 오름차순으로 결과를 정렬합니다.

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

fq 매개 변수 및 AND 논리 연산자가 포함된 요청

다음 샘플 요청은 표시 이름이 STEVE이고 납세자 ID가 DM106인 레코드를 Person 비즈니스 항목에서 검색합니다.

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

패싯이 포함된 요청

다음 샘플 요청은 표시 이름이 STEVE인 레코드를 Person 비즈니스 항목에서 검색한 다음 결과를 좁혀 부서별로 그룹화합니다.

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

필터(exact filter)가 포함된 요청

다음 샘플 요청은 표시 이름이 STEVE인 레코드를 Person 비즈니스 항목에서 검색한 후 지정된 도시와 국가를 기준으로 필터링합니다.

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

필터 범위가 포함된 요청

다음 샘플 요청은 표시 이름이 STEVE인 레코드를 Person 비즈니스 항목에서 검색한 후 35 - 45의 연령대로 필터링합니다.

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

샘플 API 응답

다음 샘플 응답은 이름 STEVE로 검색한 결과를 보여 줍니다.

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


제안기

제안기 REST API는 현재 사용자의 데이터베이스에 있는 데이터에 기반하여 검색 문자열에 대해 관련 용어 목록을 반환합니다. 이 API를 사용하여 사용자 인터페이스 필드에 입력하는 문자를 허용하고, 입력한 내용을 자동으로 완성하는 제안을 반환할 수 있습니다. 제안 목록에서 문자열을 찾아 선택할 수 있습니다. 제안기 API는 검색 가능한 필드에 사용합니다.

해당 API는 GET 메서드를 사용합니다.

참고: API를 사용하여 검색 가능한 필드에 대해 자동 완성 제안 목록을 제공하려면 필드의 suggester 속성을 true로 설정하고 데이터를 다시 인덱싱하십시오.

요청 URL

제안기 REST URL의 형식은 다음과 같습니다.

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

URL에 대한 다음 HTTP GET 요청을 만드십시오.

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

참고: Solr 검색 엔진을 사용하는 경우 처리 서버가 다시 시작될 때마다 제안기 인덱스를 다시 작성해야 합니다. 처리 서버가 다시 시작될 때마다 인덱스를 다시 작성하려면 buildIndex 매개 변수를 true로 설정합니다.

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

쿼리 매개 변수

다음 테이블에는 쿼리 매개 변수가 나와 있습니다.

매개 변수	설명
suggest	필수 사항입니다. 제안이 필요한 문자열을 지정합니다.
recordsToReturn	반환할 행 수를 지정합니다.
buildIndex	선택 사항입니다. 인덱스를 작성할지 여부를 나타냅니다. 컬렉션에 대한 인덱스를 명시적으로 작성하려면 시스템을 다시 시작할 때 이 매개 변수를 true로 설정하십시오. 이 매개 변수는 향후 릴리스에서 더 이상 사용되지 않을 수 있습니다.

샘플 API 요청

다음 샘플 요청은 사용자 인터페이스에서 사용할 수 있는 제안 목록을 반환합니다.

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

샘플 API 응답

다음 샘플 응답은 제안 목록을 보여 줍니다.

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

}
```

BPM 메타데이터 가져오기

BPM 메타데이터 가져오기 REST API는 BPM 워크플로우 도구가 태스크 연계 가져오기 서비스 및 관리 서비스를 수행할 수 있는지 여부를 지정하는 두 개의 표시기 및 태스크 유형을 반환합니다.

해당 API는 GET 메서드를 사용합니다.

요청 URL

BPM 메타데이터 가져오기 URL의 형식은 다음과 같습니다.

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

BPM 메타데이터 가져오기 URL에 대한 다음 HTTP GET 요청을 만드십시오.

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

샘플 API 요청

다음 샘플 요청은 태스크 유형 및 BPM 워크플로우 도구에 대한 정보를 반환합니다.

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

샘플 API 응답

다음 샘플 응답은 태스크 유형 및 BPM 워크플로우 도구에 대한 두 표시기의 값을 보여 줍니다.


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


태스크 나열

태스크 나열 API는 워크플로우 태스크 목록을 반환합니다. 워크플로우는 비즈니스 프로세스의 활동과 해당 활동을 통한 실행 경로를 정의합니다. 각 활동은 태스크라고 합니다.

해당 API는 GET 메서드를 사용하여 태스크의 정렬되고 페이지가 지정된 목록을 반환합니다.

요청 URL

태스크 나열 URL의 형식은 다음과 같습니다.

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

태스크 나열 URL에 대한 다음 HTTP GET 요청을 만드십시오.

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

HTTP 헤더를 이 요청에 추가할 수 있습니다.

쿼리 매개 변수

태스크 데이터 필드를 쿼리 매개 변수로 사용하여 태스크 목록을 필터링하십시오.

다음 쿼리 매개 변수를 사용할 수 있습니다.

매개 변수	설명
taskType	레코드에 대해 수행할 수 있는 작업 집합입니다. 이름 특성을 사용하여 태스크 유형을 지정하십시오. 태스크 유형에 대한 자세한 내용은 *Multidomain MDM Data Director 구현 가이드*를 참조하십시오.
taskId	태스크의 ID입니다.
processId	태스크가 포함된 워크플로우 프로세스의 ID입니다.
소유자	태스크를 수행하는 사용자입니다.
title	태스크에 대한 간단한 설명입니다.
상태	워크플로우의 태스크 상태입니다. 다음 두 값 중 하나를 사용하십시오. - Open: 태스크가 시작되지 않았거나 진행 중입니다. - Closed: 태스크가 완료되거나 취소되었습니다.
우선 순위	태스크의 중요도 수준입니다. high, normal 및 low 중 하나의 값을 사용합니다.
creator	태스크를 생성하는 사용자입니다.
createDateBefore 및 createDateAfter	날짜 범위입니다. createDate 필드를 기준으로 태스크를 필터링할 수 있습니다.
dueDateBefore 및 dueDateAfter	날짜 범위입니다. dueDate 필드를 기준으로 태스크를 필터링할 수 있습니다.

쿼리 매개 변수를 요청 URL의 이름-값 쌍으로 사용하십시오.

다음 예는 쿼리 매개 변수를 사용하여 태스크를 필터링하는 방법을 보여 줍니다.

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

관련 항목:

- "날짜 및 시간 형식(UTC)" 페이지 27

정렬 매개 변수

REST API 응답에서 정렬 순서를 결정하려면 일반 정렬 매개 변수를 사용하고 태스크 필드의 쉼표로 구분된 목록을 제공하십시오. 각 필드에 대한 정렬 순서를 지정할 수 있습니다. 대시 기호(-)를 사용하여 내림차순을 지정하십시오. 기본 정렬 순서는 오름차순입니다.

다음 예는 결과를 정렬하는 방법을 보여 줍니다.

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

샘플 API 요청

다음 샘플 요청은 태스크 목록을 검색합니다.

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

이 요청에는 본문이 포함되지 않습니다.

샘플 API 응답

다음 샘플 응답은 JSON 형식의 태스크 목록을 보여 줍니다.


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


태스크 읽기

태스크 읽기 REST API는 태스크 유형, 우선 순위 및 상태 등 태스크의 세부 정보를 반환합니다.

해당 API는 GET 메서드를 사용합니다.

요청 URL

태스크 읽기 URL의 형식은 다음과 같습니다.

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

참고: 태스크 나열 API를 사용하여 태스크의 ID를 가져오십시오.

태스크 읽기 URL에 대한 다음 HTTP GET 요청을 만드십시오.

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

샘플 API 요청

다음 샘플 요청은 태스크의 세부 정보를 반환합니다.

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

샘플 API 응답

다음 샘플 응답은 태스크의 세부 정보를 보여 줍니다.


```
{
    "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"
    }
```


태스크 작성

태스크 작성 REST API는 태스크를 작성하고 워크플로우를 시작합니다.

해당 API는 POST 메서드를 사용하여 태스크를 작성하고 해당 태스크가 포함된 워크플로우 프로세스의 ID를 반환합니다.

요청 URL

태스크 작성 URL의 형식은 다음과 같습니다.

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

태스크 작성 URL에 대한 다음 HTTP POST 요청을 만드십시오.

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

콘텐츠 유형 헤더를 추가하여 요청과 함께 보내려는 데이터의 미디어 유형을 지정하십시오.

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

요청 본문

태스크를 생성할 때 태스크 특성을 지정하십시오. JSON 형식 또는 XML 형식을 사용하여 요청의 태스크 데이터를 보내십시오.

다음 테이블에서는 요청 본문의 태스크 매개 변수에 대해 설명합니다.

매개 변수	설명
taskType	레코드에 대해 수행할 수 있는 작업 집합입니다. 이름 특성을 사용하여 태스크 유형을 지정하십시오. 태스크 유형에 대한 자세한 내용은 *Multidomain MDM Data Director 구현 가이드*를 참조하십시오.
소유자	생성자가 태스크를 할당하는 사용자입니다.
title	태스크에 대한 간단한 설명입니다.
comments	태스크에 대한 설명입니다.
첨부 파일	태스크의 첨부 파일입니다.
dueDate	소유자가 태스크를 완료해야 하는 날짜입니다.
상태	워크플로우의 태스크 상태입니다. 다음 두 값 중 하나를 사용하십시오. - Open: 태스크가 시작되지 않았거나 진행 중입니다. - Closed: 태스크가 완료되거나 취소되었습니다.
우선 순위	태스크의 중요도 수준입니다. high, normal 및 low 중 하나의 값을 사용합니다. 기본값은 normal입니다.
creator	태스크를 생성하는 사용자입니다.
createDate	태스크를 생성하는 날짜입니다.
orsId	Hub 콘솔에서 데이터베이스 도구에 등록된 ORS(연산 참조 저장소)의 ID입니다.
processId	ActiveVOS® 태스크 유형 ID입니다. 자세한 내용은 *Multidomain MDM Data Director 구현 가이드*를 참조하십시오.
taskRecord	태스크와 연결된 교차 참조 레코드 또는 비즈니스 개체 루트 레코드입니다. 행 ID 또는 소스 시스템과 소스 키를 사용하여 레코드를 지정하십시오.
businessEntity	taskRecord가 속해 있는 비즈니스 항목의 이름입니다.
interactionId	상호 작용의 ID입니다. 태스크와 레코드 사이에 태스크 컨텍스트 관계를 유지하는 데 상호 작용 ID를 사용합니다.
groups	지정된 사용자 그룹의 모든 사용자에게 태스크를 할당합니다. 사용자 그룹은 MDM Hub 콘솔에서 정의합니다. 그룹을 배열로 지정합니다.

다음 샘플 코드는 rowId를 사용하여 taskRecord를 지정합니다.

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

요청 본문의 형식은 다음과 같습니다.

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

관련 항목:

- "날짜 및 시간 형식(UTC)" 페이지 27

샘플 API 요청

다음 샘플 요청은 루트 레코드에 대한 태스크를 생성합니다.

```
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"
         }
       }
```

```
        }]
    }
```

샘플 API 응답

다음 예는 성공적으로 태스크 작성 시 응답을 보여 줍니다. 해당 API는 태스크가 포함된 워크플로우 프로세스의 ID를 반환합니다.

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

태스크 업데이트

태스크 업데이트 REST API는 단일 태스크를 업데이트합니다.

해당 API는 PATCH 메서드를 사용하여 일부 태스크 필드를 업데이트하고 PUT 메서드를 사용하여 전체 태스크를 업데이트합니다. 태스크 ID를 URL 매개 변수로 제공하십시오.

요청 URL

태스크 업데이트 URL의 형식은 다음과 같습니다.

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

참고: 태스크 나열 API를 사용하여 태스크의 ID를 가져오십시오.

태스크 업데이트 URL에 대한 다음 HTTP PUT 요청을 만들어 전체 태스크를 업데이트하십시오.

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

태스크 업데이트 URL에 대한 다음 HTTP PATCH 요청을 만들어 일부 태스크 필드를 업데이트하십시오.

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

콘텐츠 유형 헤더를 추가하여 요청과 함께 보내려는 데이터의 미디어 유형을 지정하십시오.

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

요청 본문

태스크 읽기 API를 사용하여 태스크의 세부 정보를 가져오십시오. 태스크를 업데이트할 때 태스크 특성을 지정하십시오. JSON 형식 또는 XML 형식을 사용하여 요청에서 업데이트할 데이터를 보내십시오.

다음 테이블에서는 요청 본문의 태스크 매개 변수에 대해 설명합니다.

매개 변수	설명
taskType	레코드에 대해 수행할 수 있는 작업 집합입니다. 이름 특성을 사용하여 태스크 유형을 지정하십시오. 태스크 유형에 대한 자세한 내용은 *Multidomain MDM Data Director 구현 가이드*를 참조하십시오.
taskId	태스크의 ID입니다.
소유자	태스크를 수행하는 사용자입니다.
title	태스크에 대한 간단한 설명입니다.

매개 변수	설명
comments	태스크에 대한 설명입니다.
첨부 파일	태스크의 첨부 파일입니다.
dueDate	소유자가 태스크를 완료해야 하는 날짜입니다.
상태	워크플로우의 태스크 상태입니다. 다음 두 값 중 하나를 사용하십시오. - Open: 태스크가 시작되지 않았거나 진행 중입니다. - Closed: 태스크가 완료되거나 취소되었습니다.
우선 순위	태스크의 중요도 수준입니다. high, normal 및 low 중 하나의 값을 사용합니다. 기본값은 normal입니다.
creator	태스크를 생성하는 사용자입니다.
createDate	태스크가 생성된 날짜입니다.
updatedBy	태스크를 업데이트하는 사용자입니다.
lastUpdateDate	태스크가 마지막으로 업데이트된 날짜입니다.
orsId	Hub 콘솔에서 데이터베이스 도구에 등록된 ORS의 ID입니다.
processId	태스크가 포함된 워크플로우 프로세스의 ID입니다.
taskRecord	태스크에 연결된 교차 참조 레코드 또는 루트 레코드입니다. 행 ID 또는 소스 시스템과 소스 키를 사용하여 레코드를 지정하십시오.
businessEntity name	taskRecord가 속해 있는 비즈니스 항목의 이름입니다.

다음 샘플 코드는 rowId를 사용하여 taskRecord를 지정합니다.

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

PATCH 요청의 경우 요청 본문에는 사용자가 변경하려는 해당 태스크 필드가 포함됩니다. 태스크 제목, 우선 순위, 기한 및 소유자를 변경할 수 있습니다.

PUT 요청의 경우 요청 본문에는 모든 태스크 필드가 포함됩니다. PUT 요청에 대한 다음 요청 본문을 사용하십시오.

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

관련 항목:

- "날짜 및 시간 형식(UTC)" 페이지 27

샘플 API 요청

다음 샘플 PUT 요청은 전체 태스크를 업데이트합니다.

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

다음 샘플 PATCH 요청은 일부 태스크 필드를 업데이트합니다.

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

샘플 API 응답

해당 API는 성공적으로 태스크 업데이트 시 200 OK 응답 코드를 반환합니다. 응답 본문은 비어 있습니다.

태스크 완료

태스크 완료 REST API는 사용자가 워크플로우의 모든 태스크를 완료한 후 태스크 워크플로우를 닫습니다. 모든 태스크 관련 레코드를 처리한 후 이 API를 사용하여 워크플로우를 닫습니다. 예를 들어 병합 후보를 선택하는 경우 병합 워크플로우를 시작하는 태스크를 작성할 수 있습니다. 병합 태스크는 사용자가 각 후보를 미리 보고 이 후보를 병합하거나 일치 항목 아님으로 표시한 후 완료됩니다. 해당 API를 사용하여 병합 워크플로우를 닫으십시오.

해당 API는 PUT 메서드를 사용합니다.

요청 URL

태스크 완료 URL의 형식은 다음과 같습니다.

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

태스크 완료 URL에 대한 다음 HTTP PUT 요청을 만드십시오.

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

콘텐츠 유형 헤더를 추가하여 요청과 함께 보내려는 데이터의 미디어 유형을 지정하십시오.

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

요청 본문

요청 본문의 태스크 세부 정보를 보내십시오. 태스크 읽기 API를 사용하여 태스크의 세부 정보를 가져오십시오.

다음 테이블에서는 요청 본문의 태스크 매개 변수에 대해 설명합니다.

매개 변수	설명
taskType	레코드에 대해 수행할 수 있는 작업 집합입니다. 이름 특성을 사용하여 태스크 유형을 지정하십시오. 태스크 유형에 대한 자세한 내용은 *Multidomain MDM Data Director 구현 가이드*를 참조하십시오.
taskId	태스크의 ID입니다.
소유자	태스크를 수행하는 사용자입니다.
title	태스크에 대한 간단한 설명입니다.
comments	태스크에 대한 설명입니다.

매개 변수	설명
dueDate	소유자가 태스크를 완료해야 하는 날짜입니다.
상태	워크플로우의 태스크 상태입니다. 다음 두 값 중 하나를 사용하십시오. - Open: 태스크가 시작되지 않았거나 진행 중입니다. - Closed: 태스크가 완료되거나 취소되었습니다.
우선 순위	태스크의 중요도 수준입니다.
creator	태스크를 생성하는 사용자입니다.
createDate	태스크가 생성된 날짜입니다.
updatedBy	태스크를 업데이트하는 사용자입니다.
lastUpdateDate	태스크가 마지막으로 업데이트된 날짜입니다.
orsId	Hub 콘솔에서 데이터베이스 도구에 등록된 ORS의 ID입니다.
processId	태스크가 포함된 워크플로우 프로세스의 ID입니다.
taskRecord	태스크에 연결된 교차 참조 레코드 또는 루트 레코드입니다. 행 ID 또는 소스 시스템과 소스 키를 사용하여 레코드를 지정하십시오.
businessEntity name	taskRecord가 속해 있는 비즈니스 항목의 이름입니다.

다음 샘플 코드는 rowId를 사용하여 taskRecord를 지정합니다.

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

관련 항목:

- "날짜 및 시간 형식(UTC)" 페이지 27

샘플 API 요청

다음 샘플 요청은 병합 워크플로우를 완료합니다.

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

샘플 API 응답

해당 API는 성공적으로 태스크 워크플로우 완료 시 200 OK 응답을 반환합니다. 응답 본문은 비어 있습니다.

태스크 작업 실행

태스크 작업 실행 REST API는 향후 처리를 위해 워크플로우로 태스크를 다시 설정합니다. 각 태스크 유형에는 태스크 작업 집합 및 태스크의 시퀀스를 지정하는 워크플로우가 있습니다. 태스크 작업을 실행하면 이 태스크가 워크플로우의 다음 단계로 이동됩니다. 태스크 작업에 후속 태스크가 없는 경우 태스크 작업을 실행하면 워크플로우가 종료됩니다.

해당 API는 POST 메서드를 사용하여 태스크 승인, 에스컬레이션 또는 취소 등의 작업을 수행합니다.

요청 URL

다음 URL은 태스크 작업 실행 URL의 형식을 지정합니다.

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

참고: 태스크 나열 API를 사용하여 태스크의 ID를 가져오십시오.

태스크 작업 실행 URL에 대한 다음 HTTP POST 요청을 만드십시오.

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

태스크 작업을 실행하기 전에 태스크를 편집하려는 경우 콘텐츠 유형 헤더를 추가하여 요청 데이터의 미디어 유형을 지정하십시오.

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

요청 본문

태스크 작업을 실행하기 전에 태스크 세부 정보를 변경하려는 경우 요청 본문에 태스크 데이터를 제공하십시오.

다음 테이블에서는 요청 본문의 매개 변수에 대해 설명합니다.

매개 변수	설명
taskType	레코드에 대해 수행할 수 있는 작업 집합입니다. 이름 특성을 사용하여 태스크 유형을 지정하십시오. 태스크 유형에 대한 자세한 내용은 *Multidomain MDM Data Director 구현 가이드*를 참조하십시오.
taskId	태스크의 ID입니다.
소유자	태스크를 수행하는 사용자입니다.

매개 변수	설명
title	태스크에 대한 간단한 설명입니다.
comments	태스크에 대한 설명입니다.
attachments	태스크의 첨부 파일입니다.
dueDate	소유자가 태스크를 완료해야 하는 날짜입니다.
상태	워크플로우의 태스크 상태입니다. 다음 두 값 중 하나를 사용하십시오. - Open: 태스크가 시작되지 않았거나 진행 중입니다. - Closed: 태스크가 완료되거나 취소되었습니다.
우선 순위	태스크의 중요도 수준입니다. high, normal 및 low 중 하나의 값을 사용합니다.
creator	태스크를 생성하는 사용자입니다.
createDate	태스크가 생성된 날짜입니다.
updatedBy	태스크를 업데이트하는 사용자입니다.
lastUpdateDate	태스크가 마지막으로 업데이트된 날짜입니다.
businessEntity	비즈니스 항목의 이름입니다.
orsId	Hub 콘솔에서 데이터베이스 도구에 등록된 ORS의 ID입니다.
processId	태스크가 포함된 워크플로우 프로세스의 ID입니다.
taskRecord	태스크에 연결된 교차 참조 레코드 또는 루트 레코드입니다. 행 ID 또는 소스 시스템과 소스 키를 사용하여 레코드를 지정하십시오.
businessEntity name	taskRecord가 속해 있는 비즈니스 항목의 이름입니다.

다음 샘플 코드는 rowId를 사용하여 taskRecord를 지정합니다.

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

관련 항목:

- "날짜 및 시간 형식(UTC)" 페이지 27

샘플 API 요청

다음 샘플 요청은 태스크를 취소하고 워크플로우를 종료합니다.

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

샘플 API 응답

해당 API는 성공적으로 태스크 작업 실행 시 200 OK 응답 코드를 반환합니다. 응답 본문은 비어 있습니다.

할당 가능한 사용자 나열

할당 가능한 사용자 나열 REST API는 태스크 유형에 속하는 태스크를 할당할 수 있는 사용자 목록을 반환합니다. 해당 API를 사용하여 태스크에 대한 대상 사용자를 가져오십시오.

해당 API는 GET 메서드를 사용합니다.

요청 URL

할당 가능한 사용자 나열 URL의 형식은 다음과 같습니다.

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

할당 가능한 사용자 나열 URL에 대한 다음 HTTP GET 요청을 만드십시오.

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

쿼리 매개 변수

다음 테이블에는 URL의 필수 매개 변수가 나와 있습니다.

매개 변수	설명
taskType	레코드에 대해 수행할 수 있는 작업 집합입니다. 태스크 유형에는 승인과 함께 업데이트, 선택적 승인과 함께 업데이트, 병합, 병합 해제, 검토(승인 없음), 최종 검토 및 거부된 레코드 업데이트가 포함됩니다.
businessEntity	비즈니스 항목의 이름입니다.

샘플 API 요청

다음 샘플 요청은 할당 가능한 사용자의 목록을 검색합니다.

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

샘플 API 응답

다음 샘플 응답은 태스크 유형 ReviewNoApprove에 대한 할당 가능한 사용자 목록을 보여 줍니다.

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

파일 메타데이터 나열

파일 메타데이터 나열 REST API는 저장소의 파일 메타데이터 목록을 반환합니다.

파일 메타데이터 나열 REST API는 BPM 또는 TEMP 저장소와 함께 사용합니다.

해당 API는 GET 메서드를 사용합니다.

요청 URL

파일 메타데이터 나열 URL의 형식은 다음과 같습니다.

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

파일 메타데이터 나열 URL에 대한 다음 HTTP GET 요청을 만드십시오.

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

샘플 API 요청

다음 샘플 요청은 TEMP 저장소에서 파일 메타데이터 목록을 검색합니다.

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

샘플 API 응답

다음 샘플 응답은 파일 메타데이터의 목록을 보여 줍니다.

