Informatica® Multidomain MDM
10.4

Data Director Implementation Guide

Informatica Multidomain MDM Data Director Implementation Guide
10.4
March 2020

Publication Date: 2020-03-19

Table of Contents

Preface

Use the Informatica® *Multidomain MDM Data Director Implementation Guide* to learn how to configure an application for Informatica Data Director that uses the subject area model. Learn about subject areas, implementation processes for applications, and manual Data Director configurations. For information about how to configure an application for Data Director with business entities, see the *Multidomain MDM Provisioning Tool Guide*.

Informatica Resources

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

Informatica Network

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

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

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

Informatica Knowledge Base

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

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

Informatica Documentation

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

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

Informatica Product Availability Matrices

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

Informatica Velocity

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

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

Informatica Marketplace

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

Informatica Global Customer Support

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

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

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

CHAPTER 1

Introduction

This chapter includes the following topics:

Overview

Attention: Data Director with subject areas is deprecated. Informatica recommends that you migrate to Data Director with business entities.

This guide describes how to create an application for Data Director with subject areas. For instructions on how to create an application for Data Director with business entities, see the *Multidomain MDM Provisioning Tool Guide* instead.

Data Director is a data governance application that enables master data solutions that are effective for every stakeholder in the data governance equation such as:

- Business users
- Data stewards
- IT managers

Data Director enables business users to effectively perform the functions described in the following table:

Functionality	Description
Create	Create high quality master data, working individually or collaboratively across your business.
Manage	Manage duplicates, resolve matches, approve and manage updates to your master data, create and assign tasks to data users.
Consume	Search all master data from a central location, view master data details.
Monitor	Track lineage and history, audit master data for compliance, customize your dashboard.

Prerequisites

This document requires familiarity with the Multidomain MDM architecture and an understanding of all the components in your environment that are used by Data Director applications.

For more information, see the Multidomain MDM product documentation.

CHAPTER 2

IDD Concepts

This chapter includes the following topics:

IDD Application

An IDD application is the main configuration and deployment unit for IDD implementations. An IDD application is what business users see when they launch and log in to IDD.

IDD Configuration Manager

The IDD Configuration Manager is a web-based utility used to add, modify and manage IDD applications.

Related Topics:

- "IDD Configuration Manager" on page 36

IDD Configuration Files

An IDD application consists of a collection of configuration files: an IDD configuration file (XML), resource bundles, internationalization message bundles, online help, and other auxiliary files. You can load or modify IDD applications in the IDD Configuration Manager, or export and edit them manually.

Related Topics:

- "Application Components" on page 104

Provisioning Tool

You can use the Provisioning tool to define business entity models, tasks, and transformations, and design the user interface for Data Director.

Informatica Data Director requires a business entity configuration for features based on the Entity 360 framework, such as the Task Manager and record views. Informatica Data Director also requires a subject area configuration for features such as the Hierarchy view, the Cross-reference view, and the Match Merge Comparison page.

This guide describes subject area configuration for Informatica Data Director. For information about Entity 360 Framework configuration and business entity configuration, see the *Multidomain MDM Provisioning Tool Guide*.

Subject Areas and Subject Area Groups

In an IDD application, data is organized around subject areas and aggregated into subject area groups.

Subject Areas

The *subject area* is a core organizing concept for an Informatica Data Director application.

Other terms or concepts that are related to, or similar to, the subject area include: business object, and hierarchical entity. Informatica Data Director uses the subject area definition to determine how to treat each foreign key relationship in an Operational Reference Store (ORS).

The Hub Store maintains detailed metadata about tables and relationships defined in an ORS. This metadata includes relationships between base object tables that can represent:

- References to lookup tables
- Links between a parent and related child data
- Associative links between tables - not representing an ownership relationship.

The Hub Store provides some of the metadata that allows Informatica Data Director to understand how relationships should be treated. For example, the base object lookup indicator tells Informatica Data Director when to treat a related table as a lookup with a pre-populated dropdown list that users see in an Informatica Data Director application.

For other relationships, an Informatica Data Director application might require additional information to correctly understand the relationships - whether they should be interpreted as relationships among tables in a subject area, or relationships between subject areas. The Informatica Data Director Configuration Manager is used to specify this additional relationship information for Informatica Data Director applications. You cannot use aliases for a subject area that is based on a Hierarchy Manager relationship.

A subject area represents a collection of data that should be treated as a unit from a business perspective. A subject area has:

- A single root record in a base object
- Some number of child and grandchild records (through one:many and many:many relationships).

Subject Area Groups

A *subject area group* is a set of one or more subject areas that have the same base object at their root (also called the *primary object*).

For example, an ORS using a Party model (a single base object representing different entity types) will have a subject area group with multiple subject areas.

Note: A base object can only be associated with one subject area group.

Relationships within Subject Areas

In an IDD application, relationships within subject areas are based on the relationships that are configured between base objects in the Hub Store (using the Schema Manager in the Hub Console).

The IDD Configuration Manager refers to the configured *match path components*, which are based on foreign key relationships.

One:Many Child Relationships

For one:many relationships, the child record has a direct foreign key to the primary object. IDD supports two kinds of one:many relationships.

The following table describes the types of one:many child relationships:

Relationship	Description
One:Many	The list of child records will be displayed in a tab below the primary data.
Logical One:One	The expectation is that there should be only one child record for each primary object. The data is shown in the form with the primary object. If there is more than one child (due, for example, to two primary object records being merged), the IDD application provides a means to resolve this.

Many:Many Child Relationships

For many:many relationships, the child record is related to the primary object through a relationship table.

The relationship table in a many-to-many child must contain two foreign keys.

IDD supports two types of many:many relationships. The following table describes the types of Many:Many child relationships:

Relationship	Description
Part Of	The child record belongs to the primary object. No other subject area should reference this child. When adding a child, both the relationship and child records are added. When editing a child, if another subject area references it, a copy of the child is created. The data referenced by the other child remains unchanged.
Reference	The child is another subject area. When adding a child, only a relationship record is added. The IDD application user must search for the subject area child to be related. To edit the child data, the subject area for that child must be opened. This child can be linked through a standard relationship Base Object or an HM relationship Base Object.

One:Many Grandchild Relationships

For one:many relationships, the grandchild record has a direct foreign key to a child object. IDD supports two kinds of one:many relationships: if the child is a many:many, the foreign key can be either to (see the data model examples below):

- The child relationship
- To the relationships record

Relationship	Description
One:Many	The list of grandchild records will be displayed in a tab below the child data.

Many:Many Grandchild Relationships

For many:many relationships, the grandchild record is related to a child object through a relationship table.

IDD supports two kinds of many:many relationships: If the child is a many:many, the foreign key can be either to the (see the data model examples below):

- Child record
- Relationships record.

The following table describes the types of many:many grandchild relationships:

Relationship	Description
Part Of	The grandchild record belongs to the primary object - no other subject area should reference this grandchild. When adding a grandchild, both the relationship and grandchild records are added. When editing a grandchild, if another subject area references it, a copy of the grandchild is created. The data referenced by the other child remains unchanged.
Reference	The grandchild is another subject area. When adding a grandchild, only a relationship record is added. The IDD application user must search for the subject area grandchild to be referenced. To edit the grandchild data, the subject area for that grandchild must be opened. This grandchild can be linked through a standard relationship Base Object or an HM relationship Base Object.

Note: When configuring the match path for grandchildren in the Schema Manager of the Hub Console, ensure that **Check Missing Child** is disabled. IDD application do not function properly if **Check Missing Child** is enabled.

Sibling References

A sibling reference is a relationship from one record in a subject area to a child record within that subject area.

For a data model, a customer could include both address and phone number child records, with the phone number having a foreign key to associate it with a specific address. IDD can be configured to support this kind of relationship.

When adding or editing the address key on the phone number, the IDD application user is given a list of addresses that contains only children of this party.

Related Topics:

- "Manual IDD Configuration" on page 56

Parent Records

A record that is a parent of the primary object can be included in the subject area.

It is shown in a child tab. Because there is always just one record in this tab, it is always shown in a form view. This data is read-only. IDD does not allow editing of this data or of the relationship to this data.

Informatica MDM Hub Feature Usage

Services Integration Framework

All interaction between a Data Director application and an ORS is through Services Integration Framework (SIF) API calls.

There is no direct access to the ORS database (with one exception: charts can be configured to use an application server datasource to fetch report data). The IDD Configuration Manager uses SIF to access metadata about an ORS, but it uses a datasource to directly access the CMX_SYSTEM.C_REPOS_DS_CONFIG table.

Some of the SIF API calls are asynchronous. To enable support for asynchronous SIF calls, row-level locking must be enabled for the ORS that the Data Director application uses. For more information, see the section on row-level locking in the *Multidomain MDM Configuration Guide*.

Using a Web Server

Before you implement a web server that acts as a reverse proxy, configure the service URL format that IDD generates for SIF calls. Configure the 'referer.url' property in the `cmxserver.properties` file to specify the service URL format.

Add the following text to the `cmxserver.properties` file to configure the format of the service URL:

```
referer.url=http://<local host>:<port number>
```

User Authentication (SSO)

By default, Data Director authenticates users with a SIF call to the Hub Server. For the authentication process, the MDM Hub implementation requires that you configure users to the Master Database. For more information about configuring MDM Hub users, see the *Multidomain MDM Security Guide*.

Alternatively, you can implement single sign-on (SSO) and authenticate users against with external identity providers. The Login Provider in Data Director communicates with the Hub Security Provider (Login Module). For more information about provider files, see the *Multidomain MDM Security Guide*.

Base Objects

Column-level security is configured in Security Access Manager (SAM) by defining role-based access to base objects and their columns, which provides for fine-grained control over user access to data.

IDD references base objects directly for all `GET` and `PUT` operations. IDD uses packages only for displaying search results.

Caches and the Clear Cache Option

Informatica Data Director maintains a cache of MDM Hub metadata that describes base objects, columns, relationships, and other details. If you change the MDM Hub metadata, click **Clear Cache** in the IDD Configuration Manager before you export an application from IDD.

The Clear Cache option in the IDD Configuration Manager clears the cache for the selected IDD application. In a Microsoft SQL Server environment, Informatica recommends that you clear the cache when you make changes to the ORS metadata through the Hub Console. For example, if you add a relationship to a base object in the Hub Console and then save and validate the change, you can redeploy the IDD application for the change to take effect. However, you must clear the cache before you export the IDD application to see the new relationship in the Metadatabundle.properties file.

You can also restart the application server to clear the cache.

IDD also maintains caches of SAM role definitions and assignments and lookup values. IDD refreshes the caches at a rate that you can configure through IDD global properties.

Match Paths

You define child relationships in IDD with match paths. You can configure match paths in the Schema Manager in the Hub Console.

Before the introduction of IDD, match paths had been used strictly to define match columns and match rules. The match path definition works equally well to define child relationships in IDD.

To add a child to a subject area, you must create a new match path for that child if one does not exist. When you create a match path, it must be based on ROWID_OBJECT.

Match paths can also be used to enable search on related tables that are not part of a subject area. For example, suppose you have a Party that is related to a Product. Product would not be part of the Party subject area. However, a match path can be defined from Party to Product. Using this match path, an IDD application user could search for a Party based on the attributes of a related Product.

Search

Searching for data in a subject area can be based on either of the following SIF search APIs: searchQuery and searchMatch.

In both cases, a display package is used to show the search results.

Basic - SQL-based Search

Basic Search uses the searchQuery API.

A search can be based on data in the:

- Primary object record
- Any of its (PO) child records
- Any records related through a match path component

You can perform a case insensitive Basic search, when you run a data query. Basic search finds results using string and string pattern comparisons.

Extended - Match-based Search

Extended Search is not case sensitive and uses the **searchMatch** API with matchType=NONE.

It is intended for searching, and therefore does not use a predefined match rule set. Any of the data in the subject area that contributes to a match column can be used as search criteria. An IDD application requires users to enter criteria into the fuzzy match key before they can run the search.

Advanced Search

Advanced search allows IDD application users to construct complex queries by defining SQL WHERE-type expressions and free form query text.

You can perform a case insensitive Advanced search, when you run a data query. Advanced search allows IDD application users to specify search conditions that go beyond the capabilities available in Basic or Extended searches.

Cleanse Functions

IDD uses the **PUT** API rather than **cleansePut**.

However, IDD can call the **cleanse** API for each base object record before it is saved. This is sometimes referred to as an *inline cleanse function*. The cleanse function can perform regular data cleansing and standardization, and also custom validations on the data. Each configured cleanse function is called before data is saved.

- In the Data View, cleanse is invoked when the **Apply** button is clicked on an edit form.
- In the Hierarchy View, cleanse is invoked when the **OK** button is clicked on a relationship add/edit dialog.

Cleansing and Standardization

The IDD Configuration Manager provides a straightforward means for connecting base object records to the inputs and outputs of a cleanse function.

The data in the base object record is updated with the outputs from the cleanse function.

Note: Only those base object columns that are selected in the layout for the subject area configuration can serve as cleanse function inputs or outputs.

Validation

A cleanse function can be used to perform custom data validation.

Validation results are processed if the cleanse function has a validationStatus output parameter.

- If the validationStatus parameter is blank, there are no validation errors and the process can continue.
- If there are validation errors, the validationStatus parameter will include a series of validation messages describing the inputParameter name and a message. In the IDD application UI, each validation error is associated with an input value in a specific input column.

Note: The Resource Kit contains the ValidationCleanseLib sample, which provides an example of a cleanse library with functions that perform validation in an IDD application.

Cleanse Functions Returning NULL

When the output of a cleanse function is a null value, the **cleanse** API returns no information about that field.

It is assumed that the function is not changing that field. If the goal is for the cleanse function to override a value with NULL the options depend on the data type, then following is required:

- String - The function can be changed to return an empty string.
- Date or numeric - A user exit must be implemented to modify the data. The beforeEverything() or beforeSave() methods of the Save handler can be used.

Related Topics:

- "User Exits" on page 78

Trust

An Data Director application is configured to use a single source system for all of its operations.

Data entered and updated through an Data Director application follows all of the standard trust rules, as described in the Admin Console online help or the *Multidomain MDM Configuration Guide*. The data entered in an Data Director application is applied to the base object record based on the trust and validation rules configured in the Informatica MDM Hub for that source system. When you view cross reference data, you can promote the value of an attribute from a cross reference record for trust-enabled columns. This results in a trust override for that attribute.

Workflows and Tasks

An IDD application can use workflows and tasks to support a change-approval process for state-enabled records in the Hub Store.

For example, consider a case where a Finance Manager wants to review all changes to client banking information before the change can be accepted as master data. You can configure an IDD application so that, when someone in the Finance department uses the application to update information, the Finance Manager is automatically assigned a task to review the pending change and approve or reject it. A change-approval process ensures that only approved records contribute to the Best Version of the Truth (BVT) records.

An IDD application coordinates task activities among the IDD Task Inbox, a business process managment (BPM) tool, and state-enabled tables in the Hub Store. To include workflow support in your application, see "Step 7. Configure MDM Workflows" on page 32.

Tasks and Actions

A *task* is a step in a workflow process.

For any task, there will be one or more *actions* that can be performed. Tasks and their associated actions can be configured as part of an IDD application.

In-Flight Data

In-flight data is business data that can go through different states (ACTIVE, PENDING, or DELETED) while progressing through a workflow.

IDD provides support for in-flight data using the Informatica MDM Hub state management functionality and task management features.

Data can be added or updated and 'Submitted for Approval' rather than saved. The data changes are stored as PENDING changes - the data is not applied to the base object. A task is created for another user to approve this change. Once approved, the PENDING data is promoted to ACTIVE, and the data is then applied to the base object.

Hierarchy Manager

If Hierarchy Manager (HM) is configured for an ORS, you can configure an IDD application to work with this configuration.

Configure the IDD application according to the following rules:

- Any HM entity that an IDD application uses must be configured as a subject area in the IDD Configuration Manager. HM is used to model the relationships between subject areas.
- An IDD application operates against a single HM configuration (profile/sandbox combination). IDD uses the SAM access control configuration, rather than different HM configurations, to manage user access control. The HM configuration used by an IDD application must include all HM entity and relationship types to be used in the IDD application.

Security Access Manager

Use the Security Access Manager to grant user roles granular access to the base objects and other resources. Data Director inherits the user roles and implements the same granular access to records.

For more information, see the *Multidomain MDM Security Guide*.

Related Topics:

- "IDD Security Configuration" on page 105

Object and Column Security

SAM provides role-based security privileges on design objects and columns defined in an ORS.

An IDD application uses this security configuration so that the data that is shown, and the operations that are available to an individual user, depend on the role(s) assigned to that user account. IDD application users see only the data and functionality to which they have been granted access. For example, if a user does not have READ access to the HISTORY table for a base object, in the IDD application, the History command for that subject area is not available to the user.

