| Author | Scott Krause | Date | 03 Nov 2013 |
|---|---|---|---|

# Table of Contents



# Overview

This document describes the contents, structure and procedures surrounding the Release Candidate promotion unit for Informatica MDM 9.1. A Release Candidate is a file archive that contains a particular version of an MDM domain application. An MDM domain application is a Master Data implementation of a single business subject such as, Customer Master or Product Hierarchy.

# Incremental and Reconstructive

Each Release Candidate is structured so that in contains both the assets for the immediate incremental promotion and the assets required for a disaster recovery reconstruction of the entire domain application; having the additional advantage of allowing for holistic comparison of differences between versions.

## Naming Convention

A Release Candidate is named to reflect the NPV Val Item, domain and version identifier, for example `lcs00616_016_product_hierarchy_v408.zip`. The characters in the name are positioned so that they will sort logically when displayed in both the Windows explorer and MKS.

## Folder Hierarchy

The folders are divided into two deployment stages, *pre* and *post*. The *pre* folders are populated with the latest (incremental) changes and these are the folders that are captured in the CII document. The *post* folders are populated after the promotion so as to include a snapshot of the entire version.

| Folder | Contents Description | Stage |
|---|---|---|
| `name/deploy_dba` | Data Definition Language scripts | *Pre* |
| `name/deploy_idd` | Zip and/ or XML files | *Pre* |
| `name/deploy_met` | Change Lists / Hub Design Objects | *Pre* |
| `name/scripts` | UNIX Shell scripts | *Pre* |
| `name/scripts/sql` | SQL scripts scheduled | *Pre* |
| `name/scripts/util` | SQL scripts un-scheduled | *Pre* |
| `name/post/schema_ddl` | Data Definition Language scripts | *Post* |
| `name/post/idd_export` | Entire exported IDD app (zip) | *Post* |
| `name/post/public_queries` | A single XML file | *Post* |
| `name/post/met_export` | Entire ORS Metadata export (change list) | *Post* |
| `name/post/java_classes` | Java Project source code | *Post* |
| `name/post/all_scripts` | All scripts that exist on the app server file system | *Post* |
| `name/post/autosys` | A single JIL file containing all ADD_PMDSM jobs | *Post* |

Table of RC folders denoting pre or post stage

## Procedures

### *Pre-deployment contents*

#### Staging on a Networked Share

The entire Release Candidate is copied onto a shared network location prior to promotion. At this point the *post* folders are empty. The Release Candidate is staged within the following directory:

```
\\LCADDFILE01\ADD_Data$\D2RADMS\DS_dMDM
```

Shared Folder UNC location

## Updating the CII

The CII (Configuration Item Inventory) should list only the contents of the folders in which the stage is *Pre*. Initially the *Post* folders will be empty.

To create a file listing of all files in the folder hierarchy including file names, sizes and time/date stamps. Execute the command below, replacing [Release Name] with the Release Candidate name (see Naming Convention).

```
dir \\LCADDFILE01\ADD_Data$\D2RADMS\DS_dMDM\Release Name\*.*.* /s
/o:n >C:\Release Name_cii_files.txt
```

Command Line Syntax

*Note 1:* You must have permissions to write a file to the root folder of drive C
*Note 2:* The three star matching pattern is required because the change list file names have two periods (name.change.xml)

*Post-deployment contents*

## Post Deployment Snapshot

The contents of the post folders are populated after the Release Candidate has been successfully promoted onto the quality environment.