```
{
    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”,
        },
        ...
    ]
}
```

파일 메타데이터 생성

파일 메타데이터 생성 REST API는 파일의 메타데이터를 생성하고 파일에 대한 파일 ID를 반환합니다.

파일 ID를 사용하여 파일을 업로드, 첨부, 업데이트, 다운로드 및 삭제할 수 있습니다.

파일 메타데이터 생성 REST API는 DB 또는 TEMP 저장소와 함께 사용합니다.

API는 POST 메서드를 사용합니다.

요청 URL

파일 메타데이터 생성 URL의 형식은 다음과 같습니다.

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

파일 메타데이터 생성 URL에 대한 다음 HTTP POST 요청을 만드십시오.

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

요청 본문

파일의 메타데이터를 지정합니다.

다음 테이블에는 요청 본문의 파일 메타데이터에 대한 매개 변수가 설명되어 있습니다.

매개 변수	설명
-fileName	파일의 이름입니다. 예를 들어 file.txt입니다.
fileType	파일 유형의 범주입니다. 예를 들어 text 또는 image입니다.
fileContentType	파일의 콘텐츠 유형입니다. 콘텐츠 유형은 /로 구분되는 유형 및 하위 유형으로 구성됩니다. 예를 들어 image/png입니다.

참고: 파일 메타데이터 생성 REST API 요청에 필요한 매개 변수는 저장소별로 다릅니다.

샘플 API 요청

다음 샘플 요청은 TEMP 저장소에 파일 메타데이터를 생성합니다.

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

{
    “fileName”: “file1.txt”,
    “fileType”: “text”,
    “fileContentType”: “text/plain”
}
```

샘플 API 응답

다음 예제는 파일 메타데이터를 TEMP 저장소에 성공적으로 생성한 후의 응답을 보여 줍니다. 이 API는 파일의 파일 ID를 반환합니다.

```
TEMP_SVR1.1VDVS
```

참고: 파일 ID의 형식은 <storage type>_<uniqueID>입니다.

파일 메타데이터 가져오기

파일 메타데이터 가져오기 REST API는 파일 ID에 연결된 파일의 메타데이터를 반환합니다.

파일 메타데이터 가져오기 REST API는 BPM, BUNDLE, DB 또는 TEMP 저장소와 함께 사용합니다.

이 API는 GET 메서드를 사용합니다.

요청 URL

파일 메타데이터 가져오기 URL의 형식은 다음과 같습니다.

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

파일 메타데이터 가져오기 URL에 대한 다음 HTTP GET 요청을 만드십시오.

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

샘플 API 요청

다음 샘플 요청은 파일 ID TEMP_SVR1.1VDVS를 사용하여 TEMP 저장소의 파일 메타데이터를 반환합니다.

```
GET http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/TEMP/TEMP_SVR1.1VDVS
```

다음 샘플 요청은 파일 ID besMetadata를 사용하여 BUNDLE 저장소의 리소스 번들 파일 메타데이터를 반환합니다.

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

샘플 API 응답

다음 샘플 응답은 TEMP 저장소에서 파일 ID가 TEMP_SVR1.1VDVS인 파일 메타데이터를 보여 줍니다.

```
{
    "fileName": "file1.txt",
    "fileType": "text",
    "fileContentType": "text/plain"
}
```

다음 샘플 응답은 BUNDLE 저장소에서 리소스 번들 파일 besMetadata의 메타데이터를 보여 줍니다.

```
{
    "fileName": "besMetadata.zip",
    "fileType": "BES Metadata Bundle",
```

```
    "fileContentType": "application/zip",
    "digest": "a08c5d97da7e6a780ed7c427ff14a8d2d396438cd65b654ad67424e226f64a41"
}
```

파일 메타데이터 업데이트

파일 메타데이터 업데이트 REST API는 파일 ID에 연결된 파일의 메타데이터를 업데이트합니다.

파일 메타데이터 업데이트 REST API는 DB 또는 TEMP 저장소와 함께 사용합니다.

이 API는 PUT 메서드를 사용합니다.

요청 URL

파일 메타데이터 업데이트 URL의 형식은 다음과 같습니다.

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

파일 메타데이터 업데이트 URL에 대한 다음 HTTP PUT 요청을 만드십시오.

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

샘플 API 요청

다음 샘플 요청은 파일 ID TEMP_SVR1.1VDVS를 사용하여 파일 메타데이터를 TEMP 저장소에서 업데이트합니다.

```
PUT http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/TEMP/TEMP_SVR1.1VDVS

{
    “fileName”: “file2.txt”,
    “fileType”: “text”,
    “fileContentType”: “text/plain”
}
```

다음 샘플 요청은 파일 ID DB_SVR1.0JU1를 사용하여 파일 메타데이터를 DB 저장소에서 업데이트합니다.

```
PUT http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/DB/DB_SVR1.0JU1
{
    "fileName": "Document_2.pdf",
    "fileType": "pdf",
    "fileContentType": "application/pdf"
}
```

샘플 API 응답

이 API는 파일 메타데이터 업데이트 후 200 OK 응답 코드를 반환합니다. 응답 본문은 비어 있습니다.

파일 콘텐츠 업로드

파일 콘텐츠 업로드 REST API는 파일 ID에 연결된 파일의 콘텐츠를 업데이트합니다.

파일 콘텐츠 업로드 REST API는 BUNDLE, DB 또는 TEMP 저장소와 함께 사용합니다.

이 API는 PUT 메서드를 사용합니다.

요청 URL

파일 콘텐츠 업로드 URL의 형식은 다음과 같습니다.

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

파일 콘텐츠 업로드 URL에 대한 다음 HTTP PUT 요청을 만드십시오.

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

샘플 API 요청

다음 샘플 요청은 파일 ID TEMP_SVR1.1VDVS를 사용하여 파일 콘텐츠를 TEMP 저장소에 업로드합니다.

```
PUT http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/TEMP/TEMP_SVR1.1VDVS/content

Test attachment content: file 1
```

다음 샘플 요청은 파일 ID DB_SVR1.0JU1를 사용하여 파일 콘텐츠를 DB 저장소에 업로드합니다.

```
PUT http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/DB/DB_SVR1.0JU1/content
Content-Type: application/octet-stream
<file object (upload using REST client)>
```

다음 샘플 요청은 파일 ID besMetadata를 사용하여 리소스 번들 파일 콘텐츠를 BUNDLE 저장소에 업로드합니다.

```
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
```

샘플 API 응답

이 API는 파일 콘텐츠 업로드 후 200 OK 응답 코드를 반환합니다. 응답 본문은 비어 있습니다.

파일 콘텐츠 가져오기

파일 콘텐츠 가져오기 REST API는 파일 ID에 연결된 파일의 콘텐츠를 반환합니다.

파일 콘텐츠 가져오기 REST API는 BPM, BUNDLE, DB 또는 TEMP 저장소와 함께 사용합니다.

이 API는 GET 메서드를 사용합니다.

요청 URL

파일 콘텐츠 가져오기 URL의 형식은 다음과 같습니다.

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

파일 콘텐츠 가져오기 URL에 대한 다음 HTTP GET 요청을 만드십시오.

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

샘플 API 요청

다음 샘플 요청은 파일 ID urn:b4p2:22203::file1.txt를 사용하여 BPM 저장소의 파일 콘텐츠를 반환합니다.

```
GET http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/BPM/urn:b4p2:22203::file1.txt/content
```

참고: BPM 저장소의 태스크 첨부 파일에 대한 파일 ID를 검색하려면 태스크 읽기 REST API를 사용합니다.

다음 샘플 요청은 파일 ID DB_SVR1.0JU1를 사용하여 DB 저장소의 파일 콘텐츠를 반환합니다.

```
GET http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/DB/DB_SVR1.0JU1/content
```

참고: 레코드에 첨부하는 파일의 파일 ID를 검색하려면 레코드 읽기 REST API를 사용합니다.

다음 샘플 요청은 파일 ID besMetadata를 사용하여 BUNDLE 저장소의 리소스 번들 파일 콘텐츠를 반환합니다.

```
GET http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/BUNDLE/besMetadata/content
```

샘플 API 응답

다음 샘플 응답은 BPM 저장소의 TXT 파일에 대한 콘텐츠를 보여 줍니다.

```
Test attachment content: file 1
```

다음 샘플 응답은 BUNDLE 저장소의 리소스 번들 파일에 대한 콘텐츠를 보여 줍니다.

```
Content-Disposition → attachment; filename=besMetadata.zip
Content-Type → application/octet-stream
Transfer-Encoding → chunked
```

파일 삭제

파일 삭제 REST API는 파일 메타데이터 및 콘텐츠를 포함하는 파일 ID에 연결된 파일을 삭제합니다.

파일 삭제 REST API는 BUNDLE, DB 또는 TEMP 저장소와 함께 사용합니다.

이 API는 DELETE 메서드를 사용합니다.

요청 URL

파일 삭제 URL의 형식은 다음과 같습니다.

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

파일 삭제 URL에 대한 다음 HTTP DELETE 요청을 만드십시오.

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

샘플 API 요청

다음 샘플 요청은 파일 ID TEMP_SVR1.1VDVS에 연결된 파일을 파일 메타데이터 및 콘텐츠를 포함하여 TEMP 저장소에서 삭제합니다.

```
DELETE http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/TEMP/TEMP_SVR1.1VDVS
```

다음 샘플 요청은 파일 ID DB_SVR1.0JU1에 연결된 파일을 파일 메타데이터 및 콘텐츠를 포함하여 DB 저장소에서 삭제합니다.

```
DELETE http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/DB/DB_SVR1.0JU1
```

다음 샘플 요청은 파일 ID가 besMetadata인 리소스 번들 파일을 파일 메타데이터 및 콘텐츠를 포함하여 BUNDLE 저장소에서 삭제합니다.

```
DELETE http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/BUNDLE/besMetadata
```

샘플 API 응답

이 API는 파일 삭제 후 200 OK 응답 코드를 반환합니다. 응답 본문은 비어 있습니다.

승격 미리 보기

승격 미리 보기 REST API는 사용자가 보류 중인 변경 내용을 승격시키는 경우 결과 레코드의 미리 보기를 반환합니다.

해당 API는 GET 메서드를 사용합니다. 보류 중인 변경 내용을 레코드에 적용하는 경우 레코드가 어떻게 나타나는지 볼 수 있습니다. 해당 API 응답에는 새 값이 있는 레코드 및 이전 값이 있는 변경 요약이 포함되어 있습니다. 이 API는 사용자가 삭제하는 데이터에 대한 정보를 반환하지 않습니다. URL의 보류 중인 변경 내용에 대한 상호작용 ID를 제공하십시오.

요청 URL

승격 미리 보기 URL의 형식은 다음과 같습니다.

```
http://<host>:<port>/<context>/<database ID><business entity>/<rowId>?
action=previewPromote&interactionID=<interaction ID>
```

승격 미리 보기 URL에 대한 다음 HTTP GET 요청을 만드십시오.

```
GET http://<host>:<port>/<context>/<database ID><business entity>/<rowId>?
action=previewPromote&interactionID=<interaction ID>
```

쿼리 매개 변수

보류 중인 변경 내용의 상호 작용 ID는 URL의 필수 매개 변수입니다.

다음 테이블에는 쿼리 매개 변수가 나와 있습니다.

매개 변수	설명
contentMetadata	병합 미리 보기에 대한 메타데이터입니다. 쉼표로 구분된 목록을 제공하십시오. 다음 값을 사용할 수 있습니다. - BVT. 병합 미리 보기에 사용할 가장 신뢰할 수 있는 값이 포함된 레코드의 rowid를 지정합니다. 교차 참조 레코드에 대한 정보와 원래 레코드 ID를 반환합니다. - MERGE. 병합할 레코드의 rowid를 지정합니다. 하위 레코드가 병합된 방법에 대한 정보를 반환합니다.
interactionId	보류 중인 변경 내용의 상호 작용 ID입니다.
effectiveDate	선택 사항입니다. 변경 내용을 미리 보려는 날짜입니다. 시간 표시 막대가 사용되는 기본 개체에 대한 매개 변수를 사용하십시오.

관련 항목:

- "날짜 및 시간 형식(UTC)" 페이지 27

샘플 API 요청

다음 샘플 요청은 Person 비즈니스 항목의 루트 레코드에 대한 미리 보기를 작성합니다.

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/233?
action=previewPromote&interactionId=72300000001000
```

샘플 API 응답

다음 샘플 응답은 새 값 및 이전 값의 변경 요약과 함께 레코드의 미리보기를 반환합니다.


```
{
    "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"
    }
}
```

승격

승격 REST API는 변경 요청의 상호 작용 ID에 따라 레코드에 대해 보류 중인 모든 변경 내용을 승격시킵니다.

해당 API는 POST 메서드를 사용합니다. 상호 작용 ID를 쿼리 매개 변수로 제공하십시오.

요청 URL

승격 URL의 형식은 다음과 같습니다.

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<rowId of the root record>?
action=promote&interactionId=<interaction ID>
```

승격 URL에 대한 다음 HTTP POST 요청을 만드십시오.

```
POST http://<host>:<port>/<context>/<database ID>/<business entity>/<rowId of the root record>?
action=promote&interactionId=<interaction ID>
```

쿼리 매개 변수

보류 중인 변경 내용의 상호 작용 ID는 필수 매개 변수입니다. 해당 API는 상호 작용 ID를 사용하여 보류 중인 변경 내용과 관련된 모든 레코드를 찾습니다.

샘플 API 요청

다음 샘플 요청은 변경 요청의 상호 작용 ID에 따라 보류 중인 변경 내용을 모두 승격시킵니다.

```
POST http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/1038246?
action=promote&interactionId=69120000294000
```

샘플 API 응답

다음 샘플 응답에는 보류 중인 변경 내용을 승격시킨 후 레코드의 행 ID가 포함되어 있습니다.

```
{
        Person: {
                rowidObject: "1038246"
            }
}
```

보류 중인 항목 삭제

보류 중인 항목 삭제 REST API는 변경 요청의 상호 작용 ID에 따라 레코드에 대해 보류 중인 모든 변경 내용을 삭제합니다.

해당 API는 DELETE 메서드를 사용하고 레코드의 행 ID를 반환합니다.

요청 URL

보류 중인 항목 삭제 URL의 형식은 다음과 같습니다.

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<rowid>?
action=deletePending&interactionId=<interaction ID>
```

보류 중인 항목 삭제 URL에 대한 다음 DELETE 요청을 만드십시오.

```
DELETE http://<host>:<port>/<context>/<database ID>/<business entity>/<rowid>?
action=deletePending&interactionId=<interaction ID>
```

쿼리 매개 변수

삭제하려는 보류 중인 변경 내용의 상호 작용 ID를 제공하십시오.

샘플 API 요청

다음 샘플 요청은 변경 요청의 상호 작용 ID에 따라 보류 중인 변경 내용을 모두 삭제합니다.

```
DELETE http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/233?
action=deletePending&interactionId=72300000001000
```

샘플 API 응답

다음 샘플 응답에는 보류 중인 변경 내용을 삭제한 후 레코드의 행 ID가 포함되어 있습니다.

```
{
    Person: {
        rowidObject: "233"
    }
}
```

병합 미리 보기

병합 미리 보기 REST API는 사용자가 두 개 이상의 루트 레코드를 병합하는 경우 통합된 루트 레코드의 미리 보기를 반환합니다.

해당 API는 POST 메서드를 사용하고 루트 레코드 및 필드 수준 재정의 목록을 허용하여 병합된 레코드의 미리 보기를 반환합니다. 대상 레코드의 행 ID는 필수 매개 변수입니다.

요청 URL

병합 미리 보기 URL의 형식은 다음과 같습니다.

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<rowid of the record>?action=previewMerge
```

다음 병합 미리 보기 URL 형식은 반환할 하위 수준 수를 지정합니다.

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<rowid of the record>?
action=previewMerge&depth=2
```

병합 미리 보기 URL에 대한 다음 HTTP POST 요청을 만드십시오.

```
POST http://<host>:<port>/<context>/<database ID>/<business entity>/<rowid of the record>?
action=previewMerge
```

참고: 요청 본문에 keys 속성을 추가하고, 대상 레코드와 병합할 루트 레코드를 지정합니다.

하위 레코드에 대한 일치 항목을 재정의하려면 contentMetadata 매개 변수를 추가합니다.