Note: A Hub user with Administrator access (configured in the Users tool in the Hub Console) is a super-user for IDD and has full privileges on all objects.

Data Security

SAM does not provide row-level data security (restricting users from seeing certain records based on the contents of those records).

IDD, however, provides a simple data security mechanism. For each subject area, *security filters* can be defined in the IDD configuration file. A security filter specifies a filter condition that IDD applies to any data accessed by users assigned to a specific role. For example, a security filter can specify `COUNTRY_CODE = 'US'`, which can apply to users with the US Data Steward role. Each filter can apply to multiple roles. Any number of filters can be created for a subject area for any number of roles.

Data Masking

IDD provides a mechanism for hiding (masking) information based on security roles.

You can define a mask for each field in a column layout. Mask can be specified for a single role, for a set of roles or for all non- admin users. When you specify a mask, all or part of the value is replaced with asterisk (*).

Related Topics:

- "Data Masking" on page 129

History

IDD provides a subject area view of the history of changes for each record.

This functionality requires that history be enabled on the base object. If history is not enabled for a base object, the History View is not available for the associated subject area in the IDD application. IDD shows a timeline view of events for the record and its child records. A point-in-time view of the data can also be shown.

Lookup Tables

A lookup table, also called a lookup or lookup base object, is a table that stores a list of predefined values. Data Director (IDD) queries the lookup table to retrieve a value based on the input source value and the lookup condition. Data Director then populates a drop-down list of values in the application. For example, if you enter a value in a Country field, the application lists the countries stored in the LU_COUNTRY lookup base object table.

You can define lookup values in the following ways:

- In a physical lookup base object table with a foreign key between the base object and the lookup base object. Data Director uses metadata about this foreign key to populate the lookup values.
- In a physical lookup base object table with no foreign key between the base object and the lookup base object. The IDD configuration describes the foreign key relationship, which populates the lookup values.
- In a static list of values in the IDD configuration.

For the lookups defined in a physical table, the LOOKUP_IND column in C_REPOS_TABLE indicates whether the table contains lookup values or regular data. You enable the lookup indicator through the Schema tool in the Hub Console. By default, the lookup indicator is disabled when you create a base object. When you enable the lookup indicator, the MDM Hub considers the base object as a lookup. For more information about the Schema tool, see the *Multidomain MDM Configuration Guide*.

Note: When you create a lookup, use a unique display name. Data Director cannot distinguish different lookup tables that share the same display name.

When Data Director recognizes that a column has a foreign key to another table, Data Director determines whether the related table is a lookup table. If the related table is a lookup table, Data Director creates a drop-down list in the application for that column, populated with values from the lookup table. The column in the lookup table that is used depends on the **Lookup Display Name** field configured for the relationship in the Schema tool.

Related Topics:

- "Lookup Column" on page 59

Dependent Lookups

A dependent lookup is a lookup table that depends on another lookup table.

A typical example of a dependent lookup table is a type lookup table and a subtype lookup table. The list of values that appears in the subtype field depends on the selected value in the type field in IDD. For example, if you selected United States in a Country field, when you enter a value in a State field, IDD lists U.S. states stored in the LU_STATE dependent lookup.

Timeline

Timeline lets you view and manage data change events of business entities and their relationships. You can define the data change events or versions of business entities and their relationships in terms of their effective periods.

Data changes occur over time and are independent of their relationship to other data. The changes to data result in a new period of effectiveness or an updated period of effectiveness for the past, present, or future. The timeline feature lets you track these changes to data over a period of time.

For example, John Smith lived in Los Angeles effective from 31 January 2008 to 20 October 2010. He now lives in San Francisco effective from 21 October 2010. He will live in Las Vegas effective from 25 November 2014. Use timelines to track past, present, and future changes to data such as the address data of John Smith.

Note: You can specify effective period in the date format. The system uses the database time locale for dates.

Timeline capabilities provide a two dimensional visibility into data, based on the period of effectiveness and history. The period of effectiveness for a record is defined by the effective start date and the effective end date of a base object record. History is a date from the history of a record for which you need to view the value. You can manage data events of business entities, such as customer address, phone number, and their relationships, by enabling timeline for relevant base objects. To enable the timeline for a child base object, you must first enable the timeline for the parent base object. The MDM Hub uses the cross-reference (XREF) tables that are associated with the timeline-enabled base objects to maintain the effective periods for the base object records.

Note: You must enable timeline for each base object in the Hub Console, except for the hierarchy-enabled child relationship base object.

For more information, see the *Multidomain MDM Configuration Guide*.

Timeline Rules

When you define and maintain timeline information, the MDM Hub enforces timeline rules.

You require an understanding of the rules that the MDM Hub enforces to manage timelines of business entities and relationships. At any point in time, the MDM Hub considers only one version of a record to be

effective, based on effective start and effective end dates. When you use batch processes, Services Integration Framework, or Data Director to change data, the MDM Hub persists the current effective data. Also, when many systems contribute to a base object record, the MDM Hub enforces rules to update the version of the record, based on the contributing effective records.

You can also use user exits to define and enforce custom rules to manage timelines and effective dates.

For more information, see the *Multidomain MDM Configuration Guide*.

Bookmarks

Bookmarks are URLs that open an IDD application and shows a view, task, or search.

Note: Bookmarks are available for IDD applications that use the subject area model.

The URL specifies which IDD application to invoke, which part of the application to open, and which entity to display. Bookmarks can be used to invoke IDD from an external application (for example, an Informatica MDM Data Control, or IDC), or from a browser. Users can share a bookmark URL with another user. When the user opens the URL in a browser, the user must successfully log in to the IDD application to see the view.

Within an IDD application, you can link to Show Bookmark commands on the pages. These commands provide the URL link for the current entity. Bookmarks are available for the following functionality: Data View, Hierarchy View, tasks, and searches.

The format for the URL is:

```
http://<host>[:<port>]/bdd/?deeplink=<operation>;<iddAppName>/
<subjectAreaID>;<param1>[;<param2>]
```

Where:

Variable	Description
host	Name of the machine that hosts the Informatica MDM Hub.
port	Optional. Port number.
operation	One of the following values: - openrecord;dv - opens an entity in the Data View - openrecord;hm - opens an entity in the Hierarchy View - opentask - opens a task window - search - opens a search window
iddAppName	Name of the IDD application.
subjectAreaID	Identifies the subject area. Uses the following format: `subjectAreaGroupName/SubjectAreaName`
param1	Defines what data to display and is dependent on the operation.
param2	Optional. Operation dependent.

Note: Any character that is not allowed in a URL must be double encoded. Double encoding (running the encoding process twice) is required to allow web servers to accept requests containing slashes (“/” and “\”)

in its parameters. Requests containing single-encoded slashes used in parameters are rejected by web servers. Only the parameter values should be double encoded.

Data View

The *openrecord;dv* operation is used to open a data view.

The *subjectAreaID* identifies the subject area, and *param1* identifies the record. As with SIF APIs, a record can be identified either by rowid or by system name and source key. When using source key, be sure to include any leading or trailing spaces in the value.

Additionally, *param2* can be used to specify *xref*, *history*, *duplicates* to open the Data View with the **Cross References**, **History**, or **Find Duplicates** screens.

Examples:

```
http://<host>[:<port>]/bdd/?deeplink=openrecord;dv;test/Customer;rowid:268
http://<host>[:<port>]/bdd/?deeplink=openrecord;dv;test/Customer;
systemName:SFA,sourceKey:CST1160
http://<host>[:<port>]/bdd/?deeplink=openrecord;dv;test/Customer;rowid:268;xref
```

Merged records are a special case. If you merge a record with another record, the merged record has the rowid of the surviving record. You can, however, continue to use a bookmark URL that references the non-surviving rowid. In this case, the URL is redirected to the merged record rowid. For example, let's say that you merge two records with rowids 1 and 2, and the merged record has the rowid 1. If you use a bookmark URL and specify rowid 2, the link is redirected and retrieves the merged record with rowid 1.

Hierarchy View

The *openrecord;hm* operation is used to open a Hierarchy View.

The *subjectAreaID* identifies the subject area, and *param1* identifies the record. The usage of these parameters is the same as with the Data View parameters.

Examples:

```
http://<host>[:<port>]/bdd/?deeplink=openrecord;hm;test/Customer;rowid:268
http://<host>[:<port>]/bdd/?deeplink=openrecord;hm;test/Customer;
systemName:SFA,sourceKey:CST1160
```

Task

The *opentask* operation is used to open a task.

The *subjectAreaID* identifies the subject area, and *param1* identifies the task - this is simply the value of ROWID_TASK for the task.

Example:

```
http://<host>[:<port>]/bdd/?deeplink=opentask;test/Customer;3162
```

Search

The *search* operation is used to open a search tab and execute a search.

The *subjectAreaID* identifies the subject area, and *param1* defines the fields and values on the search form. Use the Show Bookmark command to see examples of *param1*.

CHAPTER 3

Implementation Process

This chapter includes the following topics:

Implementation Process Overview

This section describes the recommended high-level process for configuring IDD applications.

This process should be used as a template for creating IDD implementation plans. The main goal is to outline the steps in the build/test cycle that would provide an efficient model for rapid IDD development. Such an approach allows you to use the intermediate stages of the configuration process for getting additional feedback and validating requirements with the customer.

Before You Begin

This section assumes the following prerequisites:

- Informatica MDM Hub, cleanse adapters, and Process Servers are already configured and operational in your environment. For more information, see the *Multidomain MDM Installation Guide*.
- The ORS schemas are configured and contain some test data. IDD application configuration requires the use of both the IDD Configuration Manager and the Hub Console. The Hub Console is used to create the necessary configuration elements in the target ORS (such as base objects, packages, lookups, match path components, and so on).
- All base objects (and associated metadata) required for an IDD application need to be configured as SECURE in the Secure Resources tool in the Hub Console.
- Configuration and initial testing should be done using an MDM Hub user account with unrestricted privileges for the target ORS schemas. You can either use the admin account or any other account that is configured with all privileges to the ALL_GLOBAL_RESOURCES group.

  Note: ALL_GLOBAL_RESOURCES does not include the custom resources added as part of the IDD application - those must be configured individually.

- The analysis and data modeling for defining the subject areas and the business rules has been completed.
- If you want to support workflows, in the MDM Hub you must enable state management on the target base object tables and decide which BPM tool you want to use for your workflow engine. You may need to complete integration steps for standalone BPM tools. For more information, see the *Multidomain MDM Configuration Guide*.
- Other areas of the Hub Store need to be configured:
  - Security
  - Cleanse functions (if they will be used to check IDD user-entered data in an IDD application)
  - Hierarchy Manager (if it will be used in an IDD application).

    Note: If you enable state management on any Hierarchy Manager entity or relationship tables, you must enable the feature on all of them.

For more information about Hub Console tools, see the Admin Console online help or the *Multidomain MDM Configuration Guide*.

Configuration Process

Follow the configuration process to make configuration changes to Informatica Data Director.

The configuration process is iterative process that is not a linear or one-time procedure. You can manage most of the IDD application configuration directly in the Informatica Data Director Configuration Manager. Some steps in the configuration process require manual editing of the IDD application components.

If you changed the metadata in the Operational Reference Store, click **Clear Cache** to get the latest MDM Hub metadata.

Note: Do not deploy IDD while an MDM Hub Load batch job runs, or while another user is making changes in the MDM Hub Console. If you deploy IDD during these MDM Hub activities, IDD generates Operational Reference Store validation errors.

Related Topics:

- "IDD Configuration Manager" on page 36

Step 1. Create the IDD Application

Create the IDD application in the IDD Configuration Manager.

1. For IDD instances that span multiple ORS databases that is you can create subject areas from different ORSes, but the subject area child for a subject area should be from the same ORS; create the individual subject areas for each ORS separately (in separate IDD applications).
2. Export the configuration.
3. Integrate the individual XML configuration files by merging them to create a multi-ORS IDD instance.

Consider the following configuration issues:

Consideration	Description
Application Source System	The most important property defined at the IDD application level is the source system that an IDD application uses for tracking updates made within the IDD application itself (such as edits made by IDD application users in a Data View). By default, the Admin system is used. Using the Systems and Trust tool in the Hub Console you can create an application source system. To configure the trust on BO columns for another system, you must create a dummy staging table and map it to the IDD source system. Regardless of which IDD application source system you use, it needs to be configured to have the highest level of trust to guarantee that the changes applied by IDD application users override any other contributing values and end up in the BVT (master record). If this is not the case, the results of an update will be very confusing to IDD application users.
HM Configuration	If you are planning to use the IDD HM functionality, you need to define the HM profile (using the Hierarchies tool in the Hub Console) that will be used for configuring the IDD Hierarchy Manager functionality. The HM configuration needs to be specified up front to ensure that the subject area definitions are consistent with the HM Entity definitions.

Step 2. Configure Subject Area Groups

Configure subject area groups.

- Use the IDD Configuration Manager to create any necessary subject area groups.

  For example, you might create a Customer subject area group to contain two subject areas: Person and Organization.

Step 3. Configure Subject Areas

Configure subject areas.

- If the subject area group contains multiple subject areas, identify the data attribute of the subject area root object that will be used to differentiate the subject areas.

  For example, a party_type attribute would distinguish party entities by type.

Step 3.1. Configure Subject Areas in the Hub Console

Configure subject areas in the Hub Console.

1. In the Schema Manager, review the match path components that are configured for the root object of the subject area and verify that there are match paths for each of the child objects that need to be included in the subject area and for related objects that need to be used in searches.
2. In the Packages tool, create the search display package that will be used to display the search results for the subject area. This is a package with the subject area root object as its primary table.

3. In the Schema Manager, check subject area lookup dependencies.

Lookup Mechanism	Description
Code Lookup tables	Code lookup tables must have the Lookup Indicator set to TRUE (checked) in the base object properties in the Schema Manager.
Entity Lookups	Entity lookups can be specified only to entities that are configured as subject areas. This can introduce complex dependencies between the subject areas. As part of the iterative development of an IDD application, you can exclude entity lookups from the initial IDD configuration if there are dependencies on the other subject areas that have not been configured. The lookup fields can be added after all subject area dependencies are satisfied.

Step 3.2. Configure Subject Areas in the IDD Configuration Manager

Configure subject areas in the IDD Configuration Manager.

1. Create the basic subject area configuration and test it by validating and deploying the application.

   This configuration includes setting up the layout (columns to display with the field type and size for each - this is the minimum that must be configured), match settings used for duplicate checks, configuration of any cleanse functions to use for checking data entered by IDD application users (used for data cleanse and/or validation), configuration of the label for the subject area, and subject area task assignments.

2. Add the children and grandchildren to the subject area.

   All children and grandchildren need to have a properly-configured match path to the root object of the subject area (configured in the Match/Merge Setup Details panel in the Schema Manager). When creating a new child, the IDD Configuration Manager displays the names of the match path components rather than the names of child objects.

   Only match patch components that are relevant for the child type are shown. This configuration includes setting up the layout (columns to display with the field type and size for each) and configuration of a cleanse function (optional) to apply to the record (used for cleanse and/or validation).

Tip About Adding Children and Grandchildren

To simplify troubleshooting for issues with child and grandchild configuration, consider adding these one at a time, and then deploying/testing the configuration after each is added (before adding the next one) to isolate any configuration issues that might arise incrementally.

Layout Configuration

Configuration of the layout is used to:

- Specify which fields to show from the base object.
- Specify the number of columns for form layouts.
- Specify the date and time format.
- Specify the UI field size for all fields (small, medium, large).
- Specify the required fields - which ones are not allowed to have NULL value (this is configured in the IDD configuration file).
- Specify which fields to be shown as hyperlink.

  Note: Only column data type of String that is defined in the Hub Console can be marked with **Show As Hyperlink** in the IDD configuration manager. Only fields with valid URL or email address will be parsed as a hyperlink.

Step 3.3. Validate, Deploy, and Test the Changes

In the IDD application, validate, deploy, and test the changes.

1. Create a query for a new search.
2. Verify that all appropriate attributes are available (attributes defined in the layouts of the root and child objects).
3. Add a new entity (record) to a subject area.
    a. Validate that all children can be created and that all fields display in the expected order.
    b. Validate that all lookup fields display correctly and have the correct lists of values. If fields do not display the lookup controls, you need to adjust the Lookup field configuration (set the Lookup Indicator to TRUE in the Schema Manager).

Step 3.4. Configure Other Child Tabs

To configure additional subject area child tabs, update the Informatica Data Director configuration file.

You can configure the **Primary Object Part Of** and **XREF** subject area child tabs.