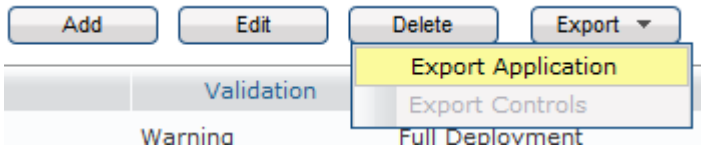
1. **post/schema_ddl –** This folder is populated with the DDL results. A post deployment snapshot may contain more than one schema, for example ADD_ORS and MDM_PRDT_HIER. A DDL script can be generated with the following script:

```
set heading off
set pagesize 0
set long 90000
set feedback off
set echo off

spool release_candidate.sql
SELECT DBMS_METADATA.GET_DDL('VIEW',u.view_name)  FROM USER_VIEWS u;
SELECT DBMS_METADATA.GET_DDL('TABLE',u.table_name)  FROM USER_TABLES u;
SELECT DBMS_METADATA.GET_DDL('INDEX',u.index_name)  FROM USER_INDEXES u;
SELECT * FROM USER_SOURCE;
spool off;
```

PL/SQL Syntax to generate a DDL script

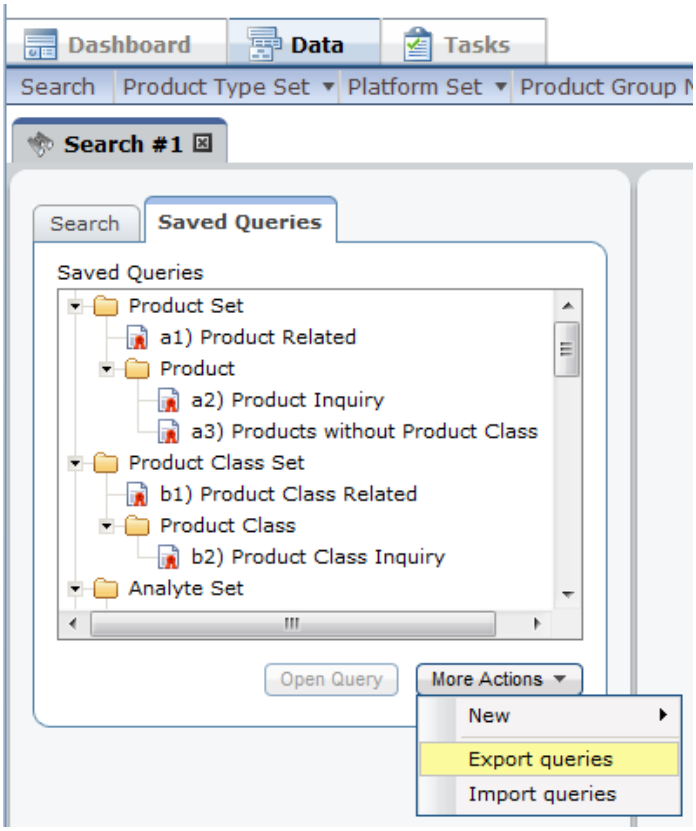
2. **post/idd_export** – This folder will contain a single / entire IDD application export. This is a zip archive with the same name as the application being exported. To export an app, sign into IDD Config, select an application, click the Export button then select Export Application. When prompted to for the file location navigate to the /post/idd_export folder.