```
POST http://<host>:<port>/<context>/<database ID>/<business entity>/<rowid of the record>?
action=previewMerge&contentMetadata=MERGE/<json/xml>
```

참고: 요청 본문에 overrides 속성을 추가하고 병합 재정의를 지정합니다.

요청과 함께 보내려는 데이터의 미디어 유형을 지정하려면 콘텐츠 유형 헤더를 추가합니다.

```
POST http://<host>:<port>/<context>/<database ID>/<business entity>/<rowid of the record>?
action=previewMerge
Content-Type: application/<json/xml>
```

쿼리 매개 변수

대상 레코드의 행 ID는 필수 매개 변수입니다.

다음 쿼리 매개 변수를 사용할 수 있습니다.

매개 변수	설명
contentMetadata	병합 미리 보기에 대한 메타데이터입니다. 쉼표로 구분된 목록을 제공하십시오. 다음 값을 사용할 수 있습니다. - BVT. 우위 교차 참조 레코드에 대한 정보와 원래 레코드 ID를 반환합니다. - MERGE. 하위 레코드가 병합된 방법에 대한 정보를 반환합니다.
depth	반환할 하위 수준 수입니다.
effectiveDate	미리보기를 생성하려는 날짜입니다.

관련 항목:

- "날짜 및 시간 형식(UTC)" 페이지 27

요청 본문

시작하기 전에 일치된 레코드 읽기 API를 사용하여, 원래 루트 레코드와 병합할 수 있는 일치된 레코드를 확인합니다. 병합 미리 보기 API의 요청 본문에 레코드 목록을 보냅니다.

루트 레코드의 필드 값을 재정의할 수 있습니다. 예를 들어 병합된 루트 레코드 모두에 이름의 올바른 철자가 포함되어 있지 않으면 해당 이름을 요청 본문에 지정할 수 있습니다. 병합 후보를 제거하거나 다른 병합 후보를 지정하여 하위 레코드를 병합하는 방법을 재정의할 수도 있습니다.

요청 본문에 다음 속성을 사용합니다.

속성/요소	유형	설명
keys	배열	필수 사항입니다. 병합에 추가할 유사한 루트 레코드의 순서가 지정된 목록입니다. 행 ID 또는 소스 시스템과 소스 키의 조합으로 레코드를 식별할 수 있습니다.
overrides	개체	루트 레코드의 필드 값 및 하위 레코드의 일치 항목을 재정의합니다.
MERGE	개체	하위 레코드가 병합되는 방법을 재정의합니다. overrides 개체 내에 하위 레코드의 유형을 추가한 다음 MERGE 개체를 추가합니다.

다음 JSON 코드 샘플은 대상 루트 레코드와 병합할 루트 레코드를 식별합니다.

```
{
  keys: [
    {
      rowid: "P2"
    }
  ]
}
```

다음 코드는 Party 루트 레코드의 필드를 재정의하는 방법 및 Telephone 하위 레코드의 병합 후보를 재정의하는 방법을 보여 줍니다.

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

샘플 API 요청

다음 샘플 요청은 통합된 레코드의 미리 보기를 반환합니다.

```
POST http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/2478245?action=previewMerge
```

```
{
  keys: [
    {
        rowid: "2478246"
    },
   {
        rowid: "2478230"
    }
  ],
  overrides: {
    Person: {
       firstName: "Charlie"
    }
  }
}
```

샘플 API 응답

다음 샘플 응답은 통합된 레코드의 미리보기를 보여 줍니다.

```
{
    "Person": {
        "rowidObject": "2478245       ",
        "partyType": "Person",
        "lastName": "Smith",
        "firstName": "Charlie",
        "displayName": "ALICE SMITH"
    }
}
```

보류 중인 병합

보류 중인 병합 REST API는 변경 요청의 상호 작용 ID에 따라 레코드에 대해 보류 중인 모든 병합 태스크를 업데이트합니다. 보류 중인 병합을 사용하면 워크플로우 프로세스가 모든 병합 태스크를 승인하기 전까지 병합 작업을 연기할 수 있습니다.

해당 API는 POST 메서드를 사용하고 레코드의 행 ID를 반환합니다.

요청 URL

보류 중인 병합 URL의 형식은 다음과 같습니다.

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<rowid>?
action=PendingMerge&interactionId=<interaction ID>
```

보류 중인 병합 URL에 대한 다음 HTTP POST 요청을 만드십시오.

```
POST http://<host>:<port>/<context>/<database ID>/<business entity>/<rowid of the record>?action=Merge
```

쿼리 매개 변수

보류 중인 병합의 상호 작용 ID는 필수 매개 변수입니다.

샘플 API 요청

다음 샘플 요청은 상호 작용 ID와 연결된 모든 보류 중인 병합 태스크를 업데이트합니다.

```
POST /Person/123?action=pendingMerge&interactionId=123
```

샘플 API 응답

다음 샘플 응답에는 영향받은 루트 기본 개체의 행 ID가 포함되어 있습니다.

```
{
    keys: [{rowid: "456"}, {rowid: "789"}],
    overrides: {...}
}
```

PromoteMerge

병합 승격 REST API는 변경 요청의 상호 작용 ID에 연결된 모든 보류 중인 병합 태스크를 실행합니다.

해당 API는 POST 메서드를 사용하고 우위 레코드의 행 ID를 반환합니다.

요청 URL

병합 승격 URL의 형식은 다음과 같습니다.

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<rowid>?
action=PromoteMerge&interactionId=<interaction ID>
```

병합 승격 URL에 대한 다음 HTTP POST 요청을 만드십시오.

```
POST http://<host>:<port>/<context>/<database ID>/<business entity>/<rowid of the record>?action=Merge
```

쿼리 매개 변수

보류 중인 병합 태스크의 상호 작용 ID는 필수 매개 변수입니다. 해당 API는 상호 작용 ID를 사용하여 모든 보류 중인 병합 태스크를 찾고 병합을 실행합니다.

샘플 API 요청

다음 샘플 요청은 상호 작용 ID와 연결된 모든 보류 중인 병합 태스크를 승격합니다.

```
POST /Person/123?action=promoteMerge&interactionId=123
```

샘플 API 응답

다음 샘플 응답에는 보류 중인 병합 태스크를 승격시킨 후 레코드의 행 ID가 포함되어 있습니다.

```
POST /Person/123?action=promoteMerge&interactionId=123
```

레코드 병합

레코드 병합 REST API는 두 개 이상의 루트 레코드를 병합하여 하나의 통합된 레코드를 작성합니다. 통합된 레코드의 행 ID는 다른 레코드와 병합할 레코드의 행 ID입니다.

해당 API는 POST 메서드를 사용합니다. 요청 본문의 병합된 레코드에 대한 필드 수준 재정의를 지정할 수 있습니다.

요청 URL

레코드 병합 URL의 형식은 다음과 같습니다.

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<rowid of the record>?action=Merge
```

레코드 병합 URL에 대한 다음 HTTP POST 요청을 만드십시오.

```
POST http://<host>:<port>/<context>/<database ID>/<business entity>/<rowid of the record>?action=Merge
```

콘텐츠 유형 헤더를 추가하여 요청과 함께 보내려는 데이터의 미디어 유형을 지정하십시오.

```
POST http://<host>:<port>/<context>/<database ID>/<business entity>/<rowid of the record>?action=Merge
Content-Type: application/<json/xml>
```

쿼리 매개 변수

다음 테이블에는 URL에서 사용할 수 있는 매개 변수가 나와 있습니다.

매개 변수	설명
taskComment	API에 의해 트리거되는 워크플로우 태스크에 설명을 추가합니다.
taskAttachments	태스크 첨부 파일이 활성화된 경우 API에 트리거되는 워크플로우 태스크에 파일을 첨부합니다.

요청 본문

시작하기 전에 병합 미리 보기 API를 사용하여, 선택한 루트 레코드 병합 결과를 미리 봅니다. 미리 보기 결과가 만족스러우면 레코드 병합 API에 대한 요청 본문에 동일한 속성을 사용합니다.

루트 레코드의 필드 값을 재정의할 수 있습니다. 예를 들어 병합된 루트 레코드 모두에 이름의 올바른 철자가 포함되어 있지 않으면 해당 이름을 요청 본문에 지정할 수 있습니다. 병합 후보를 제거하거나 다른 병합 후보를 지정하여 하위 레코드를 병합하는 방법을 재정의할 수도 있습니다.

요청 본문에 다음 속성을 사용합니다.

속성/요소	유형	설명
keys	배열	필수 사항입니다. 병합에 추가할 유사한 루트 레코드의 순서가 지정된 목록입니다. 행 ID 또는 소스 시스템과 소스 키의 조합으로 레코드를 식별할 수 있습니다.
overrides	개체	루트 레코드의 필드 값 및 하위 레코드의 일치 항목을 재정의합니다.
MERGE	개체	하위 레코드가 병합되는 방법을 재정의합니다. overrides 개체 내에 하위 레코드의 유형을 추가한 다음 MERGE 개체를 추가합니다.

다음 JSON 코드 샘플은 대상 루트 레코드와 병합할 루트 레코드 두 개를 식별합니다.

```
{
  keys: [
    {rowid: "2478246"},
    {rowid: "2478269"}
  ]
}
```

레코드 병합 API에서 overrides 및 MERGE 속성을 사용하는 방법의 예제를 보려면 병합 미리 보기 API에 대한 요청 본문을 참조하십시오.

샘플 API 요청

다음 샘플 요청은 통합된 레코드에 레코드를 병합합니다. 이 요청은 API에 의해 트리거되는 워크플로우 태스크에 설명 및 첨부 파일을 추가합니다.

```
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"
    }
  }
}
```

샘플 API 응답

다음 샘플 응답은 통합된 레코드를 보여 줍니다.

```
{
    "Person": {
        "link": [
            {
                "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/2478245",
```

```
                "rel": "self"
            }
        ],
        "key": {
            "rowid": "2478245"
        },
        "rowidObject": "2478245"
    }
}
```

레코드 병합 해제

레코드 병합 해제 REST API는 사용자가 레코드를 병합하기 전에 존재한 개별 루트 레코드로 루트 레코드를 병합 해제합니다.

해당 API는 POST 메서드를 사용합니다.

요청 URL

레코드 병합 해제 URL의 형식은 다음과 같습니다.

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

레코드 병합 해제 URL에 대한 다음 HTTP POST 요청을 만드십시오.

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

콘텐츠 유형 헤더를 추가하여 요청과 함께 보내려는 데이터의 미디어 유형을 지정하십시오.

```
POST http://<host>:<port>/<context>/<database ID>/<business entity>/<rowId>?action=unmerge
Content-Type: application/<json/xml>
```

쿼리 매개 변수

다음 테이블에는 URL에서 사용할 수 있는 매개 변수가 나와 있습니다.

매개 변수	설명
taskComment	API에 의해 트리거되는 워크플로우 태스크에 설명을 추가합니다.
taskAttachments	태스크 첨부 파일이 활성화된 경우 API에 트리거되는 워크플로우 태스크에 파일을 첨부합니다.

요청 본문

요청 본문의 통합된 레코드에서 병합 해제하려는 레코드 목록을 보내십시오. xref 행 ID 또는 소스 시스템과 소스 키를 사용하여 레코드를 지정하십시오.

레코드 읽기 API를 사용하여 병합 해제할 레코드의 xref rowId를 가져올 수 있습니다. 다음 샘플 요청은 레코드의 XREF 메타데이터를 검색합니다.

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/2638243?contentMetadata=XREF
```

샘플 API 요청

다음 샘플 요청은 통합된 레코드에서 레코드를 병합 해제합니다. 이 요청은 API에 의해 트리거되는 워크플로우 태스크에 설명 및 첨부 파일을 추가합니다.

```
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"
}
```

샘플 API 응답

다음 샘플 응답은 사용자가 통합된 레코드에서 병합 해제하는 레코드를 보여 줍니다.

```
{
    "Person": {
        "link": [
            {
                "href": "http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/2478249",
                "rel": "self"
            }
        ],
        "key": {
            "rowid": "2478249       "
        },
        "rowidObject": "2478249       "
    }
}
```

관계 읽기

관계 읽기 REST API는 두 레코드의 표시 이름, rowID 및 파티 유형 같이 관계 레코드의 세부 정보를 반환합니다.

해당 API는 GET 메서드를 사용합니다.

요청 URL

관계 읽기 URL의 형식은 다음과 같습니다.

```
http://<host>:<port>/<context>/<database ID>/<relationship>/<row ID of the relationship record>
```

URL에 대한 다음 HTTP GET 요청을 만드십시오.

```
GET http://<host>:<port>/<context>/<database ID>/<relationship>/<row ID of the relationship record>
```

쿼리 매개 변수

다음 테이블에는 쿼리 매개 변수가 나와 있습니다.

매개 변수	설명
suppressLinks	선택 사항입니다. API 응답에 상위-하위 링크가 표시되는지 여부를 나타냅니다. 응답에서 모든 상위-하위 링크를 억제하려면 매개 변수를 true로 설정합니다. API 응답에 링크를 표시하지 않으려면 매개 변수를 false로 설정합니다. 기본값은 false입니다.
depth	선택 사항입니다. 반환할 하위 수준 수입니다.

샘플 API 요청

다음 샘플 요청은 행 ID가 85이고 관계 유형이 ProductGroupProductGroupIsParentOfProductProducts인 관계 레코드의 세부 정보를 반환합니다.

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/
ProductGroupProductGroupIsParentOfProductProducts/85
```

다음 샘플 요청은 depth가 2인 세부 정보를 반환합니다.

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/
ProductGroupProductGroupIsParentOfProductProducts/85?depth=2
```

샘플 API 응답

다음 예제는 행 ID가 85이고 관계 유형이 ProductGroupProductGroupIsParentOfProductProducts인 관계 레코드의 세부 정보를 보여 줍니다.


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


관계 생성

관계 생성 REST API는 지정된 레코드 간에 관계를 생성합니다. 레코드 간에 관계를 생성하려면 해당 레코드가 속해 있는 비즈니스 항목 간에 관계가 있어야 합니다. 예를 들어 Informatica와 John Smith 간에 관계를 지정하려면 Organization 및 Person 비즈니스 항목 간에 관계가 있어야 합니다. 관계 데이터를 요청 본문에 보내야 합니다.

API는 PUT 메서드와 POST 메서드를 사용합니다.

요청 URL

관계 생성 REST URL의 형식은 다음과 같습니다.

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

참고: 소스 시스템의 이름은 URL의 필수 매개 변수입니다.

URL에 대한 다음 HTTP POST 또는 PUT 요청을 만드십시오.

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

콘텐츠 유형 헤더를 추가하여 요청과 함께 보내려는 데이터의 미디어 유형을 지정하십시오.

```
Content-Type: application/<json/xml>
```

URL 매개 변수

소스 시스템의 이름은 요청 URL의 필수 매개 변수입니다.

요청 본문

관계 레코드의 데이터를 REST 요청 본문에 보냅니다. JSON 형식 또는 XML 형식을 사용하여 데이터를 보냅니다. 요청 본문에 필수 매개 변수 값을 제공합니다.

샘플 API 요청

다음 샘플 요청은 행 ID가 101인 Organization 비즈니스 항목과 행 ID가 1101인 Person 비즈니스 항목 간에 OrganizationEmploysPerson 관계를 생성합니다.

```
POST http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/OrganizationEmploysPerson?systemName=SFA
Content-Type: application/json

{
  "from": {
    "rowidObject": "101"
    },

  "to": {
    "rowidObject": "1101          "
    },

    "relName": "Documentation",
    "relDesc": "Writer"
}
```

OrganizationEmploysPerson 관계는 Organization 비즈니스 항목에서 Person 비즈니스 항목으로의 관계를 정의합니다. from 요소는 관계가 시작되는 레코드를 지정하고 to 요소는 관계가 끝나는 레코드를 지정합니다.

샘플 API 응답

다음 샘플 응답은 행 ID가 101인 Organization 비즈니스 항목과 행 ID가 1101인 Person 비즈니스 항목 간에 관계를 생성한 이후의 응답 헤더 및 본문을 보여 줍니다.

```
{
        "OrganizationEmployesPerson": {
        "key": {
        "rowid": "414721"
        "sourceKey": "SVR1.1E7UW"
 }-

"rowidObject": "414721"

   "from": {
```

```
                    "key": {
                                "rowid": "101 "
                                                    }-
          "rowidObject": "101 "
          }-

    "to": {
                  "key": {
                              "rowid": "1101 "
          }-

   "rowidObject": "1101 "
                  }-

      }-

   }
```

관계 업데이트

관계 업데이트 REST API는 두 레코드 간의 관계를 업데이트합니다. API는 관계에 대해 정의된 추가 특성을 업데이트합니다.

API는 POST 메서드와 PUT 메서드를 사용합니다.

요청 URL

관계 업데이트 URL의 형식은 다음과 같습니다.

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

참고: 소스 시스템의 이름은 URL의 필수 매개 변수입니다.

URL에 대한 다음 HTTP POST 또는 PUT 호출을 만드십시오.

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

콘텐츠 유형 헤더를 추가하여 요청과 함께 보내려는 데이터의 미디어 유형을 지정하십시오.

요청 본문

관계 레코드에 대한 업데이트를 요청 본문에 보냅니다. JSON 형식 또는 XML 형식을 사용하여 데이터를 보냅니다. 요청 본문에 필수 매개 변수 값을 제공합니다.

샘플 API 요청

행 ID가 414721인 관계는 행 ID가 101인 Organization 항목과 행 ID가 1101인 Person 항목 간의 OrganizationEmploysPerson 관계입니다.

다음 샘플 요청은 행 ID가 414721인 관계 레코드를 업데이트합니다.

```
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"}


}
```

샘플 API 응답

다음 샘플 응답은 행 ID가 414721인 관계를 업데이트한 이후에 수신됩니다.

```
{
        "OrganizationEmployesPerson": {
                "key": {
                                "rowid": "414721"
                                "sourceKey": "SVR1.1E7UW"
                                        }-
            "rowidObject": "414721"
        "from": {
            "key": {
                            "rowid": "101"
                                    }-
                "rowidObject": "101"
                }-
        "to": {
                "key": {
                            "rowid": "1101 "
                               }-
        "rowidObject": "1101 "
                        }-
            }-
}
```

관계 삭제

관계 삭제 REST API는 두 레코드 간의 관계를 삭제합니다.

API는 DELETE 메서드를 사용합니다.

요청 URL

관계 삭제 URL의 형식은 다음과 같습니다.

```
http://<host>:<port>/<context>/<database ID>/<relationship>/<rowID of the relationship record>?
systemName=<name of the source system>
```

참고: 소스 시스템의 이름은 URL의 필수 매개 변수입니다.

삭제 URL에 대한 다음 HTTP DELETE 요청을 만드십시오.

```
DELETE http://<host>:<port>/<context>/<database ID>/<relationship>/<rowID of the relationship record>?
systemName=<name of the source system>
```

쿼리 매개 변수

소스 시스템의 이름은 필수 URL 매개 변수입니다. systemName 매개 변수를 사용하여 소스 시스템을 지정합니다.

샘플 API 요청

다음 샘플 요청은 행 ID가 414721인 관계 레코드를 삭제합니다.

```
DELETE http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/OrganizationEmploysPerson/414721?systemName=SFA
```

샘플 API 응답

다음 샘플 응답은 행 ID가 414721인 관계 레코드를 삭제한 이후의 응답을 보여 줍니다.