Step 4. Configure Cleanse and Validation

Validation and cleanse are optional elements for a primaryObject, one2ManyChild, and many2ManyChild.

The IDD Configuration Manager does not create the cleanseFunction element - it merely binds the cleanse function to columns in the base object.

The data that the IDD application user has entered into the subject area attributes is fed into the cleanse function as inputs. The base object record is then updated by the outputs from the cleanse function.

The cleanse function can report validation errors if it is configured with a validationStatus output. If validation errors are found, then the IDD application displays any errors next to any field or fields with problems.

1. Create the validation function library using the ValidationCleanseLib sample in the Informatica MDM Hub Resource Kit as a template.
2. Using the Cleanse Functions tool in the Hub Console, deploy the created cleanse library into the ORS.
3. Using the Cleanse Functions and Mappings tools in the Hub Console, create cleanse functions and mappings to be used in IDD applications.
4. Using the Configuration Manager, configure these functions for use in an IDD application (in the subject area Edit dialog).
5. Deploy and test the cleanse and validation functions. Verify that all fields are properly cleansed and validated.

Step 5. Configure Search

Search configuration involves Basic and Extended Search, as well as public queries.

Advanced Search comes pre-configured with no editable configuration settings.

Step 5.1. Configure Basic Search

Basic Search allows IDD application users to search for subject area instances through the creation of queries in the subject area.

The results are displayed using an MDM Hub package that is created in the Packages tool in the Hub Console. IDD uses the new mode of the **searchQuery** API to display the results.

The search package must satisfy the following criteria:

- It is based on the root base object of the subject area.
- It returns a single result row for each subject area entity.
- It contains the ROWID_OBJECT of the root base object of the subject area.

The package used for search must contain the columns that are needed to present the search results to the user. An IDD application searches directly against a root base object and the associated children. It does not query against the attributes in the display package.

IDD does not remove duplicates from the search results. A package must be constructed to return a single row for every found entity.

1. To ensure a search package returns a single row for each entity, test the search package directly through SQL. One method of testing is to run spot-checks on the entities with a known number of children of different types.
2. Identify the primary searchable attributes. In the Schema Manager, create the appropriate custom indexes to support these searches.
3. To test the searches, create the different types of queries and run them in an IDD application. Use different combinations of search criteria to ensure the satisfactory performance of these searches.
4. Additionally, search can be configured for objects that are not part of the subject area when you use the Search on Child tab in the search configuration. This allows you to search on any object for which there is a match path from the primary object. These objects will be available in the Query Builder.

   Search on Child allows you to search the following types of data:

   - Related data that is not part of the subject area.
   - Cross references of data within the subject area.
   - In general, any data that can be related to the primary object by a match path.

Step 5.2. Configure Extended Search

Extended Search uses the searchMatch API to request fuzzy searches through the data.

1. You need to ensure that all necessary match columns have been created. No extra configuration is required in an IDD application to enable the fuzzy search. IDD will automatically map the search criteria supplied by the IDD application user to the available match-enabled columns, and then execute the search.
2. Before testing the Extended Search configuration, verify that the data has been properly tokenized, then test the fuzzy search facilities by creating the search queries to include the subject area attributes that have underlying match-enabled columns.

   For more information, refer to "Configuring the Match Process" in the *Multidomain MDM Configuration Guide* or Hub Console online help, as well as the description of the searchMatch API in the *Multidomain MDM Services Integration Framework Guide* or Javadoc.
3. Extended Search uses the **searchMatch** API with matchType=NONE. In the default configuration, all possible match columns get generated on each searchMatch request. IDD can be configured to generate

only specific match columns. On the Search tab in the subject area dialog box, you can specify the specific set of match columns that is to be generated.

Note: By default, in this mode of searchMatch, the search level defaults to Narrow. This is the most restrictive level, but it can be overridden by configuring the following setting in cmxcleanse.properties:

```
cmx.server.match.searcher_search_level=<level>
```

where <*level*> is one of the following settings: Narrow, Typical, Exhaustive, or Extreme. For more information about search levels in match rule set properties, see "Configuring the Match Process" in the *Multidomain MDM Configuration Guide*.

Step 5.3. Configure Public Queries

IDD allows administrators and expert users to share the queries that they create with all other users.

- We recommend that you configure at least one most commonly-used search as public for each of the subject areas defined in an IDD application.

  This will allow users to quickly navigate through all subject areas without needing to create their own versions of the common queries.

Case-Insensitive Searching

Extended Search, because it is based on the Informatica MDM Hub match capability, is case insensitive.

In general, case-insensitive searching is not available for Basic Search. The exception to this is when all of the data in the subject area is already either upper-case or lower-case. In this scenario, the searchQuery API can be configured to convert the incoming search terms to upper or lower case before executing the query. For more information, see the description of SearchQuery in the *Multidomain MDM Services Integration Framework Guide* or Javadoc.

Step 6. Configure the Match Process

Configure how the match process identifies duplicate records.

You configure the match process on the **Match Settings** tab in the **Subject Area** dialog box. You specify a predefined match rule set and the match type. You can also select match columns.
For more information about configuring match settings, see the *Multidomain MDM Data Director Configuration Manager Online Help*. For more information about match rules and match rule sets, see the *Multidomain MDM Configuration Guide*.

Step 7. Configure MDM Workflows

You can configure your Data Director (IDD) application to use the predefined MDM workflows that are deployed when you install the embedded ActiveVOS Server.

Your next step depends on whether your MDM environment includes the ActiveVOS Server:

- If your environment includes the ActiveVOS Server, select the MDM workflow that you want to use as your approval workflow.
- If your environment does not include the ActiveVOS Server, you need to install it using the Hub Server installer. For more information, see the *Multidomain MDM Installation Guide*.

Related Topics:

- "Workflows and Tasks" on page 132
- "Manual IDD Configuration" on page 56

Setting a Default Approval Workflow for the Subject Area Data View

When data stewards change master data, they can send the update for approval by clicking the **Send for Approval** button. This action opens the Create Task dialog box. The default approval workflow is displayed in the Task Type field.

Before you modify the task type to set a default approval workflow, ensure that there are no tasks in the IDD Tasks dashboard.

1. In the IDD Configuration Manager, select the application and click **Edit**.
2. Click the **Tasks** tab.
3. Under Task Types, click the task type with the name of the approval workflow that you want to use as the default, and click **Edit**.
4. Select the **Create task type by default on approval** check box, and click **OK**.

   Note: If the check box is disabled, another task type has this option set. Edit the other task types to find the task type with this option set and clear the check box. Then you can set the option in the preferred workflow task type.

Updating Workflows to Support Multiple Task Actions

You can configure an ActiveVOS workflow to allow reviewers to perform multiple actions on a task without closing the Task tab. For each task action where you want to support multiple actions without closing the Task tab, set the `closeTaskView` property to `false`.

1. Open ActiveVOS designer.
2. Open the workflow `.bpel` file.
3. For each task action that you want to change, edit the action definition and set the following parameter:

   ```
   <mdmavxsd:closeTaskView>false</mdmavxsd:closeTaskView>
   ```
4. Deploy the `.bpel` file to ActiveVOS.

Step 8. Configure Security

All application security in Data Director is controlled by MDM Hub Security Access Manager (SAM) policies configured in the Hub Console.

Data Director application behaviors can be very sensitive to the security configuration.

1. To configure and test the application for Data Director, use the admin user or a user with full privileges to all secure resources.

   For more information, see the *Multidomain MDM Security Guide*.
2. For each subject area, you can configure row-level security filters. By default, no security filters are defined.

   On the Search tab in the subject area dialog box, you can configure data security rules.
3. For any given user, the assigned user roles might include several data filters.

   For example, a user might have rights to records with an address in CA via one role, and rights to records with an address in NY via another role.

Related Topics:

- "Data Security" on page 111
- "IDD Security Configuration" on page 105

Step 9. Configure User Interface Extensions

Configure user interface extensions.

1. An IDD application can be customized by embedding external content in the web page and invoking actions from places in the IDD application.

   Content can be embedded using:

Element	Description
Top-level Tab	Tabs can be added alongside the tabs for the Start workspace, the Data workspace, and Tasks workspace.
Start workspace	A component or widget can be added to the Start workspace.
Child tab in Data View	Tabs can be added as children of a subject area.

2. Custom actions can be configured so that they are invoked from menu items in various places in an IDD application.

   Contextual information can be passed when invoking the external action.
   The following table shows areas of an IDD application where these actions can be configured, along with the available contextual data.

Area	Available Contextual Data
Subject Area	rowid_object and data from the primary object
One:Many Child	rowid_object and data from the child
Many:Many Child	rowid_object and data from the child
Search Results	rowid_object of the selected data in the search result list

Related Topics:

- "User Interface Extensions" on page 70

Step 10. Localize the Application

Four sets of resource bundles contain the strings that are displayed in an IDD application.

Each set includes the following components:

- The default file.
- A placeholder English language file. This file can be empty.
- Localized versions of the file, if needed.

For example, for the MessageBundle set, there is the default file MessageBundle.properties, and the placeholder English language file MessageBundle_en.properties.

Each resource bundle file is a UTF-8 encoded properties file. Each entry in the file is a name/value pair, `<name>=<value>`. Examples:

```
title=Business Data Director
locale=Locale
search=Search
```

For each entry:

- <name> is a fixed value that is referenced by the IDD application and cannot be changed
- <value> is the part that can be localized

To localize the application:

▶ Use the IDD Configuration Manager to add resource bundle files to an IDD application, either by including them in the application ZIP file that is imported, or by importing them individually into an existing IDD application.

Related Topics:

- "Application Components" on page 104

CHAPTER 4

IDD Configuration Manager

This chapter includes the following topics:

IDD Configuration Manager Overview

The IDD Configuration Manager is used to add, modify and manage IDD applications.

An IDD application consists of an XML configuration file, resource bundles, help files, and other components. A complete IDD application can be imported or exported as a ZIP file containing all of these components.

The IDD Configuration Manager is designed to be used for creating and maintaining the configuration of an IDD application. It does not yet expose all of the available configuration options - some functionality must be configured manually by exporting and editing the XML configuration file directly, then re-importing the file back into the IDD Configuration Manager.

Related Topics:

- "Application Components" on page 104
- "Manual IDD Configuration" on page 56

Starting the Informatica Data Director Configuration Manager

To start the Informatica Data Director Configuration Manager, use a supported web browser.

1. Open a supported web browser.

   For information about supported web browsers, see the Product Availability Matrix on the Informatica My Support Portal at https://mysupport.informatica.com/community/my-support/product-availability-matrices.

2. In the address bar, enter the following URL to access the IDD Configuration Manager login page:

   ```
   http://<MDM Hub host>:<port number>/bdd/config/
   ```

3. Enter the login name and password, and then click **Log In**.

   You must log in as a user that has all privileges for all base objects. For more information about configuring user privileges, see the *Multidomain MDM Security Guide*.

The Informatica Data Director Configuration Manager launches and the Applications page appears.

Home Page

The IDD home page consists of the following elements:

Element	Description
Applications list	List of existing IDD applications.
Command Bar	Available commands (described below)
Application Summary	Summary of existing IDD applications, including the following properties: - Logical name and display name - Validation status - Deployment status - URL to launch the IDD application
Component Types	Available only if the Informatica Data Components (IDC) feature is licensed for your Informatica MDM Hub implementation. For more information, refer to the *Multidomain MDM Data Director Configuration Manager Online Help* and the *Multidomain MDM Data Components Implementation Guide*.
Login Provider Settings	Shortcut to the screen for configuring Custom Login Provider module (SSO Support).

The IDD command bar contains the following commands:

Command	Description
Add	Add a new IDD application.
Edit	Edit the configuration of the selected IDD application.

Command	Description
Delete	Delete the selected IDD application.
Export	Export an IDD application configuration (ZIP file).
Validate	Validate the selected IDD application.
Application State	Change the deployment state of the IDD application: full, limited, or not deployed.
Import	Import an IDD application configuration (see below for formats).
Re-deploy	Removes and redeploys an IDD application.
Clear Cache	Clears the local IDD cache for the selected IDD application. This cache stores Hub metadata and should be cleared if there have been changes in this metadata.

Online help is also available from any page in the Configuration Manager.

ORS Binding

An IDD application configuration declares one or more logical ORS databases.

A *logical ORS database* is an IDD configuration pointer to a physical ORS database in the Hub Store that is configured in the Hub Console. All Informatica MDM Hub objects that are referenced in a configuration are always in the context of a specific logical ORS. For an IDD configuration to be valid, the objects it references must exist in the associated physical ORS.

When an IDD application is added or imported, the logical ORS databases it declares must be bound to a physical ORS that is registered with the Informatica MDM Hub.

The ORS binding is used to connect an IDD application to an ORS and to validate the configuration. Also, the ORS binding is used by the IDD Configuration Manager to fetch metadata about the ORS.

Add an IDD Application

The Add command is used to create a new IDD application.

A new IDD application is defined by its name, display name, description, and list of logical ORS databases. After adding the application, choose the Edit command to make more detailed changes to the application configuration (such as adding subject areas).

Import an IDD Application Configuration

The Import command is used to create or update an IDD application.

It provides the following three import options - two for importing a full application, and one for importing a component into an existing application:

Import Option	Description
Import IDD configuration only (XML)	Create a new IDD application by importing the IDD configuration XML. This can be used to replace an existing IDD application with the same name. If so, the existing application is entirely replaced (as if you performed a delete followed by an import). If an application with the name of the new application already exists, you can use the option to import the application with a different name. **Note:** If you are replacing an IDD application, you have to reconfigure the **Resource Privileges** assigned for all the roles in the hub console.
Import complete IDD application (Zip)	Create a new IDD application by importing a .zip file containing the various component files such as XML, resource bundles, and help files. The maximum .zip file size that you can import is 20 megabytes. In IBM Db2 environments, to import a file larger than 1 megabyte, run the following command to set the maximum file size allowed: `ALTER TABLE CMX_SYSTEM.C_REPOS_DS_CONFIG ALTER COLUMN BLOB_DATA SET DATA TYPE BLOB(max file size, bytes);` **Note:** If you are replacing an IDD application, you have to reconfigure the **Resource Privileges** assigned for all the roles in the hub console.
Import to existing IDD application	Update an existing IDD application by importing an individual file. This is used to add or replace any of the component files of the IDD application. **Note:** You can also use this option when promoting changes from one environment to another environment.

Related Topics:

- "Application Components" on page 104

Validation, Application State, and Deployment

The following persisted parameters determine how and whether an IDD application is deployed.

Parameter	Description
valid_ind	Contains the most recent validation status for the application. The validation status is a single value that represents the highest (most severe) error that has been found.
active_ind	Managed directly by the user to reflect the intention for deployment of the application.

Validation

An IDD application configuration is loosely coupled with the metadata in an ORS.

The configuration contains references to objects in an ORS. Changes to an ORS (the addition, modification, or removal of base objects, columns, cleanse functions, and so on) are not automatically reflected in the IDD configuration. For this reason, the IDD validation process is necessary and must be repeated periodically.

Validation is run in the following circumstances:

- when requested by the user in the IDD Configuration Manager
- when importing an IDD configuration
- before deploying an application when the application server is started up

The following validation levels are available.

valid_ind	Validation Level	Description
-1	Not Validated	The IDD application has not been validated.
0	No Error	No errors or warnings were found during validation.
1	Information	Provides information to the user. No configuration changes are required.
2	Warning	A configuration might need to be changed, but should cause no run-time problems.
3	Error	A configuration error must be fixed. Run-time problems can be expected.
4	Critical Error	Same as Error, but indicates a problem that requires even more urgent attention.
5	Fatal Error	An error that prevents the IDD application from running at all. The application will not be deployed under any circumstance.

Application State

The application state is controlled by the user in the IDD Configuration Manager.

It stores the intended deployment for the IDD application.

Note: An IDD application can be deployed even if the configuration contains errors. Only fatal errors (described in the previous section) will prevent an IDD application from being deployed. It can be useful to

deploy an IDD application that contains errors when building out an application, allowing the implementer to test parts of the configuration while other parts are incomplete.

active_ind	Name	Description
-1	Not Deployed	IDD application is not deployed. Useful when the application is in development. Changes can be made and saved without the additional overhead of deploying the application.
0	Limited Deployment	IDD application is deployed, but only users that are administrators can log in. The application will not be displayed in the list of available applications. You must access the application using its full URL: `http://<hostname>[:<port>]/bdd?bdd_name=name`
1	Full Deployment	IDD application is deployed for full use. It is displayed in the list of applications and any authorized users can run the application.

Deployment

Deployment is the process of taking an IDD configuration and making it available as an application.

An application is not deployed if active_ind is -1 for that application.