IDD Config Screen / Export

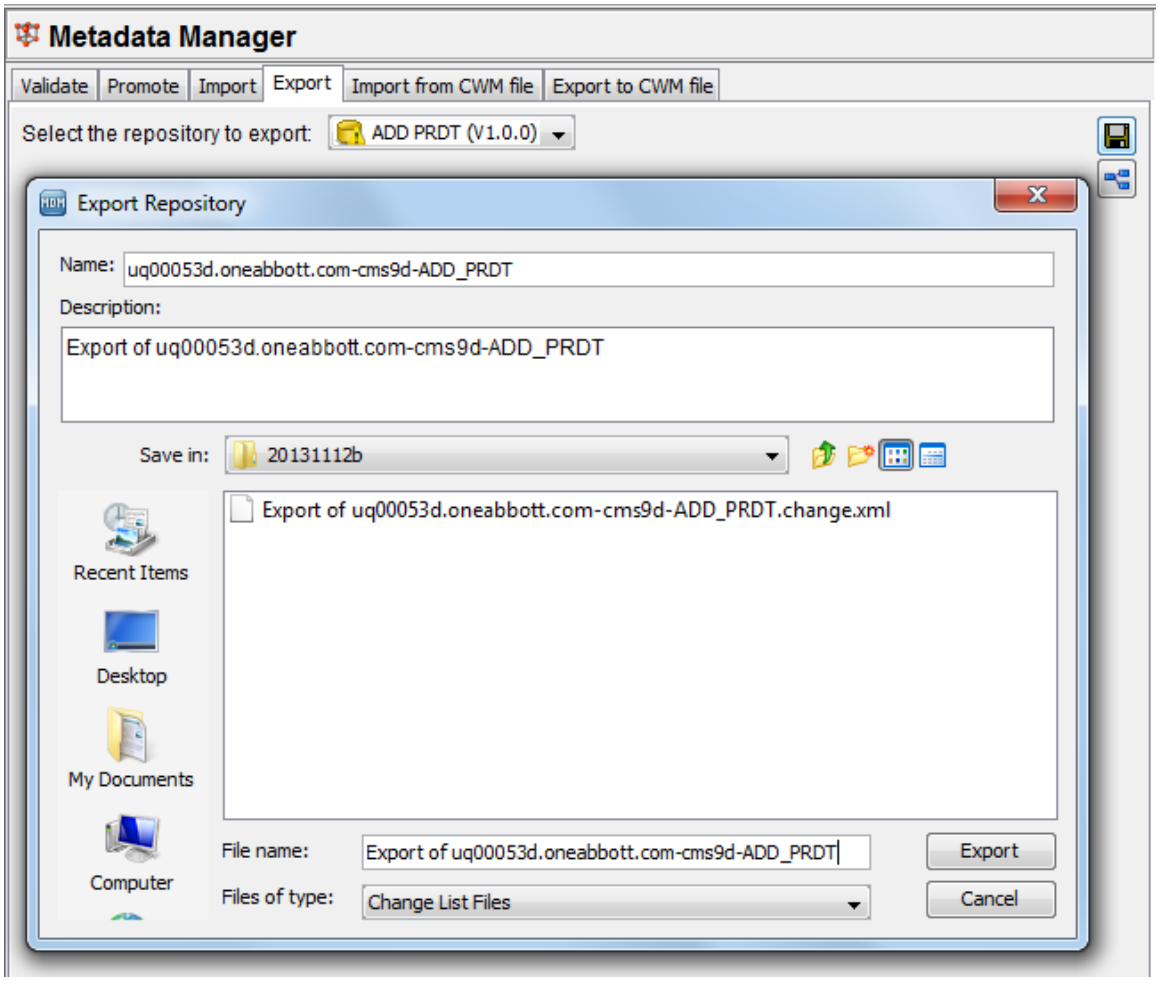
3. **post/public_queries** – This folder contains a single XML file that defines all of the public queries within a given IDD application. The file name is the name of the Release Candidate follow by "_queries.xml". This file is created by exporting the queries from within the IDD application. To

export this file, log into the source application as the Admin user, click on the Data Tab, click on the More Actions button then select Export Queries. When prompted to for the file location navigate to the /post/public_queries folder.




IDD App Export Public Queries

4. **post/met_export** – This folder contains a change list (XML) that defines all of the design objects within the ORS. This change list is the result of a metadata export using metadata manager in the Hub. The ORS must be in a valid state before it's metadata can be exported.