```
{
"OrganizationEmployesPerson": {
        "key": {
                "rowid": "414721"
                }-
"rowidObject": "414721"
            }-
}
```

관련 레코드 가져오기

관련 레코드 가져오기 REST API는 사용자가 구성한 관계를 기반으로 지정된 루트 레코드에 관련된 레코드 목록을 반환합니다. API는 관계에 대한 세부 정보도 반환합니다.

해당 API는 GET 메서드를 사용합니다.

요청 URL

관련 레코드 가져오기 URL의 형식은 다음과 같습니다.

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

관련 레코드 가져오기 URL에 대한 다음 HTTP GET 요청을 만드십시오.

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

쿼리 매개 변수

쿼리 매개 변수를 요청 URL에 추가할 수 있습니다.

다음 테이블에는 사용할 수 있는 쿼리 매개 변수가 나열되어 있습니다.

매개 변수	설명
recordsToReturn	읽을 many 하위 항목의 레코드 수입니다.
searchToken	결과 집합의 후속 페이지를 가져올 검색 토큰입니다.
returnTotal	결과 집합의 레코드 수를 반환합니다. 결과 집합의 레코드 수를 가져오려면 true로 설정합니다. 기본값은 false입니다.

필터 매개 변수

매개 변수를 URL에 추가하여 관련 레코드를 필터링할 수 있습니다.

다음 테이블에는 사용할 수 있는 필터 매개 변수가 나열되어 있습니다.

매개 변수	설명
recordStates	검색할 레코드 상태의 쉼표로 구분된 목록입니다. 지원되는 레코드 상태는 ACTIVE, PENDING 및 DELETED입니다. 기본값은 ACTIVE입니다. 예를 들어 /Party/123?action=related&recordStates=ACTIVE,PENDING 쿼리는 활성 또는 보류 중 상태의 레코드를 반환합니다.
entityLabel	항목의 레이블입니다.
relationshipLabel	관계의 레이블입니다.
entityType	항목 유형의 쉼표로 구분된 목록입니다. 예를 들어 entityType=Person,Organization 목록은 Person 및 Organization 항목 유형의 관련 레코드를 반환합니다.
relationshipType	관계 유형의 쉼표로 구분된 목록입니다. 예를 들어 relationshipType=Employee,Employer 목록은 Employee 및 Employer 관계 유형의 관련 레코드를 반환합니다.

참고: 필터 조건을 여러 개 지정하면 AND 조건을 충족하는 모든 레코드가 결과에 포함됩니다.

응답 본문

응답 본문에는 관련 레코드의 목록, 관계 및 관련 레코드의 세부 정보, 검색 토큰이 포함되어 있습니다. 검색 토큰을 사용하여 결과의 후속 페이지를 가져오십시오.

샘플 API 요청

다음 샘플 요청은 행 ID가 101인 Organization 비즈니스 항목에 대해 구성된 관련 레코드와 관계를 가져옵니다.

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Organization/101?action=related
```

샘플 API 응답

다음 샘플 응답은 행 ID가 101인 Organization 비즈니스 항목 유형에 대한 관련 레코드와 관계를 보여 줍니다.


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

일치된 레코드 읽기

일치된 레코드 읽기 REST API는 지정된 루트 레코드와 일치하는 레코드를 반환합니다. 레코드 목록을 검토하여 원래 루트 레코드와 병합할 수 있는 레코드를 결정할 수 있습니다. 레코드 병합 API를 사용하여 레코드를 병합할 수 있습니다.

해당 API는 GET 메서드를 사용합니다.

요청 URL

일치된 레코드 읽기 URL의 형식은 다음과 같습니다.

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

일치된 레코드 읽기 URL에 대한 다음 HTTP GET 요청을 만드십시오.

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

응답 본문

응답 본문에는 지정된 레코드와 일치하는 레코드 수, 일치하는 레코드의 세부 정보 및 검색 토큰이 포함되어 있습니다. 검색 토큰을 사용하여 일치 결과의 후속 페이지를 가져오십시오.

샘플 API 요청

다음 샘플 요청은 레코드와 일치하는 레코드를 비즈니스 항목에서 검색합니다.

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/1038245?action=matched
```

샘플 API 응답

다음 샘플 응답은 지정된 레코드와 일치하는 레코드의 세부 정보를 보여 줍니다.

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

일치된 레코드 업데이트

일치된 레코드 업데이트 REST API는 일치 테이블의 레코드를 작성하거나 업데이트합니다. 일치 테이블에는 비즈니스 항목에 대해 일치 프로세스를 실행한 후 비즈니스 항목의 일치된 레코드 쌍이 포함됩니다. 해당 API를 사용하여 지정된 레코드와 병합할 수 있는 레코드를 추가하십시오.

해당 API는 PUT 메서드를 사용합니다.

요청 URL

일치된 레코드 업데이트 URL의 형식은 다음과 같습니다.

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

일치된 레코드 업데이트 URL에 대한 다음 HTTP PUT 요청을 만드십시오.

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

콘텐츠 유형 헤더를 추가하여 요청과 함께 보내려는 데이터의 미디어 유형을 지정하십시오.

```
PUT http://<host>:<port>/<context>/<database ID>/<business entity>/<rowId>?action=matched
Content-Type: application/<json/xml>
```

요청 본문

요청 본문의 지정된 레코드와 일치하는 레코드 목록을 보내십시오. 행 ID 또는 소스 시스템과 소스 키를 사용하여 레코드를 지정하십시오.

샘플 API 요청

다음 샘플은 일치 테이블에서 레코드를 추가합니다.

```
PUT http://localhost:8080/cmx/cs/localhost-ORCL-DS_UI1/Person/1038245?action=matched

{
    keys: [
       {
          rowid: "1038246"
       }
```

```
    ]
}
```

샘플 API 응답

해당 API는 일치 테이블에서 성공적으로 레코드 작성 시 200 OK 응답을 반환합니다. 응답 본문은 비어 있습니다.

일치된 레코드 삭제

일치된 레코드 삭제 REST API는 일치 테이블에서 루트 레코드와 연결된 일치된 레코드를 삭제합니다.

해당 API는 DELETE 메서드를 사용합니다.

요청 URL

일치된 레코드 삭제 URL의 형식은 다음과 같습니다.

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

일치된 레코드 삭제 URL에 대한 다음 HTTP DELETE 요청을 만드십시오.

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

콘텐츠 유형 헤더를 추가하여 요청과 함께 보내려는 데이터의 미디어 유형을 지정하십시오.

```
DELETE http://<host>:<port>/<context>/<database ID>/<business entity>/<rowId>?action=matched
Content-Type: application/<json/xml>
```

요청 본문

요청 본문의 일치 테이블에서 삭제하려는 레코드 목록을 보내십시오.

샘플 API 요청

다음 샘플 요청은 일치 테이블에서 지정된 루트 레코드와 일치하는 레코드를 삭제합니다.

```
DELETE http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/1038245?action=matched
{
    keys: [
       {
          rowid: "1038246"
       }
    ]
}
```

샘플 API 응답

해당 API는 일치 테이블에서 성공적으로 레코드 삭제 시 200 OK 응답을 반환합니다. 응답 본문은 비어 있습니다.

레코드 기록 이벤트 가져오기

레코드 기록 이벤트 가져오기 REST API는 레코드에 연결된 기록 이벤트의 목록 또는 기록 이벤트의 그룹을 반환합니다. 요청 본문의 레코드 ID를 전송합니다.

API는 GET 메서드를 사용하여 기록 이벤트 그룹 각각에 대해 다음의 데이터를 반환합니다.

- 그룹의 시작 날짜와 종료 날짜

- 그룹 내의 이벤트 수

API는 각 기록 이벤트에 대해 다음의 데이터를 반환합니다.

- 기록 이벤트 ID
- 변경 날짜
- 변경을 수행한 사용자
- 변경 내용의 영향을 받는 기록 테이블의 목록
- 변경 내용의 영향을 받는 레코드 노드의 목록

요청 URL

레코드 기록 이벤트 가져오기 URL의 형식은 다음과 같습니다.

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

레코드 기록 이벤트 가져오기 URL에 대한 다음 HTTP GET 요청을 만드십시오.

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

쿼리 매개 변수

레코드의 ID는 필수 매개 변수입니다. API는 레코드 ID를 사용하여 모든 관련 기록 이벤트를 찾습니다.

다음 테이블에는 쿼리 매개 변수가 나와 있습니다.

매개 변수	설명
startDate 및 endDate	선택 사항입니다. 데이터를 검색하려는 날짜 범위입니다. 날짜 범위를 지정하면 이 범위 내의 이벤트만 응답에 포함됩니다.
granularity	선택 사항입니다. 기록 이벤트를 그룹화할 세부 정보 수준입니다. 지정할 경우 응답은 기록 이벤트를 그룹화합니다. 그렇지 않을 경우 응답은 기록 이벤트를 그룹화하지 않습니다. 다음 값 중 하나를 사용합니다. - YEAR - QUARTER - MONTH - WEEK - DAY - HOUR - MINUTE - AUTO
recordStates	선택 사항입니다. 기록 이벤트 목록에 반환되는 레코드 상태입니다. 쉼표로 구분된 목록을 제공하십시오. 다음 값을 사용할 수 있습니다. - ACTIVE - PENDING - DELETED

매개 변수	설명
contentMetadata	선택 사항입니다. 기록 이벤트 목록의 메타데이터입니다. 쉼표로 구분된 목록을 제공하십시오. 다음 값을 사용할 수 있습니다. - XREF - PENDING_XREF - DELETED_XREF - HISTORY - MATCH - BVT - TRUST
children	선택 사항입니다. 쉼표로 구분된 하위 노드 이름의 목록입니다. 지정할 경우, 하위 노드 이름이 응답에 포함됩니다.
order	선택 사항입니다. 필드 이름의 쉼표로 구분된 목록입니다(선택적 접두사 + 또는 -). + 접두사는 결과를 오름차순으로 정렬함을 나타내고 - 접두사는 결과를 내림차순으로 정렬함을 나타냅니다. 기본값은 +입니다. 매개 변수를 두 개 이상 지정하면 결과 집합은 목록의 첫 번째 매개 변수부터 차례대로 순서가 지정됩니다.
fields	선택 사항입니다. 비즈니스 항목 필드의 쉼표로 구분된 목록입니다. 지정할 경우, 나열된 필드만 응답에 포함됩니다.
filter	선택 사항입니다. 필터 식입니다.
depth	선택 사항입니다. 반환할 하위 수준 수입니다.
recordsToReturn	선택 사항입니다. 반환할 행 수를 지정합니다.
searchToken	선택 사항입니다. 이전 요청에서 반환된 검색 토큰을 지정합니다.
returnTotal	선택 사항입니다. `true`로 설정할 경우 결과의 레코드 수를 반환합니다. 기본값은 `false`입니다.
firstRecord	선택 사항입니다. 결과의 첫 번째 행을 지정합니다.
changeType	선택 사항입니다. 결과에 반환되는 변경 유형을 지정합니다. 쉼표로 구분된 목록을 제공하십시오. 다음 값을 사용할 수 있습니다. - BO - XREF - BVT - MERGE - MERGE_AS_SOURCE - MERGE_AS_TARGET - UNMERGE_AS_SOURCE - UNMERGE_AS_TARGET

관련 항목:

- "날짜 및 시간 형식(UTC)" 페이지 27

샘플 API 요청

다음 샘플 요청은 2000년 1월 1일부터 레코드에 대한 모든 병합을 연도별로 그룹화하여 반환합니다.

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/123?
action=listHistoryEvents&startDate=2010-01-01&granularity=YEAR&depth=2&changeType=MERGE
```

샘플 API 응답

다음 샘플 응답은 지정된 레코드에 대한 2000년 1월 1일 이후의 병합을 나열합니다.

```
{
    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
    ]
}
```

이벤트 세부 정보 가져오기

이벤트 세부 정보 가져오기 REST API는 레코드에 연결된 특정 기록 이벤트의 세부 정보를 반환합니다. API는 변경 유형, 변경 전과 후의 값 같은 세부 정보를 반환합니다. 요청 본문의 레코드 ID와 기록 이벤트 ID를 전송합니다.

해당 API는 GET 메서드를 사용합니다.

요청 URL

이벤트 세부 정보 가져오기 URL의 형식은 다음과 같습니다.

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

이벤트 세부 정보 가져오기 URL에 대한 다음 HTTP GET 요청을 만드십시오.

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

쿼리 매개 변수

시작하기 전에 레코드 기록 이벤트 가져오기 API를 사용하여 레코드에 연결된 기록 이벤트를 나열합니다. 반환되는 결과에서 기록 이벤트 ID와 레코드 ID를 쿼리 매개 변수로 사용합니다.

다음 테이블에는 쿼리 매개 변수가 나와 있습니다.

매개 변수	설명
eventID	필수 사항입니다. 레코드 기록 이벤트 가져오기 API는 기록 이벤트의 이벤트 ID를 반환합니다.
recordStates	선택 사항입니다. 기록 이벤트 목록에 반환되는 레코드 상태입니다. 쉼표로 구분된 목록을 제공하십시오. 다음 값을 사용할 수 있습니다. - ACTIVE - PENDING - DELETED
contentMetadata	선택 사항입니다. 기록 이벤트 목록의 메타데이터입니다. 쉼표로 구분된 목록을 제공하십시오. 다음 값을 사용할 수 있습니다. - XREF - PENDING_XREF - DELETED_XREF - HISTORY - MATCH - BVT - TRUST
children	선택 사항입니다. 쉼표로 구분된 하위 노드 이름의 목록입니다. 지정할 경우, 하위 노드 이름이 응답에 포함됩니다.
order	선택 사항입니다. 필드 이름의 쉼표로 구분된 목록입니다(선택적 접두사 + 또는 -). + 접두사는 결과를 오름차순으로 정렬함을 나타내고 - 접두사는 결과를 내림차순으로 정렬함을 나타냅니다. 기본값은 +입니다. 매개 변수를 두 개 이상 지정하면 결과 집합은 목록의 첫 번째 매개 변수부터 차례대로 순서가 지정됩니다.
fields	선택 사항입니다. 비즈니스 항목 필드의 쉼표로 구분된 목록입니다. 지정할 경우, 나열된 필드만 응답에 포함됩니다.
filter	선택 사항입니다. 필터 식입니다.
depth	선택 사항입니다. 반환할 하위 수준 수입니다.
recordsToReturn	선택 사항입니다. 반환할 행 수를 지정합니다.
searchToken	선택 사항입니다. 이전 요청에서 반환된 검색 토큰을 지정합니다.
returnTotal	선택 사항입니다. `true`로 설정할 경우 결과의 레코드 수를 반환합니다. 기본값은 `false`입니다.
firstRecord	선택 사항입니다. 결과의 첫 번째 행을 지정합니다.

샘플 API 요청

다음 샘플 요청은 기록 이벤트에 대한 정보를 반환합니다.

```
GET http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/123?
action=getHistoryEventDetails&eventId=2016-07-14T02%3A01%3A24.529%2B0000
```

샘플 API 응답

다음 샘플 응답은 지정된 이벤트에 대한 세부 정보를 보여 줍니다.


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


DaaS 메타데이터 가져오기

DaaS 메타데이터 가져오기 REST API는 이름, 유형, 함께 작업하는 비즈니스 항목, 필수 필드 목록 등과 같이 DaaS 공급자에 대한 정보를 반환합니다.

해당 API는 GET 메서드를 사용합니다.

요청 URL

DaaS 메타데이터 가져오기 URL의 형식은 다음과 같습니다.

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

DaaS 메타데이터 가져오기 URL에 대한 다음 HTTP GET 요청을 만드십시오.

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

쿼리 매개 변수

providerName 매개 변수는 필수 매개 변수입니다. 매개 변수는 구성된 DaaS 공급자의 이름입니다.

샘플 API 요청

다음 샘플 요청은 dcp DaaS 공급자의 메타데이터 정보를 반환합니다.

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

다음 샘플 요청은 ondemand DaaS 공급자의 메타데이터 정보를 반환합니다.

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

샘플 API 응답

다음 예제는 dcp DaaS 공급자에 대한 JSON 형식의 메타데이터 정보를 보여 줍니다.

```
{
                providerName: "dcp",
                providerType: "READ",
                businessEntity: "Organization",
                systemName: "SFA",
                requiredFields: [
                                "dunsNumber"
                ]
}
```

다음 예제는 ondemand DaaS 공급자에 대한 JSON 형식의 메타데이터 정보를 보여 줍니다.

```
{
                providerName: "ondemand",
                providerType: "SEARCH",
                businessEntity: "Organization",
                systemName: "SFA",
                requiredFields: [
                            "displayName"
                ]
}
```

DaaS 검색

DaaS 검색 REST API는 비즈니스 항목의 몇 가지 입력 필드를 사용하여 외부 DaaS 서비스를 호출하고 응답을 레코드 목록으로 변환합니다.

API는 POST 메서드를 사용합니다.

요청 URL

DaaS 검색 URL의 형식은 다음과 같습니다.

```
http://<host>:<port>/<context>/<database ID>/daas/search/<daas provider>
```

DaaS 검색 URL에 대한 다음 HTTP POST 요청을 만드십시오.

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

참고: 요청 본문에 비즈니스 항목의 세부 정보와 필수 필드를 제공합니다.

요청 본문

요청 본문에는 urn:co-ors.informatica.mdm 네임스페이스에 있는 Organization 또는 OrganizationView 유형의 XML 요소 또는 JSON 요소가 포함되어야 합니다.

샘플 API 요청

다음 예제는 표시 이름이 INFA인 organization 비즈니스 항목을 검색하는 DaaS 공급자 ondemand에 대한 요청입니다.

```
POST http://localhost:8080/cmx/cs/localhost-orcl-MDM_SAMPLE/daas/search/ondemand
{
  "Organization": {
    "displayname": "INFA"
  }
}
```

샘플 API 응답

다음 샘플 응답은 표시 이름이 INFA인 Organization 비즈니스 항목에 대해 DaaS 공급자가 반환하는 검색 결과를 보여 줍니다.


```
{
  "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 읽기

DaaS 읽기 REST API는 비즈니스 항목의 몇 가지 필드를 사용하여 외부 DaaS 서비스를 요청하고, 응답을 전체 레코드로 변환합니다.

API는 POST 메서드를 사용합니다. DaaS 공급자에 대한 요청에 필수 필드를 지정합니다.

요청 URL

DaaS 읽기 URL의 형식은 다음과 같습니다.

```
http://<host>:<port>/<context>/<database ID>/daas/read/<daas provider>
```

URL에 대한 다음 POST 요청을 만드십시오.

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

참고: 요청 본문에 레코드의 세부 정보와 필수 필드를 제공합니다.

쿼리 매개 변수

다음 테이블에는 사용할 수 있는 쿼리 매개 변수가 나열되어 있습니다.

매개 변수	설명
logChanges	선택 사항입니다. true로 설정할 경우, 결과 레코드에는 쓰기 비즈니스 항목 서비스에 전달되는 SDO(서비스 데이터 개체) 변경 요약이 포함됩니다. 기본값은 false입니다.

요청 본문

요청 본문에는 urn:coors.informatica.mdm 네임스페이스에 있는 OrganizationView 유형의 XML 요소 또는 JSON 요소가 포함되어야 합니다.

샘플 API 요청

다음 예제는 표시 이름이 INFA인 organization 비즈니스 항목을 검색하는 DaaS 공급자 ondemand에 대한 요청입니다.

```
POST http://localhost:8080/cmx/cs/localhost-orcl-MDM_SAMPLE/daas/search/ondemand
{
  "Organization": {
    "displayname": "INFA"
  }
}
```

샘플 API 응답

다음 샘플 응답은 D-U-N-S 번호가 001352574인 조직에 대해 DaaS 공급자가 반환한 세부 정보를 보여 줍니다.

```
{
  "Organization": {
    "displayName": "Infa Lab Inc",
    "dunsNumber": "001352574",
    "TelephoneNumbers": {
      "link": [],
      "item": [
        {
          "phoneNum": "09736250550"
        }
      ]
    },
    "Addresses": {
      "link": [],
      "item": [
        {
          "Address": {
            "cityName": "Rockaway",
            "addressLine1": "11 WALL ST"
          }
        }
      ]
    }
  }
}
```

WriteMerge

WriteMerge REST API는 DaaS 공급자에서 검색된 레코드 목록을 허용하고, 적절한 소스 시스템을 사용하여 개별 XREF에 유지한 다음 단일 레코드로 병합합니다. 모든 XREF는 동일한 레코드에 속합니다.

API는 POST 메서드를 사용합니다.

요청 URL

WriteMerge URL의 형식은 다음과 같습니다.