Deployment occurs in response to the following events:

Event	Description
Application Server Startup	All IDD applications with active_ind that is not -1 are first validated. If the validation level is not Fatal Error, then the IDD application is deployed. At this time, only a partial validation is run to check for fatal errors.
Import / Save	Any time an IDD application is imported or saved, it is also deployed unless its active_ind is -1.
Re-deployment	User re-deploys an IDD application.

Edit Application

From the Edit Application screen, you can view and edit configuration details for a selected IDD application. IDD uses metadata from the logical ORS to present the configuration options available.

The following tabs are available at the bottom of the screen:

Tab	Description
Subject Areas	Define subject area groups, subject areas, subject area children, and subject area grandchildren for the selected IDD application.
Tasks	Define tasks for the selected IDD application. For more information, see the Configuration Manager online help.
Controls	Available only if the Informatica Data Components (IDC) feature is licensed for your MDM Hub implementation. For more information, see the *Multidomain MDM Data Director Configuration Manager Online Help* and the *Multidomain MDM Data Components Implementation Guide*.

The following command buttons are also available:

Button	Description
Save	Saves the latest changes to the database. If the application state is anything but Not Deployed (-1), then the IDD application is redeployed after you save the changes.
Validate	Runs validation on the current IDD application configuration and displays the validation report.
Bind	Used to change the logical ORS binding.
Generate Business Entity Schema	Generates configuration files for all business entities in the IDD application.

Related Topics:

- "Subject Areas" on page 43

Logical ORS Databases

When editing a configuration, the first task to complete is the configuration for the logical ORS databases.

For each of these ORS databases, you must select a source system.

If Hierarchy Manager is to be used by the IDD application, then the HM configuration must also be selected. The icon to the right of the HM Configuration drop down is used for additional HM parameter settings (such as hops and relationship settings).

Note: In the IDD Configuration Manager, in the **HM Settings** window, the value for **Total Relationships** should not exceed more than 2000.

Session Timeout

From the Edit Application screen, you can set a session timeout for a selected IDD application.

To set the session timeout, enter a value in minutes in the **Session Timeout** field. Then save the IDD application. By default, a session times out after 30 minutes.

If you change the session timeout value, all active sessions in IDD become invalid and users must log in again.

Subject Areas

The Subject Areas tab in the lower portion of the screen provides a tree that shows how the IDD application is configured.

As items are selected in the tree, the Add, Edit, and Delete buttons update to reflect the options available. The levels in the tree are:

Tree Level	Description
IDD Application	Subject area groups can be added.
Subject Area Group	The subject area group can be edited or deleted. Subject areas can be added. The subject area group identifies the logical ORS to which the child subject areas belong, and which base object is the primary table for these subject areas. A subject area group can have one or more child subject areas - all sharing the same primary table. These subject areas are grouped together in the IDD application.
Subject Area	The subject area can be edited or deleted. Subject area children can be added. If the subject area group contains more than one subject area, each subject area defines the HM entity type or sub-type qualifier that identifies the subject area. You also specify: - the package used to display search results - the match rule set and match type to use for duplicate checks - the columns from the Primary table that are part of this subject area
Subject Area Child	The subject area child can be edited or deleted. For each subject area child, you must specify: - the type of relationships (one:many, many:many, and so on) - which match path leads to the child table (the match path list is populated based on the relationship type selection) - the columns from the child table that are to be displayed.
Subject Area Grandchild	The subject area grandchild can be edited or deleted. For each subject area grandchild, you must specify: - the type of relationship (one:many, many:many, and so on) - which match path leads to the child table (the match path list is populated based on the relationship type selection) - the columns from the child table that are to be displayed.

Subject Area Group Properties

The dialog used for adding and editing a subject area group is used to configure:

- Name and display name. Name is the internal identifier for this subject area and must consist of alphanumeric characters only - special characters are not allowed.
- Logical ORS the subject area group is bound to

- Primary table for the subject areas in the group:

Feature	Description
Name and Display Name	These are used to identify the subject area group. Name is the internal identifier for this subject area group and must consist of alphanumeric characters only - special characters are not allowed.
Logical ORS	Configures which Logical ORS from which the objects in this subject area group come.
Primary Table	Configures which base object is the primary or root table for the subject areas in the subject area group.
Search Only	This is selected for a subject area group that has data that is created and maintained outside of an IDD application. The subject areas defined in this group are visible within an IDD application only when creating a foreign key from another subject area (search is used to find the record to relate).

Subject Area Properties

The dialog used for adding and editing a subject area is used to configure the following properties:

- Name and display name: Name is the internal identifier for this subject area and must consist of alphanumeric characters only - special characters are not allowed. A A subject area name cannot begin with a number.
- HM entity type: This property defines the kinds of objects that can be related, if any.
- Search results display package: This property is used to display search results for this subject area. The package must have the primary table of the subject area group as its primary table.
- Potential Matches link columns: This property defines which column from a layout must be displayed as a hyperlink that opens a Potential Match entity in a new Data View tab.
- Subtype column: This property specifies the column used for the subtype filter: Type code (category) for this subject area. Automatically set if an HM Entity Type is selected.
- Subtype value: This property specifies the value used for the subtype filter. Automatically set if an HM Entity Type is selected.
- Number of Frozen Columns: This property shows the number of columns frozen in search results for subject area.
- Show XREF: If selected, the IDD application displays a child tab for the subject area that displays the cross references for the primary object.
- Tabs to configure the following settings:

Feature	Description
Layout	Configures which columns from the base object are available in the IDD application for display and edit, what type of UI component to use and, if it is a lookup, whether the lookup data is localized.
Match Settings	Configures the match rule set and match type to use for duplicate checks.
Search	Configures search properties.

Feature	Description
Data Security	Configures role-based, row-level security for the subject area.
Data Masking	Configures role-based data masking for columns that are selected in Layout tab.
Cleanse	Configures the cleanse function to be used for cleanse and validation.
Label	Configures how a label is generated for the subject area. This label is used, for instance, as the title for a Data view tab.
Task Assignment	Configures how tasks are assigned. It specifies the roles list and user for each task type.
Order of Children	Configures the order of child tabs for the subject area.

Subject Area Child and Grandchild Properties

The dialog used for adding and editing a subject area is used to configure the following properties:

- Name and display name. Name is the internal identifier for a subject area child or grandchild, and must consist of alphanumeric characters only - special characters are not allowed.
- Child type - the type of relationship to the parent
- Match Path to Child - the match path component that leads to this child object
- Tabs to configure the following settings:

Feature	Description
Layout	Configures which columns from the base object are available in the IDD application for display and edit, what type of UI component to use and, if it is a lookup, whether the lookup data is localized. **Note:** This setting is not applied to the filters for child records. All columns are available for filters.
Data Masking	Configures role-based data masking for columns that are selected in Layout tab.
Cleanse	Configures cleanse functions to be used for cleanse and validation.

Related Topics:

- "Lookup Localization" on page 45
- "Step 4. Configure Cleanse and Validation" on page 30

Lookup Localization

An Informatica Data Director application populates a list of acceptable values for columns that you configure in the Schema Manager as lookups. To create localized lookups, you require a localization table. When you create a lookup, use a unique display name. The Informatica Data Director cannot distinguish lookups with different codes that share the same display name.

Informatica Data Director also supports localization of the lookup display values. You can configure lookup display values in the Layout tab of the Informatica Data Director Configuration Manager for subject areas and subject area children.

For example, an Operational Reference Store has the following tables:

- C_PARTY
- C_LU_SALUTATION
- C_LCL_SALUTATION

The C_PARTY table has salutation lookup code configured in the C_LU_SALUTATION table. For each salutation code, the display name might have a localized value configured in the C_LCL_SALUTATION table.

To generate the list of values for the locale of a particular user, the Informatica Data Director first searches for a lookup name in C_LCL_SALUTATION based on the locale. If the Informatica Data Director does not find a lookup name in C_LCL_SALUTATION, then it uses the lookup name from the SALUTATION_DISP lookup table.

Note: The language code and country code determine the locale. The values for language code and country code are two-letter ISO codes.

The configuration for the previous scenario specifies that the column has localized lookup values and which table and columns are used. The following sample XML shows the configuration for the previous example:

```
<column columnUid="C_PARTY|SALUTATION_CODE"
        editStyle="FIELD"
        horizontalStyle="SMALL">
  <columnI18NLookup languageCdUid="C_LCL_SALUTATION|LANGUAGE_CODE"
                    countryCdUid="C_LCL_SALUTATION|COUNTRY_CODE"
                    lookupFKUid="C_LCL_SALUTATION|SALUTATION_CODE"
                    localizedNameUid="C_LCL_SALUTATION|LOCALIZED_STRING"/>
</column>
```

Related Topics:

- "Lookup Tables" on page 22
- "Locale Codes" on page 149
- "Manual IDD Configuration" on page 56

Import a Data Import Template

An Informatica Data Director (IDD) application developer can configure an IDD application to permit authorized users to import data from a source file. The data steward creates a data import template, which you import into the IDD application configuration.

Note: Data import is available for IDD applications that implement the subject area data model and the legacy IDD views.

For more information about importing data, see the *Multidomain MDM Data Director User Guide*.

Importing the Data Import Template

From the IDD Configuration Manager, an MDM administrator imports the data import template into the Data Director (IDD) application. The import process validates the template.

1. Log in to the IDD Configuration Manager.
2. Select the application.
3. Click **Import** > **Import to existing IDD application**.

   The **Import to existing IDD application** window opens.
4. From the **Configuration type** list, select **Data Import Template**.

5. Click **Browse** and select the XML file that contains the data import template.
6. Click **Import**.

   The import process validates the template. The **Validation Result** window opens and displays any errors.
7. If there are validation errors, resolve the errors in the template and then re-import the template.
8. In the **Validation Result** window, click **OK**.

Custom Login Provider Package

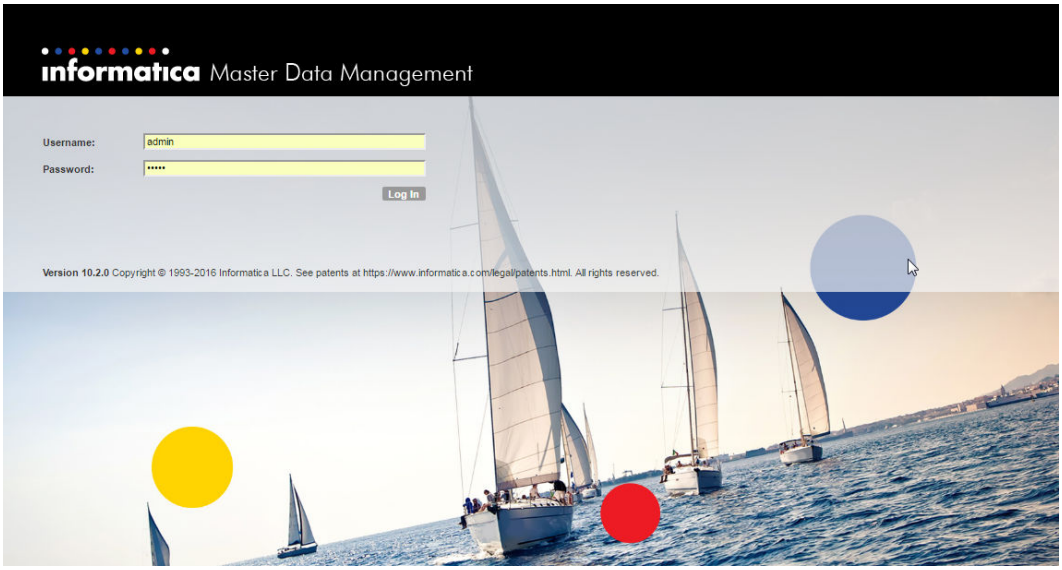
The custom login provider package is an archive file that contains Java classes. You can use the Informatica Data Director Configuration Manager to upload the archive file. The custom login provider package must be in a ZIP file.

In the Entity 360 framework, the Custom Login Provider package must be in a ZIP file that contains the following:

- META-INF folder. This folder contains a MANIFEST.MF file which has a Login-Provider-Class-Name entry that contains a name of a class that implements the LoginProvider interface.
- JAR file with Custom Login Provider implementation.
- Other JAR files that contain dependencies for the Custom Login Provider implementation, including utility classes, logging, and third-party libraries.

You can configure the custom login provider package to use the Informatica Data Director login form or the login form of an external identity provider, such as Google or Salesforce.

The following image shows the Informatica Data Director login form:



If you do not upload a custom login provider package, the default Informatica Data Director implementation authenticates users against the credentials stored in the MDM Hub master database.

Custom Login Provider Packages in the Resource Kit

The Resource Kit contains sample login provider packages that you can use with the Informatica Data Director application. These single sign-on login packages are stored as both JAR and ZIP files. Database

administrators and other technical members of an MDM implementation team can use these files to create their own custom login provider packages.

The sample single sign-on package files are in the following directory:

<*infamdm installation directory*>/hub/resourcekit/samples/sso

Uploading the Custom Login Provider Package

To upload the custom login provider package, use the Informatica Data Director Configuration Manager.

1. From the navigation pane in the Informatica Data Director Configuration Manager, click **Login Provider Settings**.
2. From the **Login Provider Settings** pane, click **Edit**.
3. From the **Edit Login Provider Settings** dialog box, click **Browse**.
4. Select the custom login provider archive file, and then click **OK**.
5. Enter the name of the ZIP file with the implementation of Login Provider class in the Login Provider Implementation Archive field.

   You must wait for the ZIP file to complete uploading to the server.
6. Enter the name of the class that implements `com.siperian.bdd.security.LoginProvider` in the Login Provider Class name field.

   This is the fully qualified name of the class that implements LoginProvider.
7. Click **OK**.

   IDD validates the uploaded ZIP file and creates an instance of the specified Login Provider class.

Third-Party Libraries

In IDD, you could use a custom login provider with third-party libraries. In the Entity 360 framework, however, all third-party libraries must be packaged in the same ZIP file as the custom login provider package.

Implement Custom Login Provider

Custom Login Provider is a Java class that implements the LoginProvider interface (`com.siperian.bdd.security.LoginProvider`) defined in IDD. It provides support for Single Sign-On (SSO) authentication mechanism.

Login Provider works together with Hub Login Module. All the data that is required by Hub Login Module for verification of authenticated user must be passed from Login Provider as a `securityPayload` byte array field of `com.siperian.bdd.security.LoginCredentials` class. This field is passed from Login Provider to Hub Login Module as is and contains implementation-specific encoded information about users.

Custom Login Provider with External Login Form

If a particular authentication mechanism requires a non-IDD login page, then the implementation of Custom Login Provider must use the interface methods listed and described in the following table:

Interface Method Name	Description
initialize	IDD calls this method before any other method of the Login Provider implementation and passes a set of properties that describe the context of execution. In IDD, these properties contain an entry, which can be referenced as LoginProvider. The property SSO_POST_REDIRECT_PAGE_PROPERTY contains the URL of the jsf page that can POST data to external login provider. A Login Provider implementation might use this page to redirect IDD to External Login Page using POST method.
isUseIDDLoginForm	This method must return FALSE.
redirectToProviderLoginPage	This method must form URL to external login form and call redirection to that page. You can also redirect to external login page using the POST method.
extractLoginCredentials	IDD invokes this method when a new user authentication request arrives. If the request contains information from external identity provider, such as request parameters, and cookies, then this method must extract them and return the LoginCredentials (com.siperian.bdd.security.LoginCredentials) instance with properly filled fields. If the request does not contain authentication information, then the method must return NULL.
encodeComponentUrl	This method is not implemented as user name and password is requested by external login form that IDD does not recognize.
onLogout	This method is called when a user logs out. It can run a logout on external identity provider and cleanup parameters defined by the requestLoginCredentials method.
getLogoImageBody	This method must return NULL.

After a successful login, you are directed to the IDD main page or the Informatica Data Controls (IDC) component page, depending on your initial request.

Also, you can bypass external authentication by using the `internal_login_form=true` parameter in the IDD URL that displays the IDD login.

For example,

```
http://localhost:8080/bdd?internal_login_form=true
```

In this case, the user name and password is checked against the list of MDM Hub users.

Pass Credentials to External Link

If you need to embed external links into IDD and the links use the same SSO provider (for example, SalesForce.com) as the installed Custom Login Provider, then use this method for adding authentication information to the link URL. If no information is added, then the method should return the same URL string that was passed to it as a parameter or null.

Example:

Assume that you implement LoginProvider for work with SalesForce.com.

You also define the external link with URL `https://na7.salesforce.com/home/home.jsp` to see the home page of the SalesForce.com account embedded in the IDD screen.