Hub: Workbenches > Metadata Manager > Export > Save

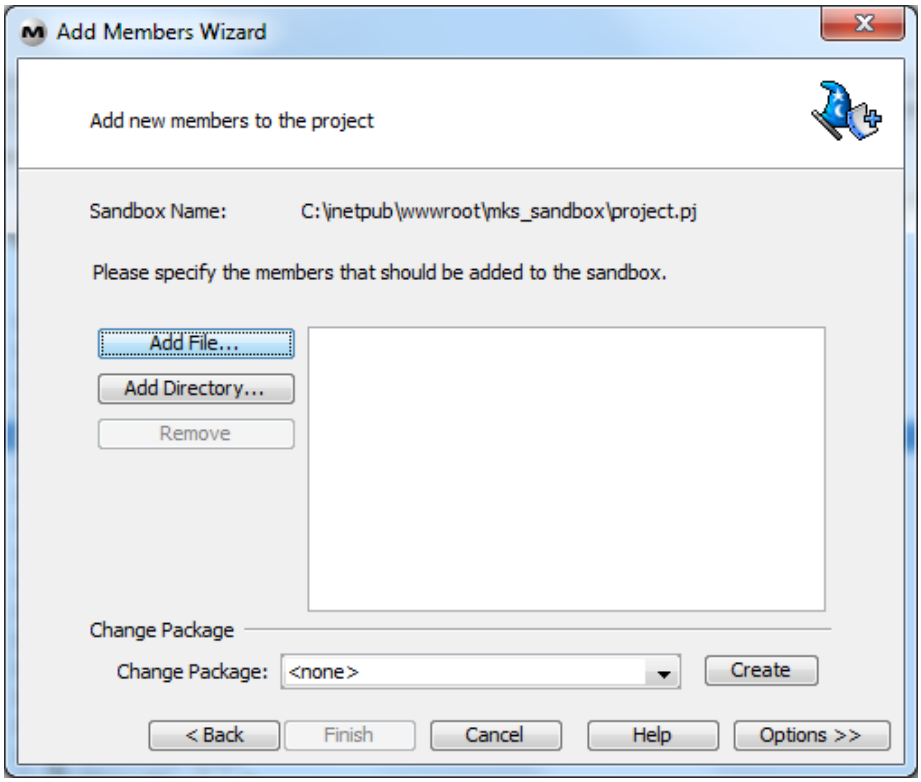
5. **post/java_classes** – The folder contains the Java source code / project(s) for one or more user exits. These files have the .java extension.

6. **post/all_scripts** – This folder contains all shell scripts and SQL scripts as they exist on the application server file system.

7. **post/autosys** – This folder contains the current JIL file. This file describes all of the jobs within the ADD_PMDSM project. A real-time version of this file is available here.

## MKS – Checking In

To check a Release Candidate into the MKS Integrity repository follow these steps; Copy the entire folder structure into your sandbox folder, log into MKS, Member > Add from the menu, Click Add Directory then navigate to the RC.

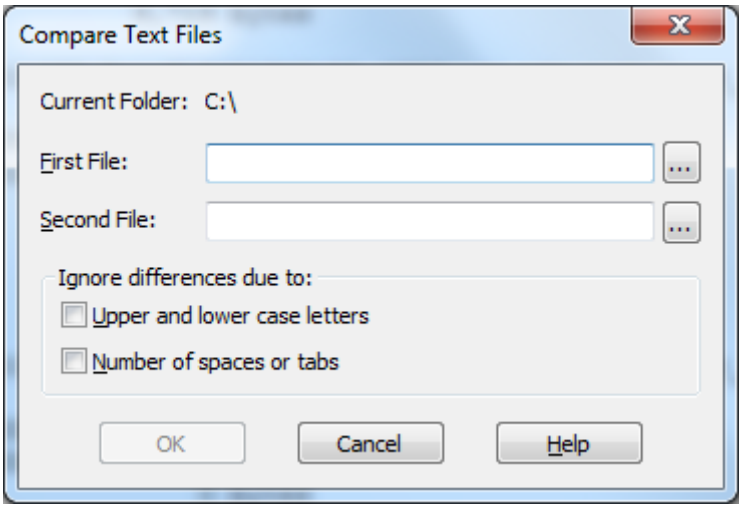



MKS member add dialog

Note: All Release Candidates are checked into the ADD/D030G / Dmdm_ORS_Change_List sub-project.