```
http://<host>:<port>/<context>/<database ID>/daas/write-merge/<business entity name>
```

WriteMerge URL에 대한 다음 HTTP POST 요청을 만드십시오.

```
POST http://<host>:<port>/<context>/<database ID>/daas/write-merge/<business entity name>
```

요청 본문

요청 본문에는 urn:cobase.informatica.mdm 네임스페이스에 있는 DaaSEntity.Pager 유형의 XML 또는 JSON 요소가 포함되어야 합니다.

응답 헤더

응답이 성공적이면 API는 응답 헤더에 interactionId 및 processId를 반환합니다.

샘플 API 요청

다음 샘플 요청은 DaaS 검색의 결과를 입력으로 사용하여 Organization 비즈니스 항목 유형의 레코드를 생성합니다.

```
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"
    }
  ]
}
```

샘플 API 응답

다음 샘플 응답은 레코드 목록을 단일 레코드로 병합한 이후의 응답 헤더와 본문을 보여 줍니다.

```
{
  "Organization": {
    "key": {
      "rowid": "121921",
      "sourceKey": "140000000000"
    },
    "rowidObject": "121921",
    "TelephoneNumbers": {
      "link": [],
      "item": [
        {
          "key": {
            "rowid": "21721",
            "sourceKey": "140000001000"
          },
          "rowidObject": "21721"
        }
      ]
    }
  }
}
```

DaaS 가져오기

DaaS 가져오기 REST API는 XML 형식의 데이터를 허용하고 레코드로 변환합니다.

API는 POST 메서드를 사용합니다.

요청 URL

DaaS 데이터 가져오기 URL의 형식은 다음과 같습니다.

```
http://<host>:<port>/<context>/<database ID>/daas/import/FamilyTreeMemberOrganizationToOrgView
```

DaaS 데이터 가져오기 URL에 대한 다음 HTTP POST 요청을 만드십시오.

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

쿼리 매개 변수

소스 시스템의 이름은 필수 매개 변수입니다.

다음 테이블에는 요청에 사용할 수 있는 매개 변수가 나열되어 있습니다.

매개 변수	설명
systemName	필수 사항입니다. 데이터 변경을 수행하는 소스 시스템의 이름입니다.
interactionId	선택 사항입니다. 모든 변경 내용에 할당할 상호 작용 ID입니다. 일반적으로 Hub에서는 호출의 결과로 보류 중인 변경 내용을 생성할 때 ID를 생성합니다.
effectivePeriod	선택 사항입니다. 시작 날짜와 종료 날짜를 포함합니다. 레코드가 유효한 기간을 지정합니다. 시간 표시 막대가 사용되는 레코드에 대해 이러한 매개 변수를 제공하십시오.
validateOnly	선택 사항입니다. TRUE로 설정할 경우, 수정된 레코드에 유효성 검사 규칙만 적용되고 변경 내용이 지속되지 않습니다.
recordState	선택 사항입니다. 생성되거나 업데이트된 레코드의 Hub 상태입니다. 지원되는 레코드 상태는 ACTIVE 및 PENDING입니다.
processId	선택 사항입니다. 태스크가 포함된 워크플로우 프로세스의 ID입니다. 서비스 호출의 결과로 워크플로우가 시작된 경우, 시작된 워크플로우 프로세스의 식별자가 매개 변수에 포함됩니다.

관련 항목:

- "날짜 및 시간 형식(UTC)" 페이지 27

요청 본문

요청 본문에는 urn:co-ors.informatica.mdm 네임스페이스에 있는 DaaSChangeFamilyTreeMemberOrganizationToOrgView 유형의 XML 또는 JSON 요소가 포함되어야 합니다.

응답 헤더

응답이 성공한 경우 해당 API는 응답 헤더의 interactionId와 processId 및 응답 본문의 레코드 세부 정보를 반환합니다.

이 프로세스에서 상호 작용 ID를 생성한 후 이 ID를 사용하여 레코드를 생성하는 경우 해당 API는 상호 작용 ID를 반환합니다. 이 프로세스에서 데이터베이스에 레코드를 직접 저장하는 대신 워크플로우를 시작하는 경우 해당 API는 워크플로우 프로세스의 ID인 프로세스 ID를 반환합니다.

다음 예는 상호 작용 ID 및 프로세스 ID가 포함된 응답 헤더를 보여 줍니다.

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

샘플 API 요청

요청은 항상 XML 형식입니다.

다음 샘플 요청은 연결 네임스페이스에 있는 ChildAssociation 유형의 XML 데이터를 보여 줍니다.

```
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>
```

샘플 API 응답

다음 샘플 응답은 Organization 비즈니스 항목을 가져오고 생성한 이후의 응답 헤더와 본문을 보여 줍니다.

```
BES-interactionId: 72202100242034
BES-processId: 156048
{
  "Organization": {
    "key": {
      "rowid": "101921",
      "sourceKey": "697557825"
    },
    "rowidObject": "101921"
  }
}
```

DaaS 업데이트

DaaS 업데이트 REST API는 변경 전과 후에 해당하는 XML 형식의 데이터를 허용합니다. API는 변경 내용을 레코드에 적용합니다.

API는 POST 메서드를 사용합니다.

요청 URL

DaaS 업데이트 URL의 형식은 다음과 같습니다.

```
http://<host>:<port>/<context>/<database ID>/daas/update/FamilyTreeMemberOrganizationToOrgView
```

DaaS 업데이트 URL에 대한 다음 HTTP POST 요청을 만드십시오.

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

쿼리 매개 변수

소스 시스템의 이름은 필수 매개 변수입니다.

다음 테이블에는 요청에 사용할 수 있는 매개 변수가 나열되어 있습니다.

매개 변수	설명
systemName	필수 사항입니다. 데이터 변경을 수행하는 소스 시스템의 이름입니다.
interactionId	선택 사항입니다. 모든 변경 내용에 할당할 상호 작용 ID입니다. 일반적으로 Hub에서는 호출의 결과로 보류 중인 변경 내용을 생성할 때 ID를 생성합니다.
effectivePeriod	선택 사항입니다. 시작 날짜와 종료 날짜를 포함합니다. 레코드가 유효한 기간을 지정합니다. 시간 표시 막대가 사용되는 레코드에 대해 이러한 매개 변수를 제공하십시오.
validateOnly	선택 사항입니다. TRUE로 설정할 경우, 수정된 레코드에 유효성 검사 규칙만 적용되고 변경 내용이 지속되지 않습니다.
recordState	선택 사항입니다. 생성되거나 업데이트된 레코드의 Hub 상태입니다. 지원되는 레코드 상태는 ACTIVE 및 PENDING입니다.
processId	선택 사항입니다. 태스크가 포함된 워크플로우 프로세스의 ID입니다. 서비스 호출의 결과로 워크플로우가 시작된 경우, 시작된 워크플로우 프로세스의 식별자가 매개 변수에 포함됩니다.

관련 항목:

- "날짜 및 시간 형식(UTC)" 페이지 27

요청 본문

요청 본문에는 urn:co-ors.informatica.mdm 네임스페이스에 있는 DaaSChangeFamilyTreeMemberOrganizationToOrgView 유형의 XML 또는 JSON 요소가 포함되어야 합니다.

응답 헤더

응답이 성공한 경우 해당 API는 응답 헤더의 interactionId와 processId 및 응답 본문의 레코드 세부 정보를 반환합니다.

이 프로세스에서 상호 작용 ID를 생성한 후 이 ID를 사용하여 레코드를 생성하는 경우 해당 API는 상호 작용 ID를 반환합니다. 이 프로세스에서 데이터베이스에 레코드를 직접 저장하는 대신 워크플로우를 시작하는 경우 해당 API는 워크플로우 프로세스의 ID인 프로세스 ID를 반환합니다.

다음 예는 상호 작용 ID 및 프로세스 ID가 포함된 응답 헤더를 보여 줍니다.

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

샘플 API 요청

API는 XML 형식으로 두 가지 응답, 즉 변경 전의 응답과 변경 후의 응답을 허용합니다. 다음 요청에서 조직에 새 전화 번호가 추가됩니다. before XML 데이터에는 전화 번호가 없고 after XML 데이터에는 전화 번호가 있습니다.

다음 요청에는 새로 추가된 전화 번호가 포함되어 있습니다.

```
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>
```

샘플 API 응답

다음 예제는 조직의 새로 생성된 전화 번호의 rowId를 보여 줍니다.

```
{
  "Organization": {
    "key": {
      "rowid": "101921",
      "sourceKey": "697557825"
    },
    "rowidObject": "101921",
    "TelephoneNumbers": {
      "link": [],
      "item": [
        {
          "key": {
            "rowid": "1722",
            "sourceKey": "09736250550"
          },
          "rowidObject": "1722"
        }
      ]
    }
  }
}
```

제 4 장

Simple Object Access Protocol 비즈니스 항목 서비스 호출

이 장에 포함된 항목:

비즈니스 항목 서비스에 대한 Simple Object Access Protocol 호출

SOAP(Simple Object Access Protocol) 끝점 호출은 모든 비즈니스 항목 서비스를 웹 서비스로 사용 가능하게 만듭니다. 비즈니스 항목에서 레코드를 작성, 삭제, 업데이트 및 검색하기 위해 SOAP 호출을 만들 수 있습니다. 레코드 병합, 병합 해제 및 일치 등의 작업을 수행할 수 있습니다. 또한 SOAP 호출을 만들어 태스크를 작성, 업데이트 및 검색하고 태스크를 수행할 수 있습니다.

비즈니스 항목 서비스에 대한 SOAP 끝점은 웹 서비스 보안(WS-Security) UsernameToken을 사용하여 사용자를 인증합니다.

참고: SOAP API를 사용하여 비즈니스 항목 서비스를 호출하기 전에 연산 참조 저장소의 유효성을 검사하십시오.

인증 방법

비즈니스 항목 서비스에 대한 모든 SOAP 호출에는 사용자 인증이 필요합니다. 웹 서비스 요청의 SOAP 메시지 헤더에서 사용자 이름 및 암호를 제공하십시오.

SOAP 헤더 요소 보안에는 보안 관련 데이터가 포함되어 있습니다. 보안 요소에는 다음 하위 요소가 있는 UsernameToken 요소가 포함되어 있습니다.

사용자 이름

토큰과 연결된 사용자 이름입니다.

암호

토큰과 연결된 사용자 이름에 대한 암호입니다.

UsernameToken 요소의 사용자 이름 및 암호를 보내십시오.

다음 예는 SOAP 메시지의 보안 헤더 요소를 보여 줍니다.

```
<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>
```

타사 응용 프로그램에서 로그인하기 위한 인증 쿠키

인증 쿠키를 사용하여 타사 응용 프로그램에서 MDM Hub 사용자를 인증하고 비즈니스 항목 서비스를 호출할 수 있습니다. 인증된 사용자의 자격 증명에 기반하여 쿠키를 가져올 수 있습니다. 쿠키를 저장하여 SOAP API를 호출하는 데 사용합니다. 사용자 이름과 암호는 하드 코딩하지 않아도 됩니다.

사용자 이름과 암호를 사용하여 Entity 360 View에 로그인하기 위해 다음과 같은 POST 요청을 만드십시오.

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

로그인 작업이 성공하면 서버가 set-cookie 헤더 필드에 인증 쿠키를 반환합니다. 다음 샘플 코드는 응답 헤더의 set-cookie를 보여 줍니다.

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

해시를 저장하여 API 호출의 요청 헤더에 사용합니다. API 호출 시 사용자 이름과 암호를 제공하지 않아도 됩니다.

다음 예에서는 API 요청 헤더에 인증 쿠키를 사용하는 방법을 보여 줍니다.

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

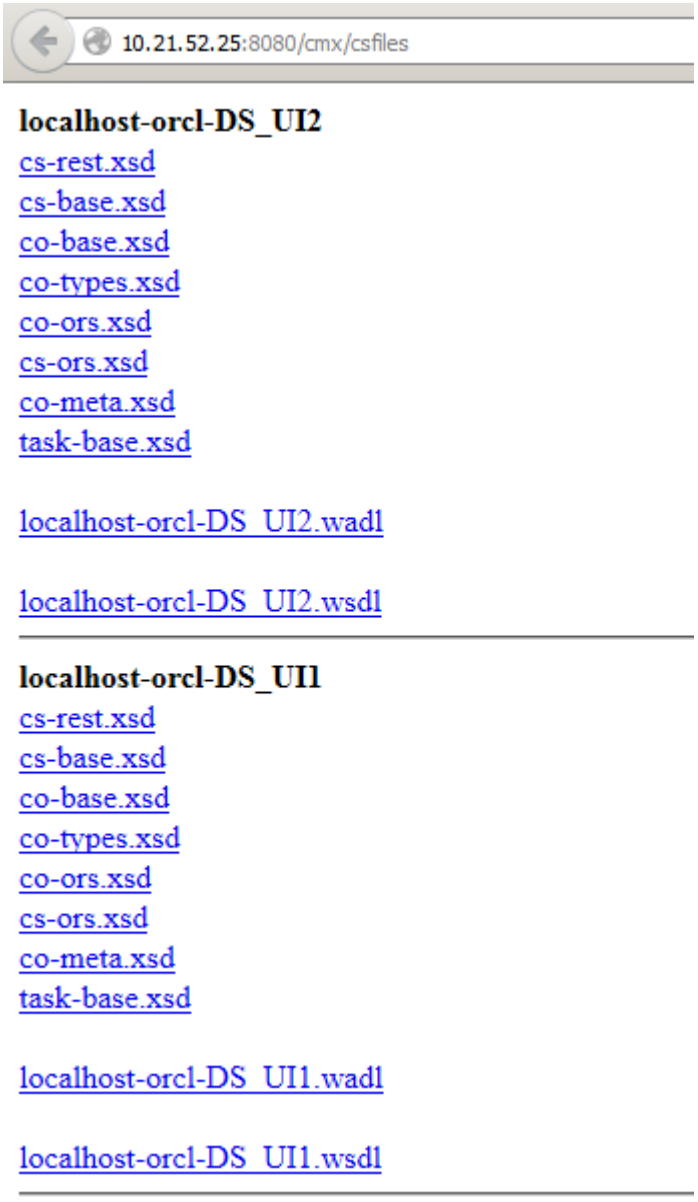
WSDL(Web Services Description Language) 파일

WSDL(Web Services Description Language) 파일에는 웹 서비스의 XML 설명, SOAP 요청과 응답의 형식 및 모든 매개 변수가 포함되어 있습니다. MDM Hub는 각 연산 참조 저장소에 대한 WSDL 파일을 생성합니다.

각 연산 참조 저장소에 대한 WSDL 파일은 다음 위치에 있습니다.

```
http://<호스트>:<포트>/cmx/csfiles
```

다음 이미지는 연산 참조 저장소에 대한 WSDL 파일을 다운로드할 수 있는 위치를 보여 줍니다.



해당 링크를 클릭하여 DS_UI1 또는 DS_UI2 연산 참조 저장소에 대한 WSDL 파일을 다운로드하십시오.

SOAP URL

SOAP URL은 SOAP 서버에 연결하는 데 사용하는 주소입니다.

SOAP URL에는 다음 구문이 있습니다.

```
http://<호스트>:<포트>/<컨텍스트>/<데이터베이스 ID>
```

해당 URL에는 다음 필드가 있습니다.

호스트

데이터베이스를 실행하는 호스트입니다.

포트

데이터베이스 수신기에서 사용하는 포트 번호입니다.

컨텍스트

컨텍스트는 항상 cmx/services/BEServices입니다.

데이터베이스 ID

Hub 콘솔에서 데이터베이스 도구에 등록된 ORS의 ID입니다.

다음 예는 SOAP URL을 보여 줍니다.

```
http://localhost:8080/cmx/services/BEServices/localhost-orcl-DS_UI1
```

SOAP 요청 및 응답

SOAP XML 메시지 형식을 사용하여 SOAP 클라이언트를 통해 비즈니스 항목 서비스에 요청을 보내고 비즈니스 항목 서비스에서 클라이언트로 응답을 받을 수 있습니다. SOAP 요청 및 응답 형식은 동일합니다.

SOAP 메시지에는 다음 요소가 포함되어 있습니다.

엔벨로프

메시지의 시작과 끝을 정의합니다.

헤더

선택 사항입니다. 메시지 처리를 위한 인증 세부 정보 등 추가적인 특성을 포함합니다. 머리글 요소가 있는 경우 이는 엔벨로프 요소의 첫 번째 하위 요소여야 합니다.

본문

클라이언트 또는 웹 서비스가 처리하는 XML 데이터를 포함합니다.

SOAP 메시지의 형식은 다음과 같습니다.

```
<?xml version="1.0"?>
<env:Envelope  xmlns:env="http://schemas.xmlsoap.org/soap/envelope/" >

<env:Header>
</env:Header>

<env:Body>
</env:Body>

</env:Envelope>
```

SOAP 요청의 형식은 다음과 같습니다.