The encodeComponentUrl method receives this URL and converts it to the following:

```
https://na7.salesforce.com/secur/frontdoor.jsp?sid=<SFDC_API_SESSIONID>&retUrl=https://
na7.salesforce.com/home/home.jsp
```

After this transformation, an Iframe in the IDD page displays the requested home page without redirection to the SalesForce login form.

Using a POST Page

IDD uses the POST page to redirect users to an external login page. This page is submitted after it is loaded to the client.

The source of the page uses the JSF requestScope predefined variable to access the parameters described in the following table:

Parameter Name	Usage
providerGateURL	Must be a string value. It defines the URL where the form will be submitted (form action).
authParameters	It is a map of key–value pairs. Each value pair is used for creating hidden input. Map entry key is used as the input name and value as input field value.

In the following example, the postRedirectPageUrl variable is set up during a call to an initialize method:

```
public void redirectToProviderLoginPage(HttpServletRequest httpRequest,
                                        HttpServletResponse httpResponse,
                                        String returnUrl) throws LoginProviderException {
RequestDispatcher dispatcher =
                  httpRequest.getRequestDispatcher(postRedirectPageUrl);
httpRequest.setAttribute( PROVIDER_GATE_URL_ATTR, authReq.getOPEndpoint() );
httpRequest.setAttribute( AUTH_PARAMETERS_ATTR, authReq.getParameterMap() );
dispatcher.forward( httpRequest, httpResponse );
}
```

To send a redirect to the new page on logout, you can add the following code to the redirectToProviderLoginPage() method:

```
if("gotoLogoutPage".equalsIgnoreCase(httpRequest.getParameter("logoutParam"))){
    try
        { httpResponse.sendRedirect("http://www.google.com/"); }
        catch (Exception e)
        { // TODO Auto-generated catch block e.printStackTrace(); }
}
```

The onLogout() method writes code in the response as shown in the following example:

```
{\"logoutURL\":\"/mdm/entity360view/?logoutParam=gotoLogoutPage\",\"kerberos\":\"true\"}
```

Configuring E360 to Send POST Requests to Web Service

Sometimes a custom login provider uses web services that expect a POST request. Entity 360 includes a servlet that sends POST requests. To configure the servlet to send a POST request to a third-party web service, enter the URL of where to send the POST request in the `redirectToProviderLoginPage` method.

1. Use a text editor to modify the Custom Login Provider implementation.
2. In the properties passed to the `initialize` method of the Custom Login Provider, copy the URL of the servlet.

3. In the `redirectToProviderLoginPage` method, create a request

   a. In the `AuthParameters` attribute, set the parameters with name–value pairs.

      The name–value pairs comprise the body of the POST request.

   b. In the `ProviderGateURL` attribute, enter the URL where the POST request is sent.

      Note: Ensure that the URL ends with a "/" (forward slash). Otherwise, the E360 application generates a null pointer exception.

The following code shows a sample request in a Custom Login Provider implementation:

```
    @Override
    public void redirectToProviderLoginPage(javax.servlet.http.HttpServletRequest
request,
            javax.servlet.http.HttpServletResponse response, String originalRequest)
throws
    LoginProviderException {
        RequestDispatcher dispatcher = request.getRequestDispatcher(forwardUrl);

        Map<String, String> params = new HashMap<>();

        params.put("param1", "value1");
        params.put("param2", "value2");

        request.setAttribute("AuthParameters", params);
        request.setAttribute("ProviderGateURL", "http://external.server.com/");

        dispatcher.forward(request, response);
}
```

Custom Login Provider with IDD Login Form

If the authentication mechanism uses the IDD Login form for requesting user name and password, then the implementation of Custom Login Provider must use the interface methods listed and described in the following table:

Interface Method Name	Description
initialize	IDD calls this method before any other method of the Login Provider implementation and passes a set of properties that describe the context of execution. In IDD, properties contain the only entry. It can be referenced as LoginProvider. SSO_POST_REDIRECT_PAGE_PROPERTY and contains URL of the JSF page that can POST data to external login provider.
isUseIDDLoginForm	This method must return TRUE.
redirectToProviderLoginPage	This method is not used.
extractLoginCredentials	This method extracts user credentials from an Http Request. If the request contains authentication information, then this method must return LoginCredentials (`com.siperian.bdd.security.LoginCredentials`) instance with properly filled fields. If request does not contain authentication information, then the method must return NULL.
requestLoginCredentials	This method is called after a user submits the filled-in login form. This method is used for sending requests to an external identity provider for authenticating users. Properly filled instances of LoginCredentials are returned on successful authentication. If authentication fails, then `com.siperian.bdd.security.LoginProviderException` is thrown.
encodeComponentUrl	This method receives ExternalLink URL and can add authentication parameters.

Interface Method Name	Description
onLogout	This method is called when a user logs out. It can run a logout on external Identity Provider and cleanup parameters defined by the `requestLoginCredentials` method.
getLogoImageBody	This method returns InputStream with the image file body. You can use this method to display the logo of an external identity provider in the IDD login form. The image format must be PNG, JPEG, or GIF. The image must not exceed a width of 155 pixels and a height of 29 pixels. If this method returns NULL, then IDD uses the predefined image to indicate that the login form is handled by Custom Login Provider.

Build Login Provider Library

The LoginProvider class and all the IDD classes that are required for compilation of Custom Login Provider implementation are packaged into the `siperian-bdd.jar` file. This file is included in the MDM Resource Kit, which also contains sample implementation of LoginProvider. For more information, see *Multidomain MDM Resource Kit Guide*.

Set Up SalesForce SSO Authentication (WebLogic)

If you need to set up SalesForce SSO authentication for IDD, then host name verification must be disabled in WebLogic. You can disable host name verification using the following procedure:

1. Open WebLogic Server Administration Console and login.
2. Expand **Environment** and select **Servers**.
3. Click the name of the server that runs the Hub (the default is AdminServer).
4. In the Settings page, click the **SSL** tab.
5. Click **Advanced** at the bottom of the page.
6. Set the Hostname Verification field to **None.**
7. Click **Save**.
8. Restart WebLogic Server.

Set Up SalesForce SSO Authentication (WebSphere)

If you need to set up SalesForce SSO authentication for IDD, then WebSphere must be configured to trust the SalesForce server. You must retrieve the signer certificates from the SalesForce host you are trying to connect to and add them to the WebSphere trust store, using the following procedure:

1. Open WebSphere Administration Console and login.
2. Expand **Security** and then click **SSL certificate and key management** > **Manage endpoint security configurations**.
3. Expand **Outbound** and click **HTTP**.
4. Choose **SSL keystores** from the dropdown list.
5. Click **NodeDefaultTrustStore** > **Signer certificates**.
6. Click **Retrieve from port**.

7. Enter the following values in the **Host**, **Port** and **Alias** fields:
   - Host: `www.salesforce.com`
   - Port: `443`
   - Alias: `www.salesforce.com`
8. Click **Retrieve signer information**.

   The certificate's data is displayed.
9. Click **Apply**.
10. Repeat Steps 6 to 9 for the following hosts:
    - na10-api.salesforce.com
    - c.na10.visual.force.com
11. Click **Save**.
12. Restart the Websphere server.

Google Single Sign-On Login Provider Implementation Example

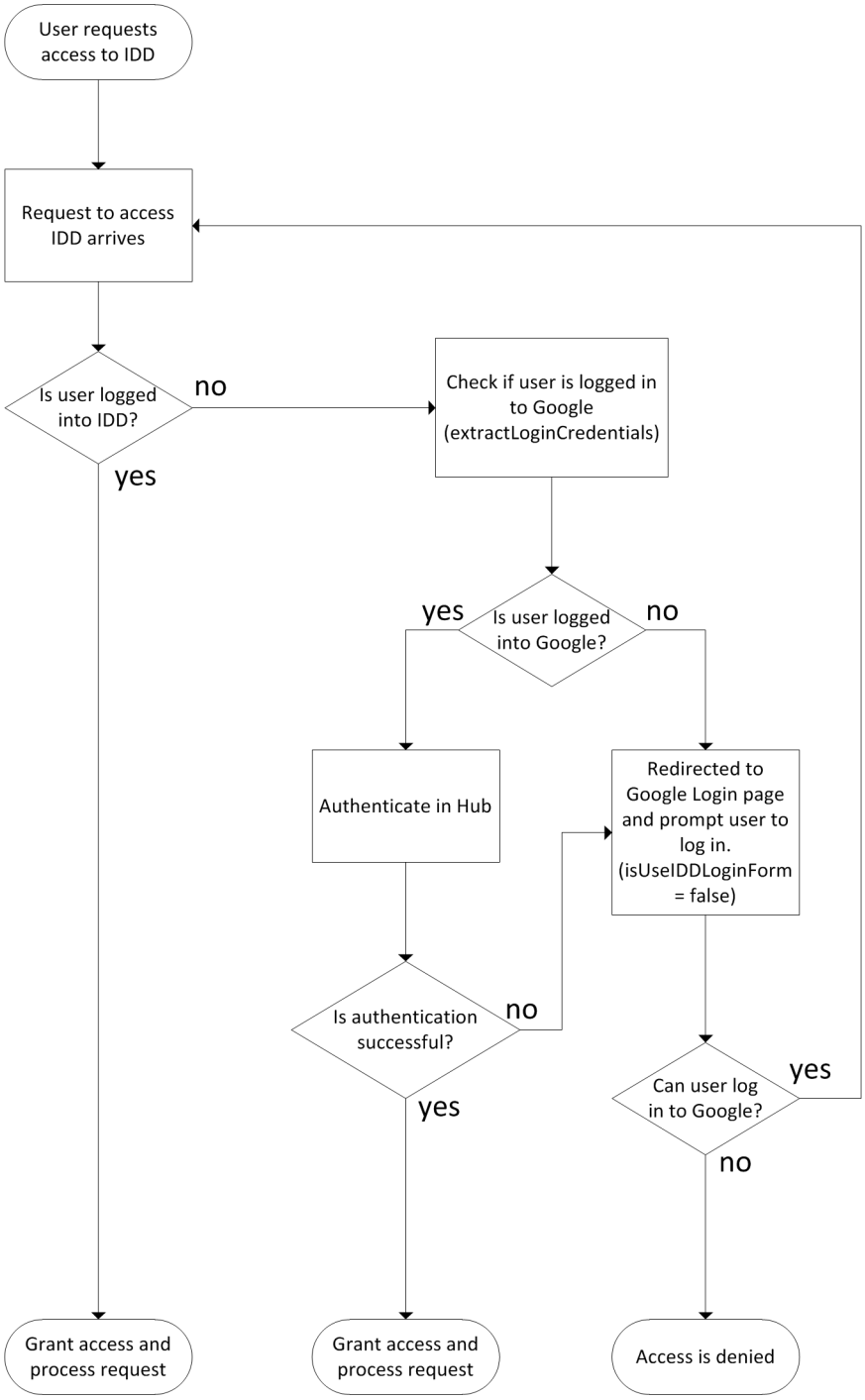
The Resource Kit contains a sample login provider implementation for Google Single Sign-On (SSO). The sample login provider implementation demonstrates one way of implementing SSO.

You can find the sample login provider implementation for Google SSO in the following file:

```
<MDM Hub installation directory>\hub\resourcekit\samples\sso\GoogleSSO\source\java\com
\siperian\dsapp\sso\google\GoogleLoginProvider.java
```

When a user requests access to Informatica Data Director, the login provider authenticates the user through a sequence of events.

The following image shows the sequence of events that occurs when you implement Google SSO with the sample login provider implementation:



The following sequences can occur based on whether the user is logged into Informatica Data Director, logged in to Google, or not logged in to Informatica Data Director or Google:

Sequence for users that are logged in to Informatica Data Director.

When a user requests access to Informatica Data Director, the login provider checks to see whether the user is logged in. If the user is logged in to Informatica Data Director, the login provider grants access to Informatica Data Director.

Sequence for users who are not logged in to Informatica Data Director, but are logged in to Google.

When the login provider determines that the user is not logged in to Informatica Data Director, it checks if the user is logged in to Google. If the user is logged in to Google, the login provider passes the Google credentials of the user to the MDM Hub. The MDM Hub Security Providers tool authenticates the Google credentials. If the MDM Hub Security Providers tool authenticates the user, the user can access Informatica Data Director. If the Security Providers tool does not authenticate the user, the login provider redirects the user to the Google login page to enter different credentials.

Sequence for users who are not logged in to Informatica Data Director, and are not logged in to Google.

When the login provider determines that the user is not logged in to Informatica Data Director or Google, the login provider redirects the user to the Google login form. In the sample implementation, the login provider redirects to the Google login form instead of the Informatica Data Director login form because `isUseIDDLoginForm` is `false`. If you set `isUseIDDLoginForm` to `true`, the login provider redirects to the Informatica Data Director login form.

After the user logs in to Google, the process begins again, but the user is now logged in to Google. The MDM Hub Security Providers tool authenticates the Google credentials for the user.

Set Up Google SSO Authentication

If you use Google SSO authentication for Informatica Data Director, configure Informatica Data Director to return to the login screen after a user logs out.

1. Open `cmxserver.properties` in the following directory:
   - On UNIX. `<infamdm installation directory>/hub/server/resources`
   - On Windows. `<infamdm installation directory>\hub\server\resources`
2. Add the following property to `cmxserver.properties`:

   ```
   cmx.bdd.redirect_to_login_after_logout=false
   ```
3. Restart the Hub Server application to reload the settings in the `cmxserver.properties` file.

CHAPTER 5

Manual IDD Configuration

This chapter includes the following topics:

Manual IDD Configuration Overview

The IDD configuration file (IDDConfig.xml) is an XML document that can be modified in the IDD Configuration Manager or exported and edited manually.

To edit the configuration for an existing application:

1. Export the IDD application to a ZIP file.
2. Extract the application ZIP file.
3. Edit the IDD configuration file (IDDConfig.xml).
4. Import the edited configuration file directly to replace the one in the database (Import IDD Configuration only). Alternatively, the IDD application can be re-zipped and the full application can be imported to replace all the files for the application (Import Complete IDD application).

XML Tools

The Informatica MDM Hub Resource Kit includes an XML schema (XSD file) for the IDD configuration file.

This is very useful when working with XML editors. It can guide you in editing the file and, most importantly, it is used by the editor to verify the correctness of the XML in an IDD configuration file. The IDD configuration file should pass this test before being imported into the IDD Configuration Manager.

While a simple text editor can be used to modify the IDD configuration, there are many XML editing tools that make working with XML much easier, including:

Editor	URL
XML Copy Editor	http://xml-copy-editor.sourceforge.net/
XML Spy	http://www.altova.com/products/xmlspy/xmlspy.html
oXygen	http://www.oxygenxml.com/

The IDD sample in the Resource Kit contains the following components that can help with manual configuration.

Resource Kit Item	Description
`siperian-bdd-config-6.xsd`	XML schema for the IDD configuration file. This file is found in the location `<Installation folder>\hub\resourcekit\sdk\bddXsdDoc\siperian-bdd-config-6.xsd`
HTML documentation for the XML schema	Javadoc-style documentation. Provides the same information found in the XML schema, but in a form that is easier to browse. **Note:** Refer to this documentation for the most detailed information on the XML elements and attributes in the IDD configuration file.
Sample IDD configuration	To be used with the sample schema.
Sample IDD user exits	An example of how to build custom java code to be integrated with IDD.
IDD library javadocs	Javadocs for the interfaces in Siperian-bdd.jar. Used for implementing IDD user exits in Java.

Work with the IDD Configuration XML File

An IDD configuration XML file can easily run to hundreds of lines.

A full file is not shown here, only the relevant snippet. You can find a full configuration file in the Resource Kit or by exporting it from the IDD Configuration Manager.