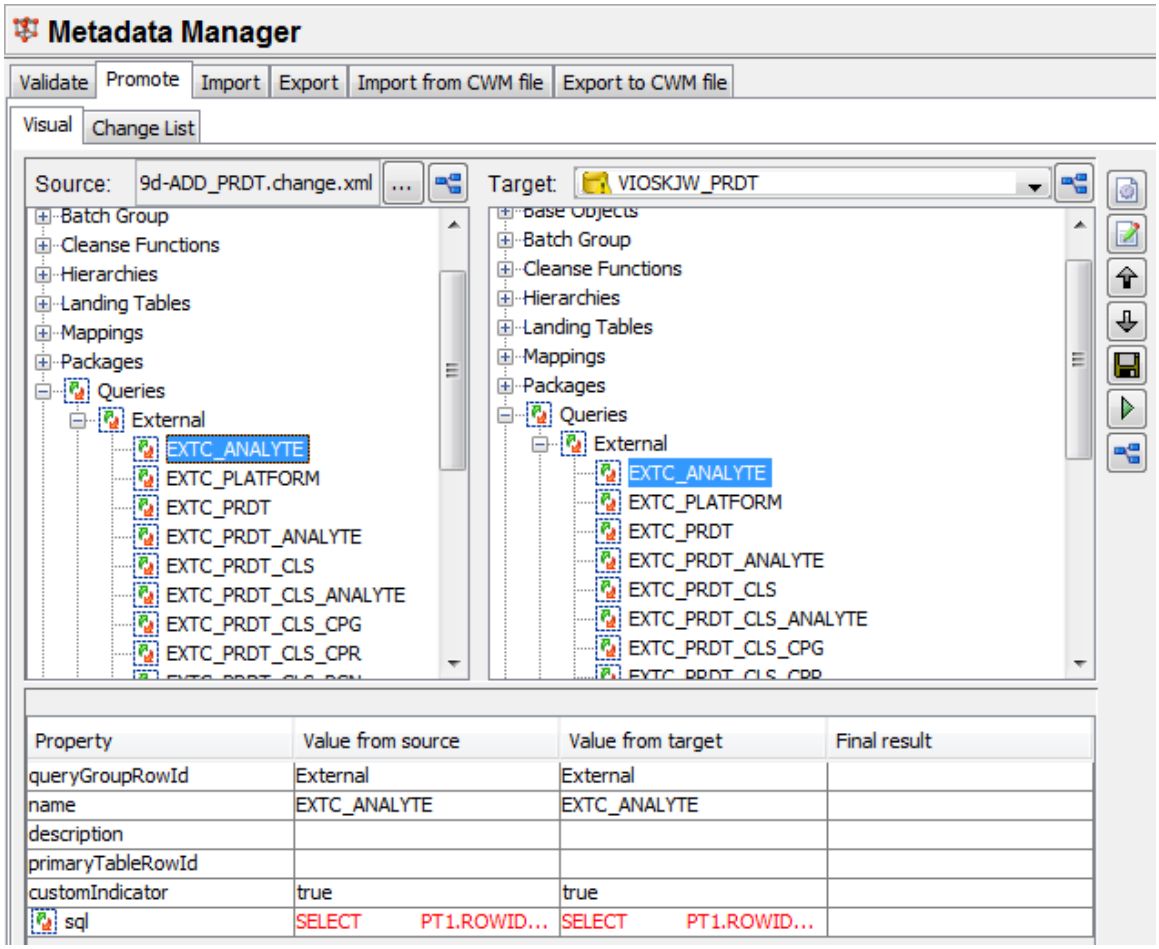
## *Version Comparison*

The differences between two separate Release Candidates can be found by performing a diff on the contents of any of the post folders. For example using Textpad as a comparison tool you could extract the differences between two different database schemas.



Textpad file Compare (Ctrl-F9)

The Hub Metadata manager is a tool to perform a visual node to node comparison between two similar ORSs.