```
POST /<host>:<port>/<context>/<database ID> HTTP/1.0
Content-Type: text/xml; charset=utf-8

<?xml version="1.0"?>
<env:Envelope  xmlns:env="http://schemas.xmlsoap.org/soap/envelope/" >

<env:Header>
</env:Header>

<env:Body>
</env:Body>

</env:Envelope>
```

SOAP 응답의 형식은 다음과 같습니다.

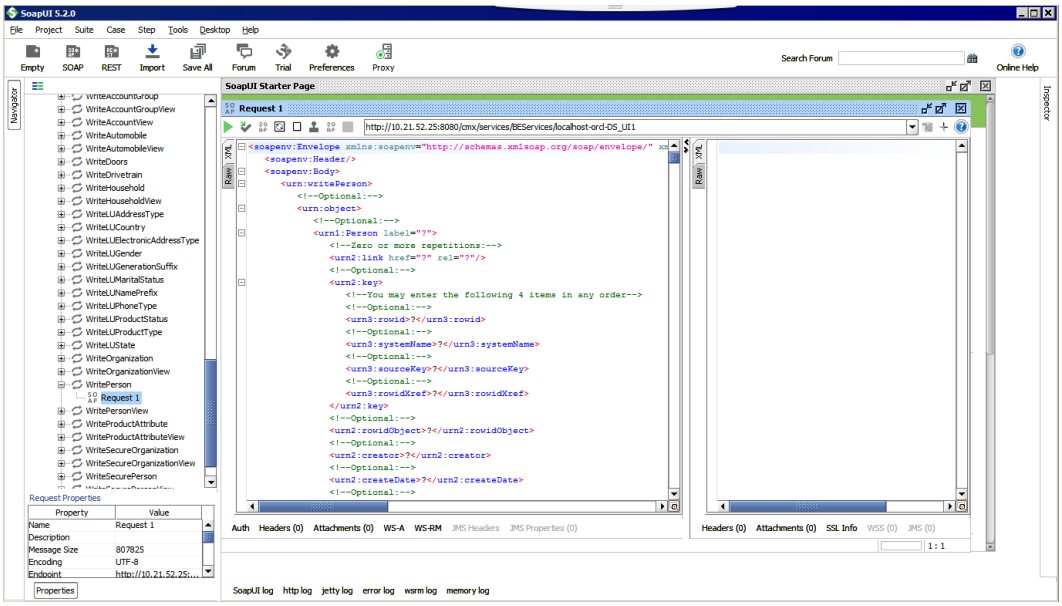
```
HTTP/1.0 200 OK
Content-Type: text/xml; charset=utf-8

<?xml version="1.0"?>
<env:Envelope  xmlns:env="http://schemas.xmlsoap.org/soap/envelope/" >

<env:Header>
</env:Header>

<env:Body>
</env:Body>

</env:Envelope>
```

입력 및 출력 매개 변수 보기

SoapUI와 같은 기능 테스트 도구를 사용하여 SOAP API 입력 및 출력 매개 변수를 볼 수 있습니다.

SOAP 프로젝트를 작성하고 WSDL 파일을 이 프로젝트에 가져오십시오. 비즈니스 항목 서비스를 사용하여 수행할 수 있는 작업이 SoapUI에 노드로 나타납니다. 각 작업에는 요청 및 응답 메시지 형식이 있습니다. WSDL 파일을 가져오면 SoapUI가 각 작업에 대한 샘플 요청을 작성합니다.

프로젝트를 열고 요청을 두 번 클릭하여 요청 편집기를 여십시오. 다음 이미지는 WritePerson SOAP API에 대한 SoapUI의 입력 매개 변수를 보여 줍니다.



SOAP API 참조

비즈니스 항목 서비스에 대한 SOAP API 참조는 SOAP API를 나열하고 각 API에 대한 설명을 제공합니다. 비즈니스 항목 서비스에 대한 설명은 WSDL 파일을 참조하십시오.

다음 SOAP API를 사용하여 비즈니스 항목에 대해 작업을 수행하십시오.

메타데이터 가져오기

비즈니스 항목의 데이터 구조를 반환합니다.

레코드 나열

조회 값 또는 외래 키 값의 목록을 반환합니다.

레코드 읽기

비즈니스 항목의 루트 레코드에 대한 세부 정보를 반환합니다.

레코드 생성

지정된 비즈니스 항목에 레코드를 작성합니다.

레코드 업데이트

지정된 루트 레코드와 그 하위 레코드를 업데이트합니다.

레코드 삭제

비즈니스 항목의 루트 레코드를 삭제합니다.

레코드 검색

검색 가능한 루트 비즈니스 항목에서 문자열 값을 검색하고 검색 조건과 일치하는 루트 레코드를 반환합니다.

승격 미리 보기

변경 요청의 상호 작용 ID에 따라 보류 중인 변경 내용을 승격시키는 경우 결과 레코드의 미리 보기를 반환합니다.

승격

변경 요청의 상호 작용 ID에 따라 레코드에 대해 보류 중인 모든 변경 내용을 승격시킵니다.

승격 삭제

변경 요청의 상호 작용 ID에 따라 레코드에 대해 보류 중인 모든 변경 내용을 삭제합니다.

병합 미리 보기

두 개 이상의 루트 레코드를 병합하는 경우 통합된 루트 레코드의 미리 보기를 반환합니다.

레코드 병합

두 개 이상의 루트 레코드를 병합하여 하나의 통합된 레코드를 작성합니다.

레코드 병합 해제

레코드가 병합되기 전에 존재한 개별 루트 레코드로 루트 레코드를 병합 해제합니다.

관련 레코드 가져오기

사용자가 계층 관리자에서 구성하는 관계를 기반으로 관련 레코드의 목록을 반환합니다.

일치된 레코드 읽기

지정된 루트 레코드와 일치하는 레코드를 반환합니다.

일치된 레코드 업데이트

일치 테이블에서 레코드를 작성하거나 업데이트합니다.

일치된 레코드 삭제

일치 테이블에서 일치된 레코드를 삭제합니다.

BPM **메타데이터 가져오기**

워크플로우 어댑터가 태스크 연계 가져오기 서비스 및 관리 서비스를 수행할 수 있는지 여부를 지정하는 두 개의 표시기 및 태스크 유형을 반환합니다.

태스크 나열

워크플로우 태스크의 목록을 반환합니다.

태스크 읽기

태스크의 세부 정보를 반환합니다.

태스크 작성

태스크를 작성하고 워크플로우를 시작합니다.

태스크 업데이트

단일 태스크를 업데이트합니다.

태스크 작업 실행

태스크 작업을 수행하고 향후 처리를 위해 워크플로우로 태스크를 다시 설정합니다.

할당 가능한 사용자 나열

태스크 유형에 속하는 태스크를 할당할 수 있는 사용자의 목록을 반환합니다.

태스크 완료

워크플로우의 모든 태스크를 완료한 후 태스크 워크플로우를 닫습니다.

샘플 SOAP 요청 및 응답

다음 샘플 SOAP 요청은 할당 가능한 사용자의 목록을 검색합니다.

```
<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>
```

다음 샘플 SOAP 응답은 할당 가능한 사용자를 나열합니다.

```
<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>
```

제 5 장

교차 참조 레코드 및 BVT 계산을 위한 서비스

이 장에 포함된 항목:

교차 참조 레코드 및 BVT 계산을 위한 서비스 개요

교차 참조 레코드 및 BVT(최선의 진실, Best Version of the Truth) 계산을 위한 서비스를 사용하여 소스 데이터가 마스터 레코드를 구성하는 방법을 배울 수 있습니다.

이러한 서비스를 사용하여 다음과 같은 태스크를 수행할 수 있습니다.

- 소스 데이터에 대한 정보 수집
- BVT(최선의 진실, Best Version of the Truth)가 확인된 방법 확인
- 마스터 레코드에 BVT(최선의 진실, Best Version of the Truth)가 포함되도록 BVT 계산 재정의

교차 참조 데이터 가져오기 및 BVT 계산 조사

MDM Hub의 마스터 레코드는 BVT(최선의 진실, Best Version of the Truth)를 유지 관리합니다. MDM Hub는 여러 소스 시스템에서 가장 신뢰할 수 있는 데이터를 각 마스터 레코드에 통합하여 BVT(최선의 진실, Best Version of the Truth)를 얻습니다. MDM Hub는 교차 참조 레코드에 소스 데이터를 저장합니다. 비즈니스 항목 서비스를 사용하여 교차 참조 레코드의 데이터를 읽고 BVT가 계산된 방법을 확인할 수 있습니다.

교차 참조 레코드 가져오기

비즈니스 항목 서비스를 사용하여 특정 마스터 레코드에 대한 교차 참조 레코드를 가져올 수 있습니다.

교차 참조 레코드를 가져오는 REST API URL의 형식은 다음과 같습니다.

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<row ID>?contentMetadata=XREF
```

다음 샘플 요청은 행 ID가 123인 Person 비즈니스 항목 레코드에 대한 교차 참조 레코드를 검색합니다.

```
GET http://localhost:8080/cmx/cs/localhost-orcl-ORS/Person/123?contentMetadata=XREF
```

.

교차 참조 레코드 가져오기 응답

다음 예제는 행 ID가 123인 Person 레코드에 대해 반환되는 교차 참조 레코드를 보여 줍니다.

```
GET /Person/123?contentMetadata=XREF

{
  "firstName": "Joe",
  "lastName": "Smith",
  "XREF": {
    "item": [
      {
        "rowidXref": 111,
        "firstName": "Joe",
        "lastName": "Smith",
      },
      {
        "rowidXref": 222,
        "firstName": "John",
        "lastName": "Smith"
      }
    ]
  }
}
```

마스터 레코드의 제공자 확인

비즈니스 항목 서비스를 사용하여 마스터 레코드에 제공되는 교차 참조 레코드 필드를 확인할 수 있습니다. 각 필드에 대한 관련 레코드는 교차 참조 레코드의 행 ID로 식별됩니다.

마스터 레코드의 제공자를 확인하는 REST API URL의 형식은 다음과 같습니다.

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<row ID>?contentMetadata=BVT
```

다음 샘플 요청은 행 ID가 123인 Person 레코드에 대한 BVT 정보를 검색합니다.

```
GET http://localhost:8080/cmx/cs/localhost-orcl-ORS/Person/123?contentMetadata=BVT
```

마스터 레코드 응답의 제공자 확인

다음 예는 마스터 레코드의 각 필드에 제공되는 교차 참조 레코드를 보여 줍니다.

```
{
  "firstName": "Joe",
  "lastName": "Smith",
  "BVT": {
    "firstName": {
      "rowidXref": 111
    },
    "lastName": {
      "rowidXref": 222
    }
  },
}
```

관련 교차 참조 레코드 필드의 트러스트 점수 가져오기

비즈니스 항목 서비스를 사용하여 마스터 레코드에 제공되는 교차 참조 레코드 필드의 트러스트 점수를 가져올 수 있습니다.

제공자를 확인하고 트러스트 점수를 가져오는 REST API URL의 형식은 다음과 같습니다.

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<row ID>?contentMetadata=TRUST
```

다음 샘플 요청은 행 ID가 123인 Person 레코드에 대한 트러스트 점수를 제공합니다.

```
GET http://localhost:8080/cmx/cs/ors/Person/123?contentMetadata=TRUST
```

관련 교차 참조 레코드 필드의 트러스트 점수 가져오기 응답

다음 응답 예제는 Person 비즈니스 항목 레코드의 각 필드에 대한 트러스트 점수를 제공합니다.

```
{
  "firstName": "John",
  "lastName": "Smith",
  "TRUST": {
    "firstName": {
      "score": 75.0,
      "valid": true,
      "trustSetting" :{
          // custom settings
      }
    },
  },
}
```

모든 교차 참조 레코드 필드의 트러스트 점수 가져오기

contentMetadata 매개 변수가 XREF_TRUST로 설정된 REST API를 사용하여 모든 교차 참조 레코드 필드의 트러스트 점수와 다운그레이드 백분율을 가져옵니다.

제공자를 확인하고 트러스트 점수를 가져오는 읽기 REST API의 요청 URL:

```
http://<host>:<port>/<context>/<database ID>/<business entity>/<row ID>?contentMetadata=XREF_TRUST
```

다음 샘플 요청은 행 ID가 123인 Person 레코드에 대한 교차 참조 데이터를 검색합니다.

```
GET http://localhost:8080/cmx/cs/localhost-orcl-ORS/Person/123?contentMetadata=XREF_TRUST
```

모든 교차 참조 레코드 필드의 트러스트 점수 가져오기 응답

다음 예제는 Person 비즈니스 항목에 대한 모든 교차 참조 레코드 필드의 트러스트 점수 및 다운그레이드 백분율을 보여 줍니다.

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

소스 시스템에 대한 정보 가져오기

교차 참조 데이터를 어느 소스 시스템에서 가져왔는지에 대한 정보 및 전체 레코드, 각 노드 또는 각 레코드에 소스 시스템이 제공하는 교차 참조 레코드의 수에 대한 정보를 가져올 수 있습니다.

요청에 다음 매개 변수를 지정할 수 있습니다.

describe

소스 시스템에 대한 설명을 반환하려면 true로 설정합니다. true 또는 false일 수 있습니다. 기본값은 false입니다.

aggregate

소스 시스템 정보를 반환할 수준을 정의합니다. ENTITY, NODE 및 RECORD일 수 있습니다. 기본값은 ENTITY입니다.

recordStates

레코드를 반환할 레코드 상태를 지정합니다. ACTIVE, PENDING 또는 DELETED일 수 있습니다. 기본값은 ACTIVE입니다.

compact

no로 설정하면 aggregate 매개 변수를 ENTITY 및 기타 집계 수준으로 지정한 경우에 항목 수준 데이터가 반환됨을 지정합니다. yes 또는 no일 수 있습니다. REST API 요청에만 사용할 수 있습니다. 기본값은 yes입니다.

소스 시스템에 대한 정보 가져오기 예제

다음 샘플 요청은 행 ID가 123인 Person 비즈니스 항목에 대해 항목 수준 및 노드 수준 소스 시스템 정보를 가져옵니다.

```
GET http://localhost:8080/cmx/cs/ors/Person/123?action=getSourceSystems&aggregate=ENTITY,NODE&compact=no
```

다음 샘플 요청은 행 ID가 456인 Person 비즈니스 항목에 대해 레코드 수준 소스 시스템 정보와 소스 시스템 설명을 가져옵니다.

```
GET http://localhost:8080/cmx/cs/ors/Person/123/Address/456?
action=getSourceSystems&aggregate=ENTITY,NODE&compact=no
```

소스 시스템에 대한 정보 가져오기 응답

다음 예제는 Person 비즈니스 항목에 대한 항목 수준 정보와 노드 수준 정보를 보여 줍니다.

```
{
      "name": "Admin",
      "xrefCount": 120
    }
  ],
  Person: {
    "rowidObject": "456",
    "sourceSystem":
      {
        "name": "Admin",
        "xrefCount": 30
      }
    ]
  }
}
```

응답 필터링 및 페이지 번호 지정

응답에 반환할 필드를 선택하고, 몇 가지 조건에 따라 결과를 필터링하고, 결과에 페이지 번호를 지정할 수 있습니다.

필터링 요청 예제

다음 테이블에는 다양한 필터가 적용된 Person 비즈니스 항목에 대한 샘플 요청 및 응답에 반환된 결과에 대한 설명이 나와 있습니다.

요청	반환된 결과에 대한 설명
/Person/123	모든 사용자 정의 필드
/Person/123?readSystemFields=true	모든 사용자 정의 필드 및 모든 시스템 필드
/Person/123?fields=firstName	사용자 정의 필드 한 개
/Person/123?fields=updatedBy	시스템 필드 한 개
Person/123?fields=firstName,updatedBy	사용자 정의 필드 한 개와 시스템 필드 한 개
/Person/123?fields=firstName&readsystemFields=true	사용자 정의 필드 한 개와 모든 시스템 필드

BVT(최선의 진실, Best Version of the Truth) 설정

데이터 스튜어드는 교차 참조 레코드의 소스 데이터를 조사한 후 마스터 레코드가 BVT(최선의 진실, Best Version of the Truth)를 나타내도록 소스 데이터가 통합되는 방법을 조정할 수 있습니다.

비즈니스 항목 서비스를 사용하면 다음과 같은 작업을 수행하여 BVT(최선의 진실, Best Version of the Truth)를 설정할 수 있습니다.

- 트러스트 설정 업데이트
- 일치하지 않는 소스 데이터 제거
- 올바른 관련 필드 선택
- 마스터 레코드에 올바른 값 기록

올바른 관련 필드 선택

트러스트 점수가 가장 높은 필드에 BVT(최선의 진실, Best Version of the Truth)가 포함되어 있지 않은 경우, 데이터 스튜어드가 올바른 데이터가 포함되어 있는 필드를 선택하여 해당 데이터를 마스터 레코드에 제공할 수 있습니다.

시스템 이름과 소스 키에 기반하여 올바른 관련 필드를 선택하는 URL 및 요청 본문의 형식은 다음과 같습니다.

```
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>"
        }
    }
}
```

교차 참조 레코드 ID에 기반하여 올바른 관련 필드를 선택하는 URL 및 요청 본문의 형식은 다음과 같습니다.

```
POST http://<host>:<port>/<context>/<database ID>/<business entity>/<row ID>?systemName=<source system
name>
{
    BVT: {
        <field name>: {
            rowidXref: "<row ID>"
        }
    }
}
```

올바른 관련 필드 선택 예제

다음 URL 및 요청 본문은 Sales 소스 시스템에서 소스 키가 0001인 교차 참조 레코드의 이름 필드를 마스터 레코드에 제공할 대상으로 선택합니다.

```
POST http://localhost:8080/cmx/cs/localhost-orcl-ORS/Person/123?systemName=Admin
{
    BVT: {
        firstName: {
            systemName: "Sales",
            sourceKey: "0001"
        }
    }
}
```

다음 URL 및 요청 본문은 행 ID가 789인 교차 참조 레코드의 이름 필드를 마스터 레코드에 제공할 대상으로 선택합니다.

```
POST http://localhost:8080/cmx/cs/localhost-orcl-ORS/Person/123?systemName=Admin
{
    BVT: {
        firstName: {
            rowidXref: "789"
        }
    }
}
```

마스터 레코드에 올바른 값 기록

마스터 레코드에 올바른 값을 기록하는 비즈니스 항목 서비스 호출을 사용할 때는 값에 대한 트러스트 설정도 지정할 수 있습니다. 트러스트 설정을 지정하지 않을 경우 MDM Hub에서는 관리자 시스템 설정을 사용합니다.

관리자 트러스트 설정을 사용하여 올바른 값을 기록하는 URL 및 요청 본문의 형식은 다음과 같습니다.

```
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
            }
        }
    }
}
```

정의된 트러스트 설정을 사용하여 올바른 값을 기록하는 URL 및 요청 본문의 형식은 다음과 같습니다.

```
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>"
            }
        }
    }
}
```

트러스트 매개 변수

다음과 같은 트러스트 매개 변수를 정의할 수 있습니다.

minimumTrust

데이터 값이 구식이 되었을 때(붕괴 기간이 경과한 후) 데이터 값이 가지는 트러스트 수준입니다. 이 값은 최대 트러스트보다 작거나 같아야 합니다.

참고: 최대 트러스트와 최소 트러스트가 같은 경우에는 붕괴 곡선이 수평선으로 나타나고 붕괴 기간과 붕괴 유형이 적용되지 않습니다.

maximumTrust

데이터 값이 변경된 직후 데이터 값이 가지는 트러스트 수준입니다. 예를 들어 소스 시스템 X에서 전화 번호 필드를 555-1234에서 555-4321로 변경할 경우, 새 값에는 시스템 X에서 전화 번호 필드에 부여하는 최대 트러스트 수준이 적용됩니다. 최대 트러스트 수준을 상대적으로 높게 설정하면 소스 시스템의 변경 내용을 기본 개체에 적용할 수 있습니다.

timeUnit

붕괴 기간을 계산하는 데 사용되는 단위(일, 주, 개월, 분기 또는 년)를 지정합니다.

maximumTimeUnits

붕괴 기간을 계산하는 데 사용되는 시간 단위(일, 주, 개월, 분기 또는 년)의 수를 지정합니다.

graphType

붕괴 기간 중 트러스트 수준이 감소하는 방식을 나타내는 붕괴 패턴입니다. 그래프 유형은 다음과 같은 붕괴 패턴 중 하나일 수 있습니다.

그래프 유형 매개 변수	설명
LINEAR	가장 단순한 붕괴 패턴입니다. 붕괴 패턴이 최대 트러스트에서 최소 트러스트를 잇는 직선 형태로 나타납니다.
RISL	트러스트가 붕괴 기간의 초반에 집중적으로 감소합니다. 붕괴 패턴은 오목한 곡선 형태로 나타납니다. 소스 시스템이 이 그래프 유형인 경우, 해당 시스템에서 가져오는 새 값은 트러스트되지만 재정의될 수 있습니다.
SIRL	트러스트가 붕괴 기간의 후반에 집중적으로 감소합니다. 붕괴 패턴은 볼록한 곡선 형태로 나타납니다. 소스 시스템이 이 그래프 유형인 경우, 마스터 레코드의 값은 붕괴 기간의 거의 끝에 도달하기 전까지는 다른 시스템에 의해 재정의될 가능성이 비교적 낮습니다.

마스터 레코드에 올바른 값 기록 예제

예제 1

마스터 레코드에서 Sam Brown이라는 이름을 John Smith로 변경하려고 합니다. 변경은 Sales 소스 시스템에서 기인합니다. 관리자 트러스트 설정이 트러스트 설정에 적용됩니다.

다음 코드는 예제 1의 URL 및 명령을 보여 줍니다.

```
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
```

```
            }
        }
    }
}
```

예제 2

마스터 레코드에서 Sam Brown이라는 이름을 John Smith로 변경하려고 합니다. 변경은 SFA 소스 시스템에서 기인합니다. 트러스트 설정은 최소 트러스트 60 및 최대 트러스트 90으로 설정되고, 3개월의 붕괴 기간 동안 트러스트가 선형으로 붕괴됩니다.

다음 코드는 예제 2의 URL 및 명령을 보여 줍니다.

```
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"
            }
        }
    }
}
```

일치하지 않는 소스 데이터 제거

교차 참조 레코드가 특정 마스터 레코드와 잘못 연결된 경우에는 데이터 스튜어드가 교차 참조 레코드를 병합 해제할 수 있습니다. 병합 해제된 교차 참조 레코드에서 새로운 마스터 레코드가 생성됩니다.