The following code snippet is an example of a subject area group with a single subject area:

```
<subjectAreaGroup name="Customer" primaryObjectUid="C_PARTY">
        <subjectArea name="Person">
            <primaryObject hmEntityTypeUid="Person">
                <subTypeQualifier columnUid="C_PARTY|PARTY_TYPE"  filterValue="Person"/>
                <cleanseFunction
```

```
                                cleanseFunctionUid="BDD Cleanse and Validation Library|
CVPerson">
                    <cleanseInput>
                        <cleanseColumn columnUid="C_PARTY|FIRST_NAME"
parameterName="firstName"/>
                        <cleanseColumn columnUid="C_PARTY|MIDDLE_NAME"
                                       parameterName="middleName"/>
                        <cleanseColumn columnUid="C_PARTY|LAST_NAME"
parameterName="lastName"/>
                    </cleanseInput>
                    <cleanseOutput>
                        <cleanseColumn columnUid="C_PARTY|FIRST_NAME"
parameterName="firstName"/>
                        <cleanseColumn columnUid="C_PARTY|MIDDLE_NAME"
                                       parameterName="middleName"/>
                        <cleanseColumn columnUid="C_PARTY|LAST_NAME"
parameterName="lastName"/>
                        <cleanseColumn columnUid="C_PARTY|DISPLAY_NAME"
                                       parameterName="displayName"/>
                    </cleanseOutput>
                </cleanseFunction>
                <layout columnsNum="3">
                    <column columnUid="C_PARTY|NAME_PREFIX_CD" editStyle="FIELD"
                            horizontalStyle="SMALL"/>
                    <column columnUid="C_PARTY|FIRST_NAME" editStyle="FIELD"
                            horizontalStyle="MEDIUM" required="true"/>
                    <column columnUid="C_PARTY|MIDDLE_NAME" editStyle="FIELD"
                            horizontalStyle="MEDIUM"/>
                    <column columnUid="C_PARTY|LAST_NAME" editStyle="FIELD"
                            horizontalStyle="MEDIUM" required="true"/>
                    <column columnUid="C_PARTY|GENERATION_SUFFIX_CD" editStyle="FIELD"
                            horizontalStyle="SMALL"/>
                    <column columnUid="C_PARTY|BIRTHDATE" editStyle="CALENDAR"
                            horizontalStyle="MEDIUM"/>
                    <column columnUid="C_PARTY|GENDER_CD" editStyle="FIELD"
                            horizontalStyle="SMALL">
                        <columnI18NLookup languageCdUid="C_LU_GENDER_LCL|LANGUAGE_CODE"
                                          countryCdUid="C_LU_GENDER_LCL|COUNTRY_CODE"
                                          lookupFKUid="C_LU_GENDER_LCL|GENDER_CODE"
                                          localizedNameUid="C_LU_GENDER_LCL|
LOCALIZED_STRING"/>
                    </column>
                    <column columnUid="C_PARTY|TAX_ID" editStyle="FIELD"
                            horizontalStyle="MEDIUM"/>
                    <column columnUid="C_PARTY|DISPLAY_NAME" editStyle="FIELD"
                            horizontalStyle="LARGE"/>
                </layout>
                <label existsFormat="{1},{2}">
                    <column columnUid="C_PARTY|LAST_NAME"/>
                    <column columnUid="C_PARTY|FIRST_NAME"/>
                    <column columnUid="C_PARTY_ELECT_ADDR|ELECTRONIC_ADDRESS"/>
                </label>
            </primaryObject>
            <search displayPackageUid="PKG_PERSON_SEARCH">
            </search>
            <match>
                <matchRuleSet uid="C_PARTY|IDL" type="BOTH"/>
            </match>
            <taskAssignmentConfig task="UpdateWithApproval">
                <securityRole roleUid="DataSteward"/>
            </taskAssignmentConfig>
            <taskAssignmentConfig task="UpdateWithOptionalApproval" >
                <securityRole roleUid="DataSteward"/>
            </taskAssignmentConfig>
            <taskAssignmentConfig task="UpdateRejectedRecord">
                <securityRole roleUid="DataSteward"/>
            </taskAssignmentConfig>
            <taskAssignmentConfig task="ReviewNoApprove">
                <securityRole roleUid="Manager"/>
            </taskAssignmentConfig>
            <taskAssignmentConfig task="FinalReview" >
```

```
            <securityRole roleUid="SrManager"/>
        </taskAssignmentConfig>
        <taskAssignmentConfig task="Merge">
            <securityRole roleUid="DataSteward"/>
        </taskAssignmentConfig>
        <taskAssignmentConfig task="Unmerge">
            <securityRole  roleUid="DataSteward"/>
        </taskAssignmentConfig>
        <dataSecurity>
            <securityFilter columnUid="MATCH_PATH_COMPONENT.C_MT_ADDRESS|STATE_CD">
                <securityValue value='CA'>
                    <securityRole roleUid="Customer-CA"/>
                </securityValue
            </securityFilter>
        </dataSecurity>
    </subjectArea>
</subjectAreaGroup>
```

Refer to the HTML documentation for the XML schema for details on the elements, attributes, and allowed values.

Subject Area

The items described in this section might require manual modification directly in the IDDConfig.xml file.

Lookup Column

An IDD application automatically populates a drop-down list of acceptable values for columns that are configured in the Schema Manager as lookups.

This is handled in the IDD Configuration Manager for columns that have a foreign key to the lookup table. If the foreign key does not exist (for example, for performance reasons), the information about the lookup table can be specified in the XML configuration.

An explicit lookup is defined using the columnLookup element, as shown in the following example.

```
<column columnUid="C_PARTY|GENDER_CD" editStyle="FIELD" horizontalStyle="SMALL">
      <columnLookup lookupFKUid="C_LU_GENDER|GENDER_CODE"
          lookupNameUid="C_LU_GENDER|GENDER_DISP"/>
   </column>
```

In this example, column C_PARTY|GENDER_CD should be treated as if it has a foreign key to the column C_LU_GENDER|GENDER_CODE, and table C_LU_GENDER is treated as a lookup table. The IDD application creates a drop-down list for the column GENDER_CD, and this list is populated with values from the table C_LU_GENDER (display values are retrieved from the GENDER_DISP column).

Element columnI18NLookup can be specified together with sub-element columnLookup if the localization of display values is needed.

```
<column columnUid="C_PARTY|GENDER_CD" editStyle="FIELD" horizontalStyle="SMALL">
      <columnLookup lookupFKUid="C_LU_GENDER|GENDER_CODE"
         lookupNameUid="C_LU_GENDER|GENDER_DISP"/>
      <columnI18NLookup languageCdUid="C_LU_GENDER_LCL|LANGUAGE_CODE"
         countryCdUid="C_LU_GENDER_LCL|COUNTRY_CODE"
         lookupFKUid="C_LU_GENDER_LCL|GENDER_CODE"
         localizedNameUid="C_LU_GENDER_LCL|LOCALIZED_STRING"/>
   </column>
```

Related Topics:

- "Lookup Tables" on page 22

Lookup Tables With Subtype Column

A single lookup table can be used to store lookup values for several different code types.

In this case, the lookup table has a subtype column that identifies the code type.

Use of a lookup table with many lookup types is configured as shown in the following example.

```
<column columnUid="C_AUTOMOBILE|DOORS_CODE" editStyle="FIELD" horizontalStyle="SMALL">
      <columnLookup lookupFKUid="C_LU_AUTO_ATTR|CODE"
            lookupNameUid="C_LU_AUTO_ATTR|DISPLAY_NAME">
         <subTypeQualifier columnUid="C_LU_AUTO_ATTR|ATTR_TYPE">
            <filter>
               <value>Doors</value>
               <value>Style</value>
            </filter>
         </subTypeQualifier>
      </columnLookup>
   </column>
```

In this example, column C_AUTOMOBILE|DOORS_CODE is a lookup column. Only values in the lookup table with ATTR_TYPE="Doors" are used for this lookup.

Lookup localization can also be combined with lookup sub-types, as shown in the following example.

```
<column columnUid="C_AUTOMOBILE|DOORS_CODE" editStyle="FIELD" horizontalStyle="SMALL">
      <columnLookup lookupFKUid="C_LU_AUTO_ATTR|CODE"
         lookupNameUid="C_LU_AUTO_ATTR|DISPLAY_NAME">
         <subTypeQualifier columnUid="C_LU_AUTO_ATTR|ATTR_TYPE">
            <filter>
               <value>Doors</value>
               <value>Style</value>
            </filter>
         </subTypeQualifier>
      </columnLookup>
      <columnI18NLookup languageCdUid="C_LU_AUTO_ATTR_LCL|LANGUAGE_CODE"
         countryCdUid="C_LU_AUTO_ATTR_LCL|COUNTRY_CODE" lookupFKUid="C_LU_AUTO_ATTR_LCL|
CODE"
         localizedNameUid="C_LU_AUTO_ATTR_LCL|LOCALIZED_STRING">
         <subTypeQualifier columnUid="C_LU_AUTO_ATTR_LCL|ATTR_TYPE "
filterValue="Doors"/>
      </columnI18NLookup>
   </column>
```

Related Topics:

- "Language Codes" on page 149

Static Lookup Values

The values for a lookup column can also be defined directly in the IDD configuration file - no lookup table is used.

The columnStaticLookups element is used to define this, as shown in the following example.

```
<column columnUid="C_PARTY|GENDER_CD" editStyle="FIELD" horizontalStyle="SMALL">
      <columnStaticLookups>
         <columnStaticLookup code="M" name="MALE"/>
         <columnStaticLookup code="F" name="FEMALE"/>
      </columnStaticLookups>
   </column>
```

This example specifies that only values 'M' and 'F' can be stored in the column C_PARTY|GENDER_CD. For this column, the IDD application creates a dropdown list populated with the values 'MALE' and 'FEMALE'.

Static lookup values can also be localized, as shown in the following example.

```
<column columnUid="C_PARTY|GENDER_CD" editStyle="FIELD" horizontalStyle="SMALL">
      <columnStaticLookups>
         <columnStaticLookup code="M" name="MALE"/>
         <columnStaticLookup code="F" name="FEMALE"/>
         <columnStaticLookup code="M" name="MANN" languageCode="de" countryCode="DE"/>
         <columnStaticLookup code="F" name="FRAU" languageCode="de"  countryCode="DE"/>
      </columnStaticLookups>
   </column>
```

Display the Secondary Fields from a Base Object in the Child Tab

To display the secondary fields from a Base Object (BO) in the child tab in IDD, use the **Part Of Primary Object** child type while creating the child subject area (SA) in the IDD Configuration Manager.

You must configure the IDD configuration file (`IDDConfig.xml`) to display the secondary fields from a BO in the child tab.

For the following example, in the hub console you must create a BO `C_EMPLOYEE` with four columns: `EMP_ID`, `EMP_NAME`, `STATE`, and `COUNTRY`, also parent SA `Employee` and Child SA `EmpDetails`.

The following code snippet displays the `EMP_NAME` (which is a secondary field) in the `EmpDetails` child tab.

```
            primaryObjectUid="C_EMPLOYEE" searchOnly="false">
            <subjectArea displayName="Employee" name="Employee" showXREF="false">
                <primaryObject>
                    <layout columnsNum="3">
                        <column columnUid="C_EMPLOYEE|EMP_ID"
                            editStyle="FIELD" editable="true"
                            hidden="false" horizontalStyle="MEDIUM"
                            lineBreak="false"
                            ns1:showInHMCompactView="false"
                            required="false" xmlns:ns1="urn:siperian.dsapp.config"/>
                        <column columnUid="C_EMPLOYEE|STATE"
                            editStyle="FIELD" editable="true"
                            hidden="false" horizontalStyle="MEDIUM"
                            lineBreak="false"
                            ns2:showInHMCompactView="false"
                            required="false" xmlns:ns2="urn:siperian.dsapp.config"/>
                        <column columnUid="C_EMPLOYEE|COUNTRY"
                            editStyle="FIELD" editable="true"
                            hidden="false" horizontalStyle="MEDIUM"
                            lineBreak="false"
                            ns3:showInHMCompactView="false"
                            required="false" xmlns:ns3="urn:siperian.dsapp.config"/>
                    </layout>
                    <label existsFormat="{0}"
                        existsNoAttributesFormat="{0}" newFormat="New {0}"/>
                </primaryObject>
                <poPartOfChild displayName="EmpDetails"
                    name="EmpDetails" ns4:showInHMCompactView="false"
xmlns:ns4="urn:siperian.dsapp.config">
                    <ns4:layout columnsNum="3">
                        <ns4:column columnUid="C_EMPLOYEE|EMP_NAME"
                            editStyle="FIELD" editable="true"
                            hidden="false" horizontalStyle="MEDIUM"
                            lineBreak="false"
                            ns4:showInHMCompactView="false" required="false"/>
                    </ns4:layout>
                </poPartOfChild>
                <search displayPackageUid="PKG_EMPLOYEE"/>
                <dataSecurity/>
            </subjectArea>
        </subjectAreaGroup>
```

Displaying a Parent of a Primary Object in a Child Tab

When a primary object has a parent, you can display the parent base object attributes in a child tab. To configure the display, you must edit the IDD configuration XML file. You can configure multiple child tabs, one for each parent base object that you want to display.

In the MDM Hub, the relationship between the base objects must be a 1:1 or 1:many relationship type. For example, you create base objects C_ADDRESS and C_PARTY and create a relationship between them.

1. In the IDD Configuration Manager, create a subject area for the primary object.
   For example, you create a subject area for C_ADDRESS.
2. Save the configuration.
3. Open the IDD configuration XML file.
4. After the `primaryObject` element, add the `poParent` element and define the fields that you want to display.

   For example, the following code sample shows how to configure the `poParent` element to show three fields from C_PARTY in the child tab.

```
<subjectArea displayName="Address" name="Address" showXREF="false">
    <primaryObject>
  ...
    </primaryObject>
    <poParent name="Party" displayName="Party" uid="C_PARTY"
mpcUid="C_MT_PARTY_ADDRESS">
  <layout columnsNum="3">
        <column columnUid="C_PARTY|FIRST_NAME" editStyle="FIELD"
horizontalStyle="MEDIUM"/>
        <column columnUid="C_PARTY|LAST_NAME" editStyle="FIELD"
horizontalStyle="MEDIUM"/>
   <column columnUid="C_PARTY|PARTY_TYPE" editStyle="FIELD"
horizontalStyle="MEDIUM"/>
  </layout>
 </poParent>
    <search displayPackageUid="PKG_ADDRESS"/>
    <dataSecurity/>
</subjectArea>
</subjectAreaGroup>
```

5. Save the file.

Expand a Child Subject Area in Data View by Default

You can configure a child subject area to expand by default when you open a record in data view.

Set the `expanded` attribute to `true` in `BDDConfig.xml` for the child subject area. When you open the primary record, the child subject area appears expanded. The other child subject areas appear collapsed.

The following code example sets the C_PARTY_NAME subject area to expand by default when you open the primary record in data view:

```
<one2ManyChild name="Names" type="ONE_2_MANY" uid="C_PARTY_NAME"
mpcUid="C_MT_PARTY_NAME" expanded="true">
    <layout columnsNum="1">
        <column columnUid="C_PARTY_NAME|NAME" editStyle="FIELD"
horizontalStyle="MEDIUM"/>
        <column columnUid="C_PARTY_NAME|AUTOMOBILE_ID" editStyle="FIELD"
horizontalStyle="MEDIUM"/>
    </layout>
</ one2ManyChild>
```

Create Sibling Reference

You can create a sibling reference to create a relationship from one record in a subject area to a child record within that subject area. For example, a customer could include both the address and the phone number child records, with the phone number having a foreign key to associate it with a specific address.

You must configure the IDD configuration file (`IDDConfig.xml`) to create the sibling reference.

The following code snippet creates a sibling reference for the column `ADDRESS_ID` field in `PERSON DETAILS` child subject area.

```
<ns10:column
          columnUid="C_PERSON_DETAILS|ADDRESS_ID"
          editStyle="FIELD" editable="true"
          hidden="false" horizontalStyle="MEDIUM"
          lineBreak="false"
          ns10:showInHMCompactView="false" required="false">
       <siblingReference childName="Addresses">
          <label existsFormat=" {1}, {2} "
                    existsNoAttributesFormat="MailingAddress"
                    newFormat="New MailingAddress" taskFormat=" {1}, {2} ">
                         <column columnUid="C_ADDRESS|ADDRESS_LINE_1"/>
                         <column columnUid="C_ADDRESS|CITY_NAME"/>
          </label>
        </siblingReference>
</ns10:column>
```

Note: You must specify the `ChildName` attribute in the `siblingReference` tag with the available child subject area reference name.

Grandchildren

When grandchildren are displayed in a table view, all grandchild records are displayed - not just those that are related to the selected child record. IDD has a configuration option that helps users understand the relationship of these grandchildren to the child.

A parentReference can be defined for the column that is the foreign key to the child record. This defines a label to be displayed in the grandchild record that contains data from the child.

In the example below, the foreign key column from the grandchild to the child is configured as a parent reference. This configures a label element with the set of columns to use for labels and the existsFormat. In this example, the label for the child record will be "<Phone Number>, (<Extension Number>)".

```
<many2ManyChild name="TestPhone" displayName="Test Phone" type="PART_OF"
        uid="C_PHONE_CHILD4" mpcUid="C_MT_PHONE_CHILD4" defaultView="form">
            <layout columnsNum="3">
                <column columnUid="C_PHONE_CHILD4_REL|PHONE_ID"
editStyle="FIELD"
                    horizontalStyle="LARGE">
                    <parentReference>
                        <label existsFormat="{0} ({1})">
                            <column columnUid="C_PARTY_PHONE|PHONE_NUM"/>
                            <column columnUid="C_PARTY_PHONE|PHONE_EXT_NUM"/>
                        </label>
                    </parentReference>
                </column>
                <column … />
            </layout>
    </many2ManyChild>
```

Subject Area Links

A subject area can contain many:many referencing children.