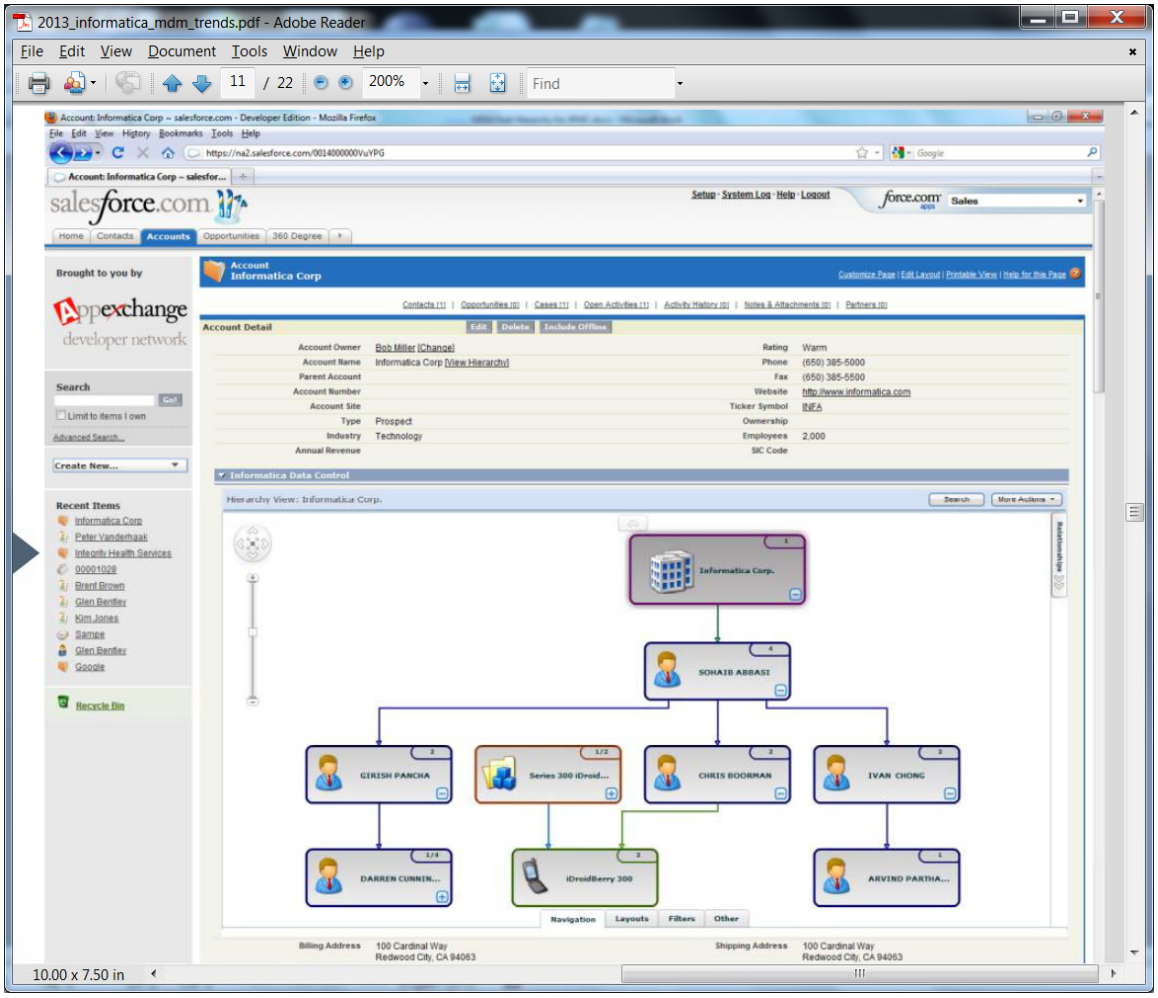

Hub Metadata Manager Visual Comparison

Informatica MDM

| Author | Scott Krause | Date | 01 Dec 2013 |
|---|---|---|---|

## Summary

Informatica Data Controls (IDC) is a licensed feature of the Informatica MDM Hub that allows user interface controls exposing MDM Hub data to be embedded in third-party applications. Using these controls, master data and the features of the MDM Hub can be made more easily available to business users within Salesforce.com.