병합 해제 호출 시 교차 참조 레코드 하나만 병합 해제할 수 있습니다. 병합 해제해야 할 교차 참조 레코드가 여러 개 있는 경우에는 교차 참조 레코드 각각에 대해 병합 해제 호출을 실행합니다.

병합 해제 이벤트에 대해 트리거가 구성되어 있으면 병합 해제 태스크가 생성됩니다. 그렇지 않으면 교차 참조 레코드가 병합 해제됩니다.

시스템 이름 및 소스 키에 기반하여 레코드를 병합 해제하는 URL 및 명령의 형식은 다음과 같습니다.

```
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>" }
}
```

교차 참조 레코드 ID에 기반하여 레코드를 병합 해제하는 URL 및 명령의 형식은 다음과 같습니다.

```
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>"}
}
```

일치하지 않는 소스 데이터 제거 예제

REST API 예제

다음 코드는 주소 레코드에서 하위 수준에 있는 교차 참조 레코드를 병합 해제하는 URL 및 명령을 보여 줍니다.

```
POST http://localhost:8080/cmx/cs/localhost-orcl-MDM_SAMPLE/Person/181921?
action=unmerge&systemName=Admin
{
"name":"Person.Address",
"key":{
"rowid":"41721 ",
"rowidXref":41722
}
}
```

설명:

- 병합 해제할 교차 참조 레코드의 행 ID는 41722입니다.
- 교차 참조 레코드를 병합 해제할 마스터 레코드의 행 ID는 41721입니다.
- 루트 레코드의 행 ID는 181921입니다.

SOAP/EJB 예제

다음 코드는 주소 레코드에서 하위 수준에 있는 교차 참조 레코드를 병합 해제하는 URL 및 명령을 보여 줍니다.

```
<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>
```

설명:

- 병합 해제할 교차 참조 레코드의 행 ID는 41722입니다.
- 교차 참조 레코드를 병합 해제할 마스터 레코드의 행 ID는 41721입니다.
- 루트 레코드의 행 ID는 181921입니다.

병합 해제 응답

병합 해제 응답에는 병합 해제된 교차 참조 레코드에서 생성된 기본 개체의 행 ID가 포함됩니다.

응답 예제 1

다음 예제는 Person 루트 노드에서 레코드를 교차 참조할 경우의 응답을 보여 줍니다.

```
{
    Person: {
        rowidObject: "7777"
    }
}
```

응답 예제 2

다음 예제는 Address 하위 노드에서 레코드를 교차 참조할 경우의 응답을 보여 줍니다.

```
{
    Person: {
        Address: {
            item: [
                rowidObject: "55555"
            ]
        }
    }
}
```

제 6 장

기업 연결 서비스 지원

이 장에 포함된 항목:

개요

D&B(Duns & Bradstreet)의 기업 연결 서비스는 요청된 조직의 상위 항목 및 모든 관련 항목을 반환합니다. D&B의 연결 서비스를 사용하면 조직의 모든 지점 및 사업부에 대한 정보를 가져올 수 있습니다. 연결 서비스의 데이터를 사용하여 레코드를 생성하고 업데이트할 수 있습니다.

기업 연결 데이터를 MDM Hub에 가져올 수 있습니다. 항목 보기에서 DaaS 공급자 사용자 지정 구성 요소의 연결 서비스를 사용할 수 있는 사용자 지정 응용 프로그램을 개발해야 합니다.

D&B 서비스에서 데이터를 가져오고 해당 데이터를 사용하여 레코드를 생성하려면 비즈니스 항목 서비스가 필요합니다. 외부 저장소에 있는 데이터가 변경된 경우에 해당하는 변경 내용을 레코드에 적용할 수 있어야 합니다. D&B는 데이터의 변경 내용을 알려 주는 모니터링 서비스를 제공합니다. 변경 전과 후의 데이터를 허용하고, 변경 내용을 해당하는 레코드에 적용할 수 있는 서비스가 필요합니다.

DaaS 가져오기 및 업데이트를 위한 비즈니스 항목 서비스

DaaS 가져오기 비즈니스 항목 서비스는 연결 서비스의 데이터를 XML 형식으로 허용하고 레코드로 변환합니다. DaaS 업데이트 비즈니스 항목 서비스는 두 가지 XML 파일 형식으로 외부 서비스의 데이터를 허용합니다. 두 가지 XML 파일은 변경하기 전과 후의 데이터에 해당합니다. 업데이트 서비스는 변경 내용을 해당하는 레코드에 적용합니다.

관련 항목:

- “DaaS 가져오기” 페이지 119
- “DaaS 업데이트” 페이지 122

연결 지원 구성

D&B의 연결 서비스를 사용하여 레코드를 생성하고 업데이트하려면 프로비저닝 도구에서 구성을 추가하고, 연결 서비스에서 응답을 분할하는 사용자 지정 응용 프로그램을 생성해야 합니다.

다음 태스크를 수행하여 D&B의 연결 서비스에 대한 지원을 구성합니다.

1. 프로비저닝 도구를 사용하여 연결 서비스에 대한 WSDL을 업로드합니다.
2. 프로비저닝 도구를 사용하여 XML 문서에서 비즈니스 항목으로의 변환을 구성하고 이를 서비스로 노출합니다. 변환을 서비스로 노출할 경우, 프로세스에서 DaaS 가져오기 및 업데이트 비즈니스 항목 서비스가 생성됩니다.
3. 연결 서비스의 데이터를 요청하고 응답을 레코드 세부 정보와 연결 세부 정보로 분할할 수 있는 사용자 지정 응용 프로그램을 생성합니다.
4. 사용자 지정 응용 프로그램에 요청을 보내는 사용자 인터페이스를 개발합니다.

참고: WSDL을 업로드하고 XML에서 비즈니스 항목으로의 변환을 구성하는 방법에 대한 자세한 내용은 *Multidomain MDM 프로비저닝 도구 가이드*에서 DaaS(Data as a Service) 통합 장을 참조하십시오.

연결 데이터 분할을 위한 사용자 지정 응용 프로그램

D&B의 연결 서비스를 사용하려면 연결 정보를 레코드 세부 정보와 연결 세부 정보로 분할할 수 있는 사용자 지정 응용 프로그램을 디자인해야 합니다.

사용자 지정 응용 프로그램은 다음과 같은 기능을 수행해야 합니다.

1. 항목 보기에서 연결 서비스에 대한 요청을 허용합니다..
2. D&B에 요청을 보내고 응답을 수신합니다.
3. 응답을 XML로 변환합니다.
4. 응답을 레코드 세부 정보 및 연결 세부 정보로 분할합니다.
5. XML 정보를 비즈니스 항목 서비스로 보내 레코드로 데이터베이스에 저장합니다.
6. 데이터 변경 내용을 모니터링하고 외부 서비스의 변경 알림 나열 함수를 호출합니다.

제 7 장

데이터 정리, 분석 및 변환을 위한 외부 호출

이 장에 포함된 항목:

개요

외부 공급자는 레코드 데이터를 정리, 분석 및 변환하는 웹 서비스를 제공합니다. 레코드를 추가할 때 주소 필드가 비어 있는지 여부를 확인하는 것 같은 사용자 지정 유효성 검사에 대해 외부 웹 서비스를 사용합니다. 외부 웹 서비스는 레코드 데이터를 변환하는 사용자 지정 논리에도 사용됩니다. 예를 들어 레코드 두 개를 병합할 경우 주소는 병합하되 전화 번호는 병합하지 않을 수 있습니다.

외부 웹 서비스는 비즈니스 항목 서비스가 호출할 수 있는 작업을 하나 이상 노출합니다. 각 작업에는 요청 및 응답 유형이 있습니다. 비즈니스 항목 서비스는 필수 서비스 매개 변수를 사용하여 레코드 데이터를 외부 서비스에 보냅니다. 실행 논리의 특정 단계에 대해 외부 웹 서비스에 대한 호출을 구성할 수 있습니다. 구현하는 논리에 따라 레코드 데이터를 업데이트하는 요청이 Data Director에서 전송됩니다. 필요한 경우 외부 서비스가 데이터를 수정할 수 있습니다.

프로비저닝 도구에서 외부 서비스를 호출할 비즈니스 항목 및 이벤트를 구성합니다. 프로비저닝 도구에서 외부 서비스용 WSDL 파일을 업로드하고 SOAP 서비스 및 작업을 등록합니다. 서비스를 특정 비즈니스 항목과 이벤트에 바인딩합니다.

리소스 키트에 있는 WSDL 파일을 사용하여 서비스, 작업, 메서드 및 서비스 메서드가 교환하는 데이터 유형을 이해합니다. 외부 웹 서비스용 `custom-logic-service.wsdl` 파일은 다음 리소스 키트 위치에 있습니다. `C:\<infamdm installation directory>\hub\resourcekit\samples\BESExternalCall\source\resources\webapp\WEB-INF\wsdl\`

리소스 키트에는 사용자 지정 논리 및 유효성 검사를 구현하는 샘플 코드가 포함되어 있습니다. 리소스 키트를 설치하면 샘플 사용자 지정 논리 및 유효성 검사를 위한 `bes-external-call.ear` 파일이 응용 프로그램 서버에 배포됩니다.

지원되는 이벤트

비즈니스 항목 서비스는 서비스 단계로 구성됩니다. 사용자 지정 논리 및 유효성 검사를 모든 단계에 적용할 수 있습니다.

다음의 이벤트에 대해 외부 호출을 실행할 수 있습니다.

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

외부 호출을 구성하는 방법

비즈니스 항목 서비스에는 서비스 단계가 있습니다. 수신 요청은 각 서비스 단계를 통과합니다. 비즈니스 항목 서비스 실행 논리의 특정 단계에 외부 서비스에 대한 호출을 구성할 수 있습니다.

외부 호출을 구성하려면 다음 단계를 수행합니다.

1. `bes-external-call.ear` 파일을 빌드하고 배포합니다.
2. 프로비저닝 도구에서 다음 태스크를 수행합니다.
   a. 외부 서비스용 WSDL 파일을 업로드합니다.
   b. 웹 서비스를 SOAP 서비스로 등록합니다.
   c. 외부 호출을 구성합니다.

WSDL 파일 업로드, SOAP 서비스 등록 및 외부 호추 구성에 대한 자세한 내용은 *Multidomain MDM 프로비저닝 도구 가이드*를 참조하십시오.

EAR 파일 작성 및 배포에 대한 자세한 내용은 *Multidomain MDM 리소스 키트 가이드*를 참조하십시오.

예: 비즈니스 항목 서비스에 대한 사용자 지정 유효성 검사 및 논리

Person 레코드를 추가하고 병합할 때 사용자 지정 유효성 검사 및 논리를 테스트할 수 있습니다. 사용자 지정 유효성 검사는 Person 레코드에 주소가 있는지 확인합니다. 사용자 지정 논리는 전화 번호 두 개를 병합하도록 허용하지 않습니다. REST API를 사용하여 Person 레코드를 생성하고 병합합니다.

1. Person 레코드를 생성할 때 유효성 검사를 확인하려면 다음 단계를 수행합니다.
   a. 생성 API를 사용하여 주소가 없는 Person 레코드를 생성합니다. 유효성 검사 오류가 발생합니다.
   b. 생성 API를 사용하여 주소가 있는 Person 레코드를 생성합니다. 작업이 성공적으로 수행됩니다.
2. 레코드를 병합할 때 사용자 지정 논리를 확인하려면 다음 단계를 수행합니다.
   a. 생성 API를 사용하여 주소와 전화 번호가 있는 Person 레코드 두 개를 생성합니다.
   b. 병합 미리 보기 API를 사용하여 Person 레코드 두 개를 병합합니다. 병합 미리 보기 요청에 overrides를 추가하여 주소와 전화 번호를 병합합니다. 응답에 주소는 하나만 표시되지만 전화 번호는 두 개입니다. 사용자 지정 논리는 전화 번호가 병합되는 것을 방지합니다.

선행 조건

사용자 지정 논리 및 유효성 검사를 확인하려면 프로비저닝 도구에서 WSDL 파일을 업로드해야 합니다. SOAP 서비스 및 작업을 등록해야 합니다. 사용자 지정 논리 및 유효성 검사를 사용할 비즈니스 항목 및 이벤트에 서비스를 바인딩합니다. 지정된 비즈니스 항목 및 이벤트에 대해 논리 및 유효성 검사를 테스트할 수 있습니다.

1단계. 사용자 지정 유효성 검사 테스트

생성 API를 사용하여 주소가 없는 다음 Person 레코드를 생성합니다.

```
POST http://localhost:8080/cmx/cs/localhost-orcl-mdm_Sample/Person?systemName=Admin
{
        firstName: "John"
    }
```

유효성 검사 오류가 발생합니다.

생성 API를 사용하여 주소가 있는 다음 Person 레코드를 생성합니다.

```
POST http://localhost:8080/cmx/cs/localhost-orcl-mdm_Sample/Person?systemName=Admin
{
        firstName: "John",
        Addresses: {
            item: [
                {
                    cityName: "Toronto"
                }
            ]
        }
    }
```

요청이 Person 레코드를 생성합니다.

2단계. 사용자 지정 논리 테스트

다음 단계를 수행하여 사용자 지정 논리를 테스트합니다.

1. 생성 API를 사용하여 주소와 전화 번호가 있는 Person 레코드 두 개를 생성합니다.

```
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"
                }
            ]
        }
    }
```

응답에 다음과 같은 rowId가 포함됩니다.

- Person: 161923, 161924
- Addresses: 2123, 2124
- TelephoneNumbers: 101723, 101724

2. PreviewMerge API를 실행하여 두 Person 레코드를 병합합니다.

```
POST http://localhost:8080/cmx/cs/localhost-orcl-mdm_sample/Person/161923?action=previewMerge&depth=2
{
        keys: [
            {
                rowid: "161924"
            }
        ]
    }
```

응답은 주소 두 개와 전화 번호 두 개가 있는 Person 레코드입니다.

```
{
  "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. PreviewMerge API를 실행하여 주소 및 전화 번호를 병합하는 재정의와 함께 Person 레코드 두 개를 병합합니다.

```
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]
                            }
                        }
                    }
                ]
            }
        }
    }
}
```

응답에 주소는 하나만 표시되지만 전화 번호는 두 개입니다.

```
{
  "Person": {
```

```
 "rowidObject": "161923        ",
 "creator": "admin",
 "createDate": "2016-10-20T09:50:35.878-04:00",
 "updatedBy": "admin",
 "lastUpdateDate": "2016-10-20T09:50:35.879-04:00",
 "consolidationInd": 4,
 "lastRowidSystem": "SYSO          ",
 "hubStateInd": 1,
 "label": "Person: , Bill",
 "partyType": "Person",
 "displayName": "Bill",
 "firstName": "Bill",
"TelephoneNumbers": {
   "link": [],
   "firstRecord": 1,
   "recordCount": 2,
   "pageSize": 2,
   "item": [
     {
       "rowidObject": "101723        ",
       "creator": "admin",
       "createDate": "2016-10-20T09:50:35.904-04:00",
       "updatedBy": "admin",
       "lastUpdateDate": "2016-10-20T09:50:35.905-04:00",
       "consolidationInd": 4,
       "lastRowidSystem": "SYSO          ",
       "hubStateInd": 1,
       "label": "PhoneNumbers",
       "phoneNum": "111-1111     ",
       "phoneCountryCd": "1"
     },
     {
       "rowidObject": "101724        ",
       "creator": "admin",
       "createDate": "2016-10-20T09:50:54.768-04:00",
       "updatedBy": "admin",
       "lastUpdateDate": "2016-10-20T09:50:54.769-04:00",
       "consolidationInd": 4,
       "lastRowidSystem": "SYSO          ",
       "hubStateInd": 1,
       "label": "PhoneNumbers",
       "phoneNum": "222-2222     ",
       "phoneCountryCd": "1"
     }
   ]
 },
  "Addresses": {
   "link": [],
   "firstRecord": 1,
   "recordCount": 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": "SYSO          ",
       "hubStateInd": 1,
       "label": "Addresses"
     }
   ]
 },
 "PersonDetails": {
   "link": [],
   "recordCount": 0,
   "pageSize": 0,
   "item": []
```

```
    }
  }
}
```

부록 A

REST API를 사용하여 레코드 추가

이 부록에 포함된 항목:

REST API를 사용하여 레코드 추가 개요

비즈니스 항목 모델을 생성하고 비즈니스 항목 구조를 구성한 후 REST API를 사용하여 레코드를 추가할 수 있습니다.

다음 섹션에서는 Person 비즈니스 항목의 예를 사용하여 REST API를 통해 레코드를 추가하는 방법을 보여 줍니다. Person 비즈니스 항목에는 회사의 직원에 대한 데이터가 포함되어 있습니다.

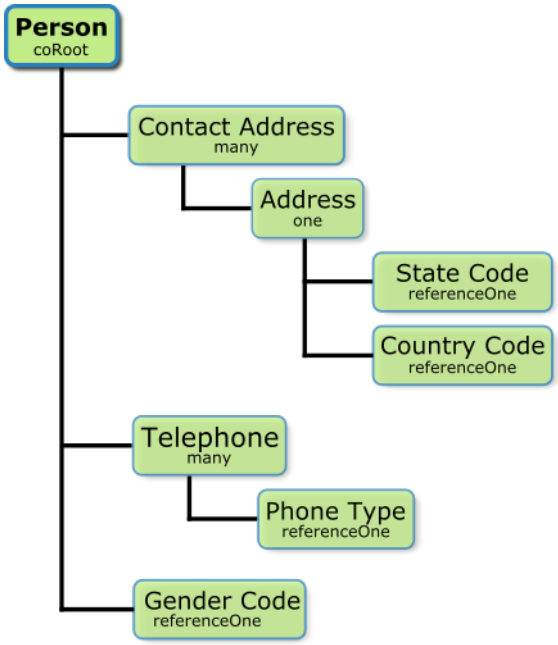
다음 API를 사용하여 직원에 대한 세부 정보를 추가하십시오.

1. 스키마에 대한 정보를 가져옵니다. 메타데이터 가져오기 REST API를 사용하여 구조, 필드 목록, 필드 유형 및 조회 필드에 대한 세부 정보를 포함하여 비즈니스 항목의 데이터 구조에 대한 정보를 가져올 수 있습니다. 또는 사용할 수 있는 요소와 특성이 설명되어 있는 XSD(XML 스키마 정의) 파일을 액세스할 수도 있습니다. XSD 파일은 `http:://<호스트>:<포트>/cmx/csfiles`에 있습니다.
2. 레코드를 생성합니다. 레코드 생성 REST API를 사용하여 레코드를 생성합니다.
3. 추가한 레코드에서 데이터를 읽습니다. 레코드 읽기 REST API를 사용하여 레코드에서 데이터를 검색합니다.

Person 비즈니스 항목 구조

REST API를 사용하여 Person 레코드를 추가해 보겠습니다. Person 루트 노드는 Person 비즈니스 항목 구조의 최상위 노드입니다. Person 루트 노드 아래에는 성별, 주소 및 전화 번호 같은 직원 세부 정보에 해당하는 노드가 있습니다.

다음 이미지는 Person 비즈니스 항목의 구조를 보여 줍니다.



Person 비즈니스 항목의 루트 노드는 Person입니다. 노드 이름 아래에 나와 있는 노드 유형은 상위 노드와 하위 노드 사이의 관계를 나타냅니다. Contact Address와 Address 사이에는 일대일 관계가 성립되며, 이는 각 연락처 주소에 주소가 하나만 연결될 수 있음을 나타냅니다. Person과 Telephone 사이에는 일대다 관계가 성립되며, 이는 Person 레코드에 여러 개의 전화 번호 레코드가 연결될 수 있음을 나타냅니다. Person과 Gender 사이에는 일대일 관계가 성립되며, 이는 Person 레코드의 성별 값은 하나뿐임을 나타냅니다. 성별 값은 조회 테이블에 저장됩니다. 마찬가지로 상태 코드 및 국가 코드 값도 조회 테이블에 저장됩니다.

1단계. 스키마에 대한 정보 가져오기

메타데이터 가져오기 REST API를 사용하여 스키마에 대한 정보를 가져올 수 있습니다. 메타데이터 가져오기 API는 비즈니스 항목의 데이터 구조를 반환합니다. 메타데이터에는 비즈니스 항목 필드, 필드 유형 및 조회 필드에 대한 세부 정보가 나열됩니다.

메타데이터 가져오기 URL의 형식은 다음과 같습니다.

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

다음 샘플 요청은 Person 비즈니스 항목에 대한 메타데이터를 검색합니다.

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

메타데이터 응답 가져오기

다음 예에서는 Person 비즈니스 항목의 데이터 구조에서 발췌한 코드를 보여 줍니다.