These show a subject area as a child of another subject area. The child subject area cannot be edited directly. The IDD application user must navigate to a separate data view for the child subject area to edit it. The `subjectAreaLinkColumn` element is used to define a column that is to be used as a hotlink.

The data in the column identified as the subject area link is underlined. When the IDD application user clicks on this column, the associated subject area is opened in a new tab.

Whether a subject area link column is configured or not, the IDD application user can right click on the record and select 'Open in a New Tab' to open the subject area.

```
<many2ManyChild name="Organization" displayName="Org" type="REFERENCE"
        uid="C_PARTY" subjectAreaLinkColumn=”C_PARTY_ORGANIZATION_NAME”
        mpcUid="C_MT_ORG_CHILD"  hmEntityTypeUid="Organization">
        <layout columnsNum="2">
            <column columnUid="C_PARTY|ORGANIZATION_NAME" editStyle="FIELD"
                horizontalStyle="LARGE" required="true"/>
            …
        </layout>
    </many2ManyChild>
```

Logical Menu Grouping

If you have multiple subject area groups, you can organize or group them, to create a logical higher level menu structure in the IDD application.

You must edit the IDD configuration file (`IDDConfig.xml`) to create logical groups of subject area groups.

The following code snippet creates a logical grouping of subject area groups.

```
<sagGroups>
       <sagLogicalGroup name="Product" displayName="Product">
              <sagReference sagName="Account" />
              <sagReference sagName="AccountGroup" />
       </sagLogicalGroup>
</sagGroups>
```

Adding Groups Within the New Window

If you have many subject areas, define groups to use in the **New** window in IDD. Set the `enableCreateBeMenuGrouping=true` global property, and then define the groups in the `IDDConfig.xml` file.

1. Set the `enableCreateBEMenuGrouping` property by using the following command:

   ```
   insert into C_REPOS_DS_PREF_DETAIL (ROWID_DS_PREF_DETAIL, Create_Date, creator,
   Last_Update_Date, Updated_By, ROWID_DS_PREF, NAME, VALUE)
   select 'PREF_DET_4', sysdate, 'CMX', sysdate, 'admin', rowid_ds_pref,
   'enabeCreateBeMenuGrouping', 'true' from C_REPOS_DS_PREF where name =
   'SYSTEM_PREFERENCES_ROOT';
   ```

2. In the IDD Configuration Manager, export the `IDDConfig.xml` file, and then add the groups to the file as shown in the following sample:

   ```
   <sagGroups>
   <sagLogicalGroup name="CustomerGroup" displayName="CustomerGroup">
   <sagReference sagName="Customer" />
   <sagReference sagName="Household" />
   </sagLogicalGroup>
   </sagGroups>
   ```

3. Restart the application server.
4. Deploy the modified `IDDConfig.xml` file.

5. Log in to the IDD application, and verify that the **New** window contains the groups.

Customizing Column Labels

You can customize column labels in IDD at subject area level to distinguish identical column label that are used in multiple subject areas or to modify any column label. You must edit the `MetadataBundle.properties` file to customize the subject area column label. For example, consider a base object, Party with subject areas, Person and Organization. If you have the column label, Tax ID in both the subject areas, you can customize the column labels to distinguish between the subject areas.

To customize column labels of a subject area, perform the following steps:

1. If you changed the metadata in the Operational Reference Store, click **Clear Cache**.
2. Export the IDD application to a ZIP file.
3. Extract the application ZIP file.
4. Edit the `MetadataBundle.properties` file.
   For example: To modify column label- Tax ID to Customer Tax ID, in the MetadataBundle.properties, edit Test.Person.COLUMN.C_PARTY|TAX_ID=Customer Tax ID.
5. In the IDD configuration manager, select the IDD application to replace the edited MetadataBundle.properties file.
6. Click the **Import** Button and select **Import to existing IDD application**.
7. In the **Import to existing IDD application** window, for **Configuration type** select **Metadata Bundle**.
8. Click **Browse** to locate and select the appropriate MetadataBundle.properties file.
9. Click **Import**.

   Login to the IDD application to view the customized column labels.

Configure Checkbox Edit Style

Value Mapping allows you to define values that must be stored in the MDM Hub for columns with the checkbox edit style.

The following table provides information about edit styles that you can configure for the supported data type.

Data Type	Edit Style
DATE	Calendar and Long calendar
INT and CHAR(1)	Field, Text area, and Checkbox
Others	Field and Text area

Note:

- For a column of data type CHAR(1), you can define three couples of values that you can setup for a Checkbox: 1/0 value, Y/N value or T/F value. Based on the couple of values assigned, the corresponding value will be saved to the base object.
- For a column of data type INT, you can define only couples of values 0 and 1.

For manual configuration, you must ensure that the `column` element with `editStyle="CHECKBOX"` must not have more than one nested `valueMapping` element. The `valueMapping` element for `editStyle="CHECKBOX"`

must have two nested `mappingItem` elements. Also, the `mapping Item` must include the `selected` values of `true` and `false`.

In the following example, the `domainValue` attribute is responsible for the value that is stored in the MDM Hub and the `selected` attribute is responsible for presentation of the checkbox control. Values `true` or `false` are defined for the selected and unselected states of checkbox, respectively.

```
<column columnUid="C_PARTY_PHONE|IS_VALID_IND" editStyle="CHECKBOX"
horizontalStyle="SMALL">
<valueMapping>
<mappingItem domainValue="1" selected="true"/>
<mappingItem domainValue="0" selected ="false"/>
</valueMapping>
</column>
```

Hierarchy Manager Configuration

The settings described here apply to the IDD Hierarchy View for all Hierarchy Manager (HM) entity types.

The following XML listing has examples of all the items described later in this section.

```
<hmConfiguration hmConfigurationUid="Default|Master" enableAddRel="false"
            simpleNodeLimit="100">
        <hmOneHopLimits totalRels="1000"/>
        <hmManyHopLimits hops="20" relsPerEntity="50" totalRels="1000"/>
        <hmRelationshipTypes>
            <hmRelationshipType hmRelationshipUid="HM_RELATIONSHIP_TYPE.employs">
                <layout columnsNum="2">
                    <column columnUid="C_RL_PARTY|REL_NAME" editStyle="FIELD"
                        horizontalStyle="LARGE" required="true"/>
                    <column columnUid="C_RL_PARTY|REL_DESC" editStyle="FIELD"
                        horizontalStyle="MEDIUM"/>
                    <column columnUid="C_RL_PARTY|NOTE" editStyle="FIELD"
                        horizontalStyle="SMALL"/>
                </layout>
            </hmRelationshipType>
            <hmRelationshipType hmRelationshipUid="HM_RELATIONSHIP_TYPE.contains member">
                <layout columnsNum="2">
                    <column columnUid="C_RL_PARTY_GROUP|HUB_STATE_IND" editStyle="FIELD"
                        horizontalStyle="MEDIUM"/>
                </layout>
            </hmRelationshipType>
        </hmRelationshipTypes>
        <hmFilter name="filter1" displayName="Filter 1">
            <showActiveRelOnly>false</showActiveRelOnly>
            <hideUnconnectedEntities>false</hideUnconnectedEntities>
            <getParents>true</getParents>
            <getChildren>true</getChildren>
            <getUndirected>true</getUndirected>
            <getBidirectional>true</getBidirectional>
            <getUnknown>true</getUnknown>
        </hmFilter>
        <hmFilter name="filter2" displayName="Filter 2">
            <showActiveRelOnly>false</showActiveRelOnly>
            <hideUnconnectedEntities>false</hideUnconnectedEntities>
            <getParents>true</getParents>
            <getChildren>true</getChildren>
            <getUndirected>true</getUndirected>
            <getBidirectional>true</getBidirectional>
            <getUnknown>true</getUnknown>
            <enabledRelationshipsUids>HM_RELATIONSHIP_TYPE.member of account group
                </enabledRelationshipsUids>
            <enabledRelationshipsUids>HM_RELATIONSHIP_TYPE.employs</
enabledRelationshipsUids>
            <enabledRelationshipsUids>HM_RELATIONSHIP_TYPE.contains member 2
```

```
                </enabledRelationshipsUids>
            <enabledRelationshipsUids>HM_RELATIONSHIP_TYPE.customer
                </enabledRelationshipsUids>
            <enabledRelationshipsUids>HM_RELATIONSHIP_TYPE.contains member
                </enabledRelationshipsUids>
            <enabledRelationshipsUids>HM_RELATIONSHIP_TYPE.associate
                </enabledRelationshipsUids>
            <enabledRelationshipsUids>HM_RELATIONSHIP_TYPE.organization has
                </enabledRelationshipsUids>
            <enabledRelationshipsUids>HM_RELATIONSHIP_TYPE.is DNB parent of
                </enabledRelationshipsUids>
            <enabledHierarchiesUids>HM_HIERARCHY.Product</enabledHierarchiesUids>
            <enabledHierarchiesUids>HM_HIERARCHY.Customer</enabledHierarchiesUids>
            <enabledHierarchiesUids>HM_HIERARCHY.DNB</enabledHierarchiesUids>
        </hmFilter>
        <externalLinkAction callback="false" displayName="Graph Google Search"
                name="hm_google_search_action">
            <externalLink name="hm_google_search_link" type="IFRAME"
                    url="http://www.google.com/search">
                <param bddParamName="SELECTED_GRAPH_OBJECTS" name="q" />
                <param name="hl" staticValue="en" />
            </externalLink>
         </externalLinkAction>
        <externalLinkAction callback="true" displayName="Test graph callback"
                name="hm_test_callback_action">
            <externalLink name="hm_test_callback" type="IFRAME"
url="test_external_hm.html">
                <param bddParamName="USERNAME" name="username" />
                <param bddParamName="SELECTED_GRAPH_OBJECTS" name="selectedHmObjects" />
                <param bddParamName="ALL_GRAPH_OBJECTS" name="allHmObjects" />
            </externalLink>
        </externalLinkAction>
    </hmConfiguration>
```

Add Relationships

The Hierarchy View can be configured to be a read-only view.

The IDD application user can navigate through the relationships, but the relationships cannot be added or edited. The enableAddRel attribute that controls this defaults to true. The example above shows how to disable relationship adds and edits.

Rendering Optimization

IDD provides a rich visualization for entities and relationships in the Hierarchy View.

As the size of a graph in this view grows into the hundreds, the time to render this view can be a problem. IDD defines a threshold above which nodes are rendered in a simplified way, thereby decreasing the rendering time. This defaults to 300, but can be manually configured using the simpleNodeLimit attribute.

Hierarchy Manager Relationship Types

Use the hmRelationshipType element to configure relationship types. You can configure layouts, cleanse functions, and user exits for relationships that are added or edited in the Hierarchy View.

The configuration is done per relationship type. There are standard columns for each relationship that are automatically managed by Data Director: hierarchy and relationship type, start and end date, and references to the related entities. The hmRelationshipTypes element specifies any additional attributes on a relationship record.

Note: A Hierarchy Manager relationship that is defined as a foreign key relationship in the Hub Console cannot have custom fields and layout definition in Data Director. This restriction is based on the nature of the

foreign key relationship. For more information, see the section on configuring foreign key relationships between base objects in the *Multidomain MDM Configuration Guide*.

Hierarchy Manager Filter

The Hierarchy View has filters that control which hierarchy and relationship types, relationship directions, and so on are displayed.

Use the hmFilter element to define filter settings that can be assigned as the default filter settings for a subject area. This setting is used as long as an IDD application user has not created a Saved Filter and made that the default for that subject area.

For example, the following code sets `filter2` as the default filter for the subject area `A1`:

```
<subjectAreaGroup displayName="SAG1" name="SAG1" primaryObjectUid="C_TEST"
searchOnly="false">
<subjectArea displayName="A1" name="A1" showXREF="false">
<primaryObject hmEntityTypeUid="A1" hmFilterName="filter2">
...
</primaryObject>
```

Enabling Inactive Relationships

To enable the ability for the user to view inactive relationships in Hierarchy Manager, set `hmInactiveRelationshipsAvailable` to `true`.

To add this parameter to the Oracle database and set the parameter to true, run the following script:

```
insert into CMX_SYSTEM.C_REPOS_DS_PREF_DETAIL
(ROWID_DS_PREF_DETAIL, ROWID_DS_PREF, NAME, VALUE)
select 'INCTR', rowid_ds_pref, 'hmInactiveRelationshipsAvailable', 'true'
from CMX_SYSTEM.C_REPOS_DS_PREF where name = '___SYSTEM_PREFERENCES_ROOT___';
```

Hierarchy View Relationship Table Records

Set the maximum record count to limit the number of relationship records that the Hierarchy View relationship table displays.

The `cmxserver.properties` file contains the parameter sif.api.hm.flyover.max.record.count. The default value is 10,000.

When you do not specify an effective date in the Hierarchy View, the relationship table displays the effective and ineffective relationship records. Many effective and ineffective relationship records might exist for a particular entity. If the total number of relationship records exceeds the maximum record count limit, Informatica Data Director displays the relationship records that are highest in the sort order. Informatica Data Director does not display the relationship records that exceed the maximum record count.

When you specify the effective date in the Hierarchy View, the relationship table displays all the effective relationships for the effective date regardless of the maximum record count limit.

Hierarchy View

In the Hierarchy View, a user can use the View Details command for a selected entity to pop up a dialog that gives a compact view of the entity and some of its child records.

This compactViewChildrenNumber attribute controls how many child records of each type should be shown (this defaults to 5).

The columns and child types that are shown in this view are controlled by the showInHMCompactView attribute on columns and child objects. For the primary object, showInHMCompatctView="true" must be set for any columns that should be displayed. For child objects, showInHMCompatctView="true" must be set for any objects that should be displayed. If this attribute is not set for any columns of the primary object or any children, only the label for the subject area is displayed in this dialog.

```
<subjectArea name="Person">
        <primaryObject hmEntityTypeUid="Person">
            ...
            <layout columnsNum="3">
                <column columnUid="C_PARTY|NAME_PREFIX_CD" editStyle="FIELD"
                        horizontalStyle="SMALL"/>
                <column columnUid="C_PARTY|FIRST_NAME" editStyle="FIELD"
                        showInHMCompactView="true"
                        horizontalStyle="MEDIUM" required="true"/>
                <column columnUid="C_PARTY|MIDDLE_NAME" editStyle="FIELD"
                        showInHMCompactView="true"
                        horizontalStyle="MEDIUM"/>
                <column columnUid="C_PARTY|LAST_NAME" editStyle="FIELD"
                        showInHMCompactView="true"
                        horizontalStyle="MEDIUM" required="true"/>
                <column columnUid="C_PARTY|GENERATION_SUFFIX_CD" editStyle="FIELD"
                        horizontalStyle="SMALL"/>
                <column columnUid="C_PARTY|BIRTHDATE" editStyle="CALENDAR"
                        horizontalStyle="MEDIUM"/>
                </column>
            </layout>
            ...
            <one2ManyChild name="Email" type="ONE_2_ONE" uid="C_PARTY_ELECT_ADDR"
                showInHMCompactView="true"
                mpcUid="C_MT_ELECTRONIC_ADDRESS">
            </one2ManyChild>
            ...
        </primaryObject>
    </subjectArea>Subject Area settings
```

The primary object settings described here control the default behavior when opening a Hierarchy View with an entity of this type as the anchor. The following attributes can be configured.

Attribute	Description
hmManyHopLimits	Controls the graph that is fetched. The default is one hop.
hmFilterName	Initial filter to apply when displaying the graph. The name should be one of the filters defined in the hmFilters described above.
hmDefaultLayout	Layout to use to display the graph. One of the following values: hierarchy, taxonomy, tree, network, circular, explorerView.

```
<primaryObject hmEntityTypeUid="Person" hmFilterName="filter1" hmDefaultLayout="tree">
        ...
        <hmManyHopLimits hops="3" relsPerEntity="50" totalRels="1000"/>
    </primaryObject>
```

Customizations

The Hierarchy View can be customized in the following ways:

- User exits that are executed when adding or modifying relationships
- User exits that can be invoked from the More Actions menu
- Custom actions that can be invoked from the More Actions menu and can pass the context of the graph being viewed

User Interface Extensions

User Interface extensions are used to add custom functionality to an IDD application.

Element	Description
uiExtensions	Can be added to the configuration to add top-level tabs and Start workspace extensions.
externalLinkChild	Can be configured to add child tabs to a subject area.
externalLinkAction	Can be configured to add actions to a subject area, subject area child, or search results.