Hierarchies embedded into the SFDC page

IDC controls are tightly bound to an IDD application. IDC controls provide generic functionality that becomes specific when bound to an IDD application configuration. From an IDD application configuration, an IDC control can use subject area definitions, layouts, search, cleanse and validation, customizations, data security, and localization.

The Informatica MDM Hub provides three built in control types:

- History View
- Hierarchy Manager
- Duplicate Prevention

**Implementation**

The IDC can be embedded into the AForce application(s) with page layout / trigger customization via VisualForce / Apex. The end result of which is a simple IFrame, or window within a window. Meaning that the IDC really still resides within the IDD application (ADDApp) but only appears integrated into the AForce UI.

What this means from an integration risk perspective is that issues like connectivity and performance can be anticipated to behave exactly as it does now within the IDD.

**Usability**

It is difficult to get a feel for exactly what the IDC user experience would be without a POC environment. However we can assume that without a compact minimal viewport the embedded HM canvas will become unwieldy at smaller resolutions (laptops). Another UX consideration is single sign-on. Currently MDM authenticates users via Active Directory but can be configured to authenticate using a vender provided SSO (Single Sign On) component specifically designed for SFDC.

**Licensing**

We are not currently licensed to use IDC. Unfortunately if we were to negotiate a new license key it would likely require deploying a new EAR file in the development or POC environment.

**Alternative - Bookmarks**

A loosely-bound alternative to IDC integration is deep linking (Bookmarks). Bookmarks are web addresses that point to specific content within the IDD. A third party application could generate a bookmark for a specific account calling the IDD, opening a new browser instance or frame showing that account displayed on the HM canvas.

| Pros | Cons |
|---|---|
| No additional licensing required | Disconnected user experience<br>Login require on first use<br>Displays entire IDD window |

A Bookmark can invoke one of the following application contexts and then display a given entity:

- Data View
- Hierarchy View
- Task View
- Search View

**Alternative – Web Service (Service Orientated Architecture)**

The Siperian framework (SIF API) can expose an ORS as a consumable web service. The IDD is simply a website that consumes a custom ORS web service. Third party consumers can also programmatically access the custom features of an ORS such as put packages and cleanse functions. In this scenario, a Salesforce Apex component would make asynchronous calls to the web service via AJAX. The interaction would be invisible to the end user. This gives the consuming application precise control over the MDM data so fields can be entered, validated and persisted into a base object(s) using the same methodology that the IDD uses.

| Pros | Cons |
|---|---|
| Seamless User Experience<br>ORS enforced validation and security<br>Precise control over data interchange | Tightly Bound API<br>Apex programming required |

**MDM Audit Posture**

Informatica MDM Hub creates and maintains several different history tables to provide detailed change tracking options, including merge and unmerge history, history of the pre-cleansed data, history of the base object, and the cross-reference history.

*Base Object*
Enabling history at the base object level will cause every change to be stored in one or more history tables. Changes can be the result of a load, merge, data steward update, put or task activity.

- Details such as the time of change, state before change, and the user / system responsible can be extracted from the history tables
- The history tables are not exposed to consuming applications

| History Table Suffix | Description |
|---|---|
| C_BO_[OBJECT NAME]_HIST | Tracks history of changes at the BO level |
| C_BO_[OBJECT NAME]_HXRF | Tracks history of changes on the XREF table |

*Staging*
Enabling Audit at the staging level will cause the raw (pre-stage) data to be retained for a specified number of cycles.

| History Table Suffix | Description |
|---|---|
| C_STG_[OBJECT NAME]_RAW | Tracks history of changes at the landing level |

*Retention*
The base object history data does not get cleared. It is recommended that this data be cleared regularly as operational maintenance.

*Performance*
Turning history on at the base object level will have a minor negative impact on overall system performance.