```
{
    "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
    }
}
```

2단계. 레코드 생성

레코드 생성 REST API를 사용하여 레코드를 생성합니다. 비즈니스 항목의 이름과 소스 시스템의 이름은 필수 매개 변수입니다. 요청 본문의 레코드에 대한 데이터를 보냅니다.

레코드 생성 URL의 형식은 다음과 같습니다.

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

systemName 매개 변수는 필수 매개 변수이며 소스 시스템의 이름을 지정합니다.

Person 비즈니스 항목에는 Person 루트 노드와 두 번째 수준의 Address, Gender 및 Phone 노드가 있습니다.

다음 샘플 요청은 Person 레코드를 생성합니다.

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

요청 본문에는 Person 레코드의 다음과 같은 세부 정보가 지정됩니다.

- 이름
- 성
- Gender.
- State Code 및 Country Code가 포함된 Address
- Phone Number 및 Phone Type(예: 집, 및 휴대폰)

레코드 응답 생성

다음 샘플 응답은 Person 레코드를 성공적으로 생성한 후의 응답을 보여 줍니다.


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

참고: 응답 본문에는 레코드 및 생성된 rowId가 포함됩니다.

레코드를 생성할 때 워크플로우 프로세스를 시작하도록 구성하면 다음과 같은 현상이 발생합니다.

- 레고드가 보류 상태로 작성됩니다.
- 워크플로우 프로세스가 시작됩니다.
- 워크플로우 프로세스 ID가 응답 헤더에 반환됩니다.

워크플로우 프로세스를 구성하지 않으면 기본적으로 레코드가 활성 상태로 작성됩니다.

상호 작용 ID를 사용하여 요청을 처리하는 경우 API가 상호 작용 ID를 응답 헤더에 반환합니다.

3단계. 레코드 읽기

레코드 읽기 REST API를 사용하여 추가한 루트 레코드에 대한 세부 정보를 검색합니다. 해당 API를 사용하여 루트 레코드의 하위 레코드에 대한 세부 정보를 검색할 수 있습니다.

레코드 읽기 URL의 형식은 다음과 같습니다.

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

depth 매개 변수를 사용하여 반환할 하위 수준 수를 지정할 수 있습니다. 루트 노드 및 직접 하위 노드를 반환하려면 2를 지정하고 루트 노드, 직접 하위 노드 및 두 수준 하위 노드를 반환하려면 3을 지정하십시오. 하위 레코드에 대한 세부 정보를 반환하려면 다음 URL 형식을 사용하십시오.

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

다음 샘플 요청은 루트 노드, 바로 아래의 하위 노드 및 두 수준 아래의 하위 노드에 대한 세부 정보를 반환합니다.

```
GET http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/Person/658248?depth=3
```

요청은 활성 레코드에 대한 세부 정보를 반환합니다.

참고: 레코드를 생성할 때 워크플로우가 시작된 경우에는 생성한 레코드가 보류 상태가 됩니다. 기본적으로 레코드 읽기 요청은 활성 레코드를 읽습니다. recordStates 매개 변수를 사용하여 레코드의 보류 중 상태를 지정할 수 있습니다.

다음 샘플 요청은 보류 중인 레코드에 대한 세부 정보를 읽습니다.

```
GET http://localhost:8080/cmx/cs/localhost-hub101-ds_ui1/Person/658248?depth=3&recordStates=PENDING
```

레코드 응답 읽기

다음 샘플 응답은 추가한 레코드에 대한 세부 정보를 보여 줍니다.

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

부록 B

REST API를 사용하여 파일 업로드

이 부록에 포함된 항목:

REST API를 사용하여 파일 업로드 개요

REST API를 사용하여 파일을 저장소 유형에 업로드할 수 있습니다. 파일을 업로드한 후 레코드 또는 태스크에 파일을 첨부하거나 파일을 사용하여 Data Director 사용자 인터페이스를 지역화할 수 있습니다.

파일을 어떻게 사용하려는지에 따라 사용하는 REST API, 파일 구성 요소 및 저장소 유형의 조합이 달라질 수 있습니다. 예를 들어 레코드 또는 태스크에 파일을 첨부하려면 파일의 메타데이터를 생성하고 파일을 임시 저장소에 업로드합니다. 파일을 업로드한 후 데이터베이스의 레코드에 파일을 첨부하거나 BPM 저장소의 태스크에 파일을 첨부할 수 있습니다. Data Director 사용자 인터페이스를 지역화하려면 ZIP 파일을 다운로드하고, ZIP 파일을 수정한 후, 수정된 ZIP 번들을 번들 저장소에 업로드합니다.

파일용 REST API

범용 REST API 집합을 사용하여 첨부 파일 또는 지역화를 위한 파일을 업로드하고 관리할 수 있습니다.

다음 테이블에는 파일용 REST API가 나열되어 있습니다.

REST API	설명	지원되는 저장소 유형
파일 메타데이터 나열	저장소의 파일 메타데이터 목록을 반환합니다.	BPM 또는 TEMP
파일 메타데이터 생성	저장소의 파일에 대한 메타데이터를 생성합니다.	DB 또는 TEMP
파일 메타데이터 가져오기	파일의 메타데이터를 반환합니다.	BPM, BUNDLE, DB 또는 TEMP
파일 메타데이터 업데이트	파일의 메타데이터를 업데이트합니다.	DB 또는 TEMP
파일 콘텐츠 업로드	파일의 콘텐츠를 저장소에 업로드합니다.	BUNDLE, DB 또는 TEMP
파일 콘텐츠 가져오기	파일의 콘텐츠를 다운로드합니다.	BPM, BUNDLE, DB 또는 TEMP
파일 삭제	파일 메타데이터 또는 콘텐츠 등 파일에 연결된 구성 요소를 포함하여 저장소의 파일을 삭제합니다.	BUNDLE, DB 또는 TEMP

파일 구성 요소

레코드 또는 태스크에 파일을 첨부하려면 파일의 메타데이터를 생성한 다음 파일 콘텐츠를 업로드합니다. Data Director 사용자 인터페이스를 지역화하려면 리소스 번들 파일을 다운로드한 다음 수정된 리소스 번들 파일을 업로드합니다.

파일 메타데이터

파일 이름, 파일 유형 및 콘텐츠 유형 등 파일에 대한 정보입니다. 저장소 유형에 따라 생성자, 생성 시간 및 업로드 날짜 같은 다른 매개 변수를 포함해야 할 수 있습니다.

파일 콘텐츠

파일의 콘텐츠입니다. 예를 들어 텍스트, 이미지, 문서 또는 리소스 번들입니다.

저장소 유형

지원되는 저장소 구현에 파일을 업로드하고 저장합니다. 사용하는 저장소 유형은 Data Director 사용자 인터페이스를 지역화할지 여부 또는 레코드 또는 태스크에 파일을 첨부할지 여부에 따라 다릅니다.

다음 목록에는 지원되는 저장소 유형이 설명되어 있습니다.

BPM

태스크에 첨부된 파일을 태스크 데이터와 함께 저장합니다. 파일을 태스크에 첨부하면 프로세스에서 TEMP 저장소의 파일이 BPM 저장소에 저장됩니다.

BPM 저장소에 저장된 파일은 다음 파일 ID 형식을 사용합니다. taskId::filename.

참고: 트리거된 태스크 또는 기존 태스크에 파일을 첨부하려면 프로비저닝 도구에서 태스크 트리거, 태스크 유형 및 태스크 작업에 대해 첨부 파일을 활성화합니다. 자세한 내용은 *Multidomain MDM 프로비저닝 도구 가이드*를 참조하십시오.

BUNDLE

Data Director 사용자 인터페이스를 지역화하는 리소스 번들 파일을 저장합니다.

BUNDLE 저장소에 저장된 파일은 다음 파일 ID 형식을 사용합니다. besMetadata.

DB

레코드의 첨부 파일을 C_REPOS_ATTACHMENTS 테이블에 저장합니다. 파일을 레코드에 첨부하면 프로세스에서 TEMP 저장소의 파일이 DB 저장소에 저장됩니다.

DB 저장소에 저장된 파일은 다음 파일 ID 형식을 사용합니다. DB_<RowID>.

참고: 레코드에 파일을 첨부하려면 프로비저닝 도구에서 FileAttachment를 데이터 유형으로 사용하여 필드를 구성합니다. 데이터 유형 구성에 대한 자세한 내용은 *Multidomain MDM 프로비저닝 도구 가이드*를 참조하십시오.

TEMP

파일을 C_REPOS_ATTACHMENTS 테이블에 임시로 저장하고 파일을 TEMP로 표시합니다. BPM 또는 DB 저장소에 성공적으로 업로드되거나 미리 구성된 만료 시간이 지난 파일은 TEMP 저장소에서 삭제됩니다.

TEMP 저장소에 저장된 파일은 다음 파일 ID 형식을 사용합니다. TEMP_<ROWID_ATTACHMENT>.

만료 시간 구성에 대한 자세한 내용은 *Multidomain MDM 구성 가이드*를 참조하십시오.

레코드에 파일 첨부

레코드에 파일을 첨부하기 전에 파일의 메타데이터를 생성한 다음 임시 저장소에 파일을 업로드합니다.

1. 파일의 메타데이터를 생성하려면 TEMP를 저장소 유형으로 지정하고 파일 메타데이터 생성 REST API를 사용합니다.
   예를 들어 다음 요청은 Document_3.pdf 파일에 대한 메타데이터를 생성합니다.

   ```
   POST http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/TEMP
   Content-Type: application/json
   {
       "fileName": "Document_3.pdf",
       "fileType": "pdf",
       "fileContentType": "application/pdf"
   }
   ```

   참고: 파일 메타데이터는 항상 TEMP 저장소에 생성합니다.

   파일 메타데이터 생성 REST API는 파일의 ID를 반환합니다. 파일 ID 형식은 다음과 같습니다. <저장소 유형>_<RowID>. 여기서 RowID는 저장소에 업로드하는 파일의 행 ID를 나타냅니다.

   이 예에서 API 호출은 Document_3.pdf 파일에 대한 다음 ID를 반환합니다. TEMP_SVR1.0JU3

   파일 ID를 사용하여 파일을 업로드, 첨부, 업데이트, 다운로드 및 삭제할 수 있습니다.
2. 파일을 업로드하려면 TEMP를 저장소 유형으로 지정하고 파일 콘텐츠 업로드 REST API를 사용합니다.

예를 들어 다음 요청은 파일을 TEMP 저장소에 업로드합니다.

```
PUT http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/TEMP/TEMP_SVR1.0JU3/content
Content-Type: application/octet-stream
<file object (upload using REST client)>
```

참고: 파일을 업로드한 후 TEMP 저장소에는 미리 구성된 기간인 60분 동안 파일이 저장됩니다. 미리 구성된 기간이 만료되기 전에 파일을 레코드에 첨부해야 합니다.

3. 레코드를 생성하고 파일을 새 레코드에 첨부하려면 레코드 생성 REST API를 사용합니다.

   예를 들어 다음 요청은 레코드를 생성하고 파일 ID TEMP_SVR1.0JU3을 사용하여 파일을 첨부합니다.

```
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_SVR1.0JU3"
         }
      ]
   }
}
```

   참고: 파일을 레코드에 첨부하면 프로세스에서 파일이 데이터베이스에 저장됩니다. 파일의 ID는 DB_<RowID>로 변경됩니다. 여기서 DB는 파일이 데이터베이스에 저장됨을 나타냅니다.

4. 레코드에 첨부된 파일을 바꾸려면 DB를 저장소 유형으로 지정하고 파일 콘텐츠 업로드 REST API를 사용합니다.
   예를 들어 다음 요청은 데이터베이스에서 파일 ID가 DB_SVR1.0JU3인 첨부된 파일을 바꿉니다.

```
PUT http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/DB/DB_SVR1.0JU3/content
Content-Type: application/octet-stream
<file object (upload using REST client)>
```

   참고: 요청 URL의 저장소 유형은 DB입니다.

5. 레코드에 파일을 첨부한 후 파일 메타데이터를 편집하려면 DB를 저장소 유형으로 지정하고 파일 메타데이터 업데이트 REST API를 사용합니다.
   예를 들어 다음 요청은 DB 저장소에서 파일 ID DB_SVR1.0JU3에 연결된 파일의 파일 메타데이터를 업데이트합니다.

```
PUT http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/DB/DB_SVR1.0JU3
Content-Type: application/json
{
    "fileName": "Document_4.pdf",
    "fileType": "pdf",
    "fileContentType": "application/pdf"
}
```

6. 레코드에 첨부된 파일을 다운로드하려면 DB를 저장소 유형으로 지정하고 파일 콘텐츠 가져오기 REST API를 사용합니다.
   예를 들어 다음 요청은 DB 저장소에서 파일 ID DB_SVR1.0JU3에 연결된 파일을 다운로드합니다.

```
GET http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/DB/DB_SVR1.0JU3/content
```

7. 레코드에 첨부된 파일을 삭제하려면 DB를 저장소 유형으로 지정하고 파일 삭제 REST API를 사용합니다.
   예를 들어 다음 요청은 DB 저장소에서 파일 ID DB_SVR1.0JU3에 연결된 파일을 삭제합니다.

```
DELETE http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/DB/DB_SVR1.0JU3
```

태스크에 파일 첨부

파일의 메타데이터를 생성한 다음 파일 콘텐츠를 임시 저장소에 업로드합니다. 파일을 업로드한 후 트리거된 태스크 또는 기존 태스크에 파일을 첨부합니다.

참고: 트리거된 태스크 또는 기존 태스크에 파일을 첨부하려면 프로비저닝 도구에서 태스크 트리거, 태스크 유형 및 태스크 작업에 대해 첨부 파일을 활성화합니다. 자세한 내용은 *Multidomain MDM 프로비저닝 도구 가이드*를 참조하십시오.

1. 파일의 메타데이터를 생성하려면 TEMP를 저장소 유형으로 지정하고 Create File Metadata REST API를 사용합니다.
   예를 들어 다음 요청은 file1.txt 파일에 대한 메타데이터를 생성합니다.

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

   {
       “fileName”: “file1.txt”,
       “fileType”: “text”,
       “fileContentType”: “text/plain”
   }
   ```

   참고: 파일 메타데이터는 항상 TEMP 저장소에 생성합니다.

   파일 메타데이터 생성 REST API는 파일의 ID를 반환합니다. 파일 ID 형식은 다음과 같습니다. <저장소 유형>_<RowID>. 여기서 RowID는 저장소에 업로드하는 파일의 행 ID를 나타냅니다.
   이 예에서 API 호출은 file1.txt에 대한 다음 ID를 반환합니다. TEMP_SVR1.1VDVS

   파일 ID를 사용하여 파일을 업로드, 첨부, 업데이트 및 삭제할 수 있습니다.

2. 파일을 업로드하려면 TEMP를 저장소 유형으로 지정하고 Upload File Content REST API를 사용합니다.
   예를 들어 다음 요청은 파일을 TEMP 저장소에 업로드합니다.

   ```
   PUT http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/TEMP/TEMP_SVR1.1VDVS/content

   Test attachment content: file 1
   ```

   참고: 파일을 업로드한 후 TEMP 저장소에는 미리 구성된 기간인 60분 동안 파일이 저장됩니다. 미리 구성된 기간이 만료되기 전에 파일을 태스크에 첨부해야 합니다.

3. 레코드를 관리할 때 트리거되는 태스크에 파일을 첨부합니다.

   - 레코드를 생성할 때 트리거되는 태스크에 파일을 첨부하려면 taskattachments 매개 변수와 함께 비즈니스 항목 생성 REST API를 사용합니다.
     예를 들어 다음 요청은 레코드를 생성하고 파일 ID TEMP_SVR1.1VDVS를 사용하여 파일을 첨부합니다.

     ```
     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"
                    }
               ]
          }
     }
     ```

   - 레코드를 업데이트할 때 트리거되는 태스크에 파일을 첨부하려면 taskattachments 매개 변수와 함께 비즈니스 항목 업데이트 REST API를 사용합니다.

예를 들어 다음 요청은 레코드를 업데이트하고 파일 ID TEMP_SVR1.1VDVS를 사용하여 파일을 첨부합니다.

```
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"
    }
}
```

- 레코드를 병합할 때 트리거되는 태스크에 파일을 첨부하려면 taskattachments 매개 변수와 함께 비즈니스 항목 병합 REST API를 사용합니다.
  예를 들어 다음 요청은 레코드를 병합하고 파일 ID TEMP_SVR1.1VDVS를 사용하여 파일을 첨부합니다.

```
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"
    }
  }
}
```

- 레코드를 병합 해제할 때 트리거되는 태스크에 파일을 첨부하려면 taskattachments 매개 변수와 함께 비즈니스 항목 병합 해제 REST API를 사용합니다.
  예를 들어 다음 요청은 레코드를 병합 해제하고 파일 ID TEMP_SVR1.1VDVS를 사용하여 파일을 첨부합니다.

```
POST http://localhost:8080/cmx/cs/localhost-orcl-DS_UI1/Person/2478248?
action=unmerge&taskAttachments=TEMP_SVR1.1VDVS
{
        rowid: "4880369"
}
```

4. 기존 태스크에 파일을 첨부합니다.

   - 태스크를 업데이트할 때 파일을 첨부하려면 태스트 업데이트 REST API를 사용하고 요청 본문에 attachments를 포함합니다.
     예를 들어 다음 요청은 태스크를 업데이트하고 파일 ID TEMP_SVR1.1VDVS를 사용하여 파일을 첨부합니다.

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

- 태스크 작업을 실행할 때 파일을 첨부하려면 요청 본문의 attachments와 함께 태스크 작업 실행 REST API를 사용합니다.
  예를 들어 다음 요청은 태스크 작업을 실행하고 파일 ID TEMP_SVR1.1VDVS를 사용하여 파일을 첨부합니다.

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

태스크에 파일을 첨부하면 TEMP 저장소에서 파일이 이동하고 태스크 데이터와 함께 BPM 저장소에 파일이 저장됩니다. 파일의 ID가 taskId::filename으로 변경됩니다.

리소스 번들 파일 업로드

Data Director 사용자 인터페이스를 지역화하려면 리소스 번들 ZIP 파일을 다운로드하고, ZIP 파일의 파일을 수정한 후, 수정된 ZIP 파일을 번들 저장소에 업로드합니다.

1. 리소스 번들 ZIP 파일을 다운로드하려면 BUNDLE을 저장소 유형으로 지정하고 파일 콘텐츠 가져오기 REST API를 사용합니다.
   예를 들어 다음 요청은 리소스 번들 ZIP 파일을 다운로드합니다.

   ```
   GET http://localhost:8080/cmx/file/localhost-orcl-MDM_SAMPLE/BUNDLE/besMetadata/content
   ```

2. 번들 파일과 관련된 언어를 추가하여 ZIP 파일을 수정합니다.
   예를 들어 필드 이름, 레이블 및 테이블 이름을 러시아어로 지역화하려면 besMetadata_ru.properties 파일을 추가합니다.

3. 수정된 리소스 번들 ZIP 파일을 업로드하려면 BUNDLE을 저장소 유형으로 지정하고 파일 콘텐츠 업로드 REST API를 사용합니다.
   예를 들어 다음 요청은 리소스 번들 ZIP 파일을 업로드합니다.

   ```
   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
   ```

인 덱 스

N

R

S

U

W

ㄱ