These extensions are invoked via an URL to which parameters can be passed. These parameters can include the username and password for the logged-in user. These can be passed in clear text or encrypted text through Blowfish symmetric encryption. Use the encryptionKey as an optional element in the bddApplication element.

```
<bddApplication xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      name="AppName"
      displayName="Application Name"
      defaultLocale="en"
      sessionTimeoutMinutes="30"
      xsi:noNamespaceSchemaLocation="./siperian-bdd-config-6.xsd">
    <encryptionKey>secretKey</encryptionKey>
    ...
  </bddApplication>
```

Top-Level Workspace Tabs

By default, Informatica Data Director displays three top level workspace tabs: Start, Data, and Tasks.

Note: You cannot disable the default Start, Data, and Tasks tabs.

Additional tabs can be configured that contain a page requested from an external URL.

Custom Top-Level Tabs

You can add custom top-level tabs to Informatica Data Director.

You can add a tab to display an external link in an iFrame. You cannot use websites such as Google and Facebook with iFrame because of the privacy policy of the websites. Ensure that the external link is for a website that is compatible with iFrames.

The following example code sample adds a Bing search page:

```
http://www.bing.com/search?q=bddUserName&hl=en

<bddApplication …>
    …
    <uiExtensions logicalOrsGroupName="CMX_ORS">
      <topLevelTab name="custom_bing_tab" displayName="Bing Search">
        <externalLink name="bing_username" type="IFRAME" url="http://www.bing.com/search"
            displayName="Bing search">
          <param name="q" bddParamName="USERNAME"/>
          <param name="hl" staticValue="en"/>
        </externalLink>
      </topLevelTab>
      …
    </uiExtensions>
    …
  </bddApplication>
```

Start Workspace

The Informatica Data Director Start workspace consists of three types of components: task list (My Tasks), reports, and custom components.

The task list is always available. This section describes custom components configuration using the externalLink element.

By default, these components are ordered as: task list, reports, and custom components. They can be reordered using the dashboardLayout element described in this section. Informatica Data Director application users can further customize the set of components they will see, and the order in which these components are displayed. This information is saved as part of the user preferences.

External Links (Custom Start Workspace Components)

Custom components are defined using the externalLink element.

An externalLink allows the display of either pages requested from an external URL, or custom HTML and JavaScript code.

The following code snippet is an example of a custom Start workspace component. Two parameters are passed as part of the URL, such as:

```
http://www.bing.com/search?q=bddUserName&hl=en

<bddApplication …>
    …
    <uiExtensions>
      …
      <dashboard>
        <externalLink name="bing_username" type="IFRAME" url="http://www.bing.com/search"
            displayName="Bing search">
          <param name="q" bddParamName="USERNAME"/>
          <param name="hl" staticValue="en"/>
        </externalLink>
        …
      </dashboard>
    </uiExtensions>
    …
  </bddApplication>
```

External Link Parameters (Static and Dynamic)

Any number of parameters can be configured for the URL specified in the externalLink. Parameters can be static or dynamic.

Parameter	Description
Static	Have pre-defined values specified in the IDD configuration file. The following example shows a static parameter definition that uses the staticValue attribute: `<param name="hl" staticValue="en"/>`
Dynamic	Substituted at run time. The definition of a dynamic parameter contains the attribute bddParamName, and the value of this attribute is substituted with data available at run time. The following dynamic parameters are supported: - Login name of the logged IDD application user (bddParamName="USERNAME") - Encrypted login name of the logged IDD application user (bddParamName="USERNAME_ENCRYPTED") - Password of the logged IDD application user (bddParamName="PASSWORD") - Encrypted password of the logged IDD application user (bddParamName="PASSWORD_ENCRYPTED")

External Link Components (IFRAME and IGOOGLE)

Two types of externalLink components are supported: IFRAME and IGOOGLE.

IFRAME

IFRAME components (type= " IFRAME ") display a page requested from external URL. You cannot use websites such as Google and Facebook with iFrame because of the privacy policy of the websites. Ensure that the external link is for a website that is compatible with iFrames.

The URL is constructed from the value specified using the url attribute and the specified URL parameters.

The preceding XML snippet defines an IFRAME component displaying a page requested from a dynamically-generated URL. This URL is constructed from the string " http://www.bing.com/search", the static parameter with name "hl" and the value "en", and the dynamic parameter with name "q" and value substituted at run-time to the name of currently logged IDD application user.

For example, if the logged-in IDD application user has a login name of 'admin', this component displays a page requested from the following URL:

```
http://www.bing.com/search?q=admin&hl=en
```

IGOOGLE

IGOOGLE components (type= " IGOOGLE ") are used to embed JavaScript imported from an external URL (constructed from the value specified using the url attribute and the specified URL parameters) and custom HTML code.

A component defined as `' <externalLink name="component_name" type="IGOOGLE" url="<external URL>"/>'` adds a Start workspace component constructed from a single HTML tag <script>:

```
<script url="external URL"/>
```

Start Workspace Layout

Components in the Start workspace are laid out in a grid - top to bottom, left to right.

By default, these components are ordered as: task list, reports, and custom components.

You can specify the default order using the dashboardLayout element. IDD application users can further customize the set of components they will see, along with the order of these components. This is saved as part of the user preferences.

Conceptually, the Start workspace layout is a grid with *n* columns. Each element can occupy one row and one or more cells in that row. Not all cells of a row must be filled with elements - in that case, the rest of the row will be blank.

The following code snippet shows an example of a two-column Start workspace layout.

```
<dashboardLayout columns="2">
      <dashboardLayoutItem name="my_tasks" type="TASKS" columns="*"/>
                        <dashboardLayoutItem name="xref_composition"    type="REPORT" />
      <dashboardLayoutItem name="igoogle_visualization" type="EXTERNAL_LINK"/>
      <dashboardLayoutItem name="google_username" type="EXTERNAL_LINK"/>
  </dashboardLayout>
```

Each element in the layout is represented with the dashboardLayoutItem element, which has the following possible attributes:

Parameter	Type	Description
name	string	Unique element id inside of the dashboardLayout element.
type	TASKS, REPORT, or EXTERNAL_LINK	Type of the element.
Columns	number or "*"	Number of columns occupied by element. Default value is "1". There is a special symbol "*" for elements that occupy the whole row.

The order of elements on the Start workspace is the order in which they are specified in the dashboardLayout element.

Custom Child Tabs

Custom Child Tabs can be added to a subject area.

These are shown in the same tab panel as One:Many and Many:Many children tabs. They are configured using the externalLinkChild element.

Custom Child Tabs with type externalLinkChild are configured to display the content of an HTML page requested from an external URL. Here is an example of the externalLinkChild definition:

```
<subjectArea name="Organization" displayName="Organization">
      <primaryObject hmEntityTypeUid="Organization">
         <subTypeQualifier columnUid="C_PARTY|PARTY_TYPE" filterValue="Organization"/>
         <layout columnsNum="3">
               <column columnUid="C_PARTY|ORGANIZATION_NAME" editStyle="FIELD"
required="true"/>
         </layout>
      </primaryObject>
      <externalLinkChild name="org_name_bing_search_child" displayName="Bing Search">
         <externalLink name="org_name_bing_search_action_link" type="IFRAME"
            url="http://www.bing.com/search">
            <param name="q" bddParamName="C_PARTY|ORGANIZATION_NAME"/>
                        <param name="hl" staticValue="en"/>
         </externalLink>
      </externalLinkChild>
   </subjectArea>
```

Custom Child Tab Attributes

Custom child tabs are defined using the externalLinkChild element in a subject area.

This element has the following attributes:

Attribute	Description
name	Internally-used name of this custom child tab. Must be unique across all custom child tabs. Use only alphanumeric characters - special characters are not allowed.
displayName	Title of the child tab. The value specified in the configuration XML is used by default, but it can be overridden in the resource bundle.

External Link Properties

Element externalLinkChild must contain the externalLink element, which defines the URL displayed in the child tab.

This element has the following attributes:

Attribute	Description
name	Internally-used name of this link. Must be unique across all external links. Use only alphanumeric characters - special characters are not allowed.
type	External links defined for custom child tabs must have type "IFRAME".
url	URL displayed in the custom child tab.

Parameters

Parameters can be appended to the URL using the param element. URL parameters can be either static or dynamic.

Static Parameters

Static parameters have pre-defined values specified in the configuration.

Here is example of a static parameter definition (which uses the staticValue attribute):

```
<param name="hl" staticValue="en"/>
<param name="loginName" bddParamName="USERNAME"/>
```

Dynamic Parameters

Values of dynamic parameters are substituted at run time.

The definition of a dynamic parameter contains the attribute bddParamName, and the value of this attribute is substituted with the following data available at run time:

- Login name of the logged IDD application user (bddParamName=" USERNAME")
- Encrypted login name of the logged IDD application user (bddParamName="USERNAME_ENCRYPTED")
- Encrypted login name of the logged IDD application user (bddParamName="USERNAME_ENCRYPTED")
- Encrypted password of the logged IDD application user (bddParamName="PASSWORD_ENCRYPTED")
- System column 'ROWID_OBJECT ' of the subject area's PrimaryObject (bddParamName=" <primaryObject TableUID>|ROWID_OBJECT")
- For timeline-enabled PrimaryObjects, the long format in milliseconds of the effective date of the subject area's PrimaryObject (bddParamName="EffectiveDate")
- Data from columns of the subject area's PrimaryObject (bddParamName=" < columnUid of PrimaryObject's column>")
- Data from columns of the subject area's Logical One:One children (bddParamName=" < columnUid of PrimaryObject's One:One child column>")
- You can specify the @LOCALHOST@ and @LOCALPORT@ parameters in the Informatica Data Director configuration file. When a callback externalLinkAction URL points to an application that is deployed on the same server as the MDM Hub, you must dynamically specify the local host name in the URL. Dynamically specify the local host name in the URL so that the externalLinkAction window can interoperate with the

Informatica Data Director browser window without cross-site browser restrictions. The following code shows how to define the externalLinkAction element with the @LOCALHOST@ parameter in the URL:

```
<externalLinkAction callback="false" displayName="View Lineage"
name="per_view_lineage">
<externalLink name="per_view_lineage_link" type="IFRAME" url="http://@LOCALHOST@:
10250/external_app "/>
</externalLinkAction>
```

To pass encrypted user names and passwords, you must set the encryption key. You must define the encryption key in the IDD configuration file (IDDConfig.xml) using the `encryptionKey` element.

The following code sample shows how to define the `encryptionKey` element:

```
<bddApplication xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                name="test"
                displayName="Test BDD application"
                defaultLocale="en"
                sessionTimeoutMinutes="30"
                xsi:noNamespaceSchemaLocation="siperian-bdd-config-6.xsd">
    <description>Description for test ds app configuration</description>
    <configSubVersion>2</configSubVersion>
    <encryptionKey>secretKey</encryptionKey>
...
...
                <externalLinkAction callback="true" displayName="Test callback"
name="person_test_callback_action">
                 <externalLink name="person_test_callback" type="IFRAME"
url="test_external.html">
                  <param bddParamName="SiperianRowID" name="SiperianRowID" />
                  <param bddParamName="EffectiveDate" name="date" />
                  <param bddParamName="USERNAME_ENCRYPTED" name="username" />
                  <param bddParamName="PASSWORD_ENCRYPTED" name="password" />
                  </externalLink>
                </externalLinkAction>
```

For example, in the configuration file, you can define the IDD encryption key as follows:

```
<encryptionKey>{C5869460-4830-4231-9D6E-8A073A97F099}</encryptionKey>
```

Custom Actions

A *custom action* is an HTTP request invoked in a pop-up browser window.

Custom actions can be configured for the following IDD application areas:

- SubjectArea (action definition is placed inside the SubjectArea definition). These custom actions are added in the subject area's More Actions menu (this menu is available in the Data View and Hierarchy View), and in the context menu shown for nodes in the Hierarchy View.
- SubjectArea's Search (action definition is placed inside the SubjectArea's Search definition). These custom actions are added in the search results context menu.
- One:Many and Many:Many children (action definition is placed inside the child definition). These custom actions are added in the child table context menu.
- Hierarchy View (action definition is placed inside the hmConfiguration definition). These custom actions are added to the More Action menu in the Hierarchy View.

Note: Based on the user roles you cannot configure custom actions.

Custom actions are defined using the externalLinkAction element, which has the following attributes:

Attribute	Description
name	Internally used name of this custom action. This name must be unique across all custom actions.
displayName	Text for menu item created for this custom action. The value specified in the configuration XML is used by default, but it can be overridden in the resource bundle.
callback	Attribute should have the value 'true' for the callback action (see below for a description of callback actions).
windowWidth	Width of modal window displaying the result of a callback action. Default value is 700.
windowHeight	Height of the modal window displaying the result of a callback action. Default value is 600.

The externalLinkAction element must contain an externalLink element that defines the custom action URL.

The externalLink element defined for the externalLinkAction supports the same settings as externalLink defined for the externalLinkChild. For more information, see the description of externalLink provided in "Custom Child Tabs" earlier in this document.

As for the custom child tab's externalLink element, externalLink defined for the externalLinkAction supports dynamic parameters substituted at run time. When the action is executed for several records (for example, the IDD application user selects in the search results several records, and executes an action from the search context menu), and action URL has a dynamic parameter substituted with data from the record's columns. The parameter value is constructed from the values of the columns of all selected records, separated by commas. For example, an action is defined for the Organization Search with the following URL definition:

```
<externalLink name="org_name_google_search_action_link" type="IFRAME"
        url="http://www.google.com/search">
      <param name="q" bddParamName="C_PARTY|ORGANIZATION_NAME"/>
      <param name="hl" staticValue="en"/>
   </externalLink>
```

When the IDD application user selects three organizations in the search results with names 'name1', 'name2', 'name3' and executes the action, the action URL will be the following:

```
http://www.google.com/search?q=name1,name2,name3&hl=en
```

Standard Custom Action

A standard custom action opens a new browser window that displays the requested page from an external URL.

Here is an example of a custom action defined for SubjectArea:

```
<subjectArea name="Organization" displayName="Organization">
      <primaryObject hmEntityTypeUid="Organization">
         <subTypeQualifier columnUid="C_PARTY|PARTY_TYPE" filterValue="Organization"/>
         <layout columnsNum="3">
            <column columnUid="C_PARTY|ORGANIZATION_NAME" editStyle="FIELD"
required="true"/>
            …
         </layout>
      </primaryObject>
      <externalLinkAction name="org_name_google_search_action" displayName="Google
Search">
         <externalLink name="org_name_google_search_action_link"
               type="IFRAME" url="http://www.google.com/search">
            <param name="q" bddParamName="C_PARTY|ORGANIZATION_NAME"/>
                  <param name="hl" staticValue="en"/>
         </externalLink>
```

```
        </externalLinkChild>
        …
    </subjectArea>
```

If the IDD application user opens Organization with the name 'Informatica', and selects item 'Google Search' in the 'More Action' menu, IDD opens a window that displays the following URL:

```
http://www.google.com/search?q=Informatica&hl=en
```

Custom Action with Callback

A custom action can also include a callback.

This is useful when the external process invoked by the custom action can modify data in the subject area. After making this modification, the custom action can invoke the callback to instruct the IDD application to refresh the subject area.

IDD defines a JavaScript function named refreshObject to refresh the subject area. This function requires one parameter - the internal IDD ID of the modified record. To make this ID available to external applications, the custom action's HTTP request should pass it as a parameter (in this case, the external application can get this ID from a request and pass it back to the IDD application). To add an internal record ID to an action's URL, a dynamic URL parameter with bddParamName='SiperianRowID' should be added to the URL definition (see the example of a callback action definition later in this section).

When a callback custom action is invoked, IDD opens a modal window containing the element <iframe>, which displays the HTML page received as a result of the action's HTTP request. This HTML page is able to call the refreshObject function using the following JavaScript code:

```
var modifiedRecordID = // get modified record ID from HTTP request
    var opener = window.parent.dialogArguments;
    opener.refreshObject(modifiedRecordID);
```

The modal window where the result of the action's request is displayed can be accessed from JavaScript as window.parent. For example, an HTML page generated as a response to an action can contain the following JavaScript function, which closes the action's modal window and refreshes IDD views:

```
function closeWindowAndRefreshBDD() {
      var modifiedRecordID = // get modified record ID from HTTP request
      var opener = window.parent.dialogArguments;
      opener.refreshObject(modifiedRecordID);
      window.parent.close();
   }
```

Important Note: Due to browser security restrictions, the HTML page is able to call the JavaScript function defined in the IDD application only if this page is located in the same domain as the IDD application (this page is served by the same application server in which the IDD application is deployed).

Here is an example of the callback action defined for SubjectArea:

```
<subjectArea name="Organization" displayName="Organization">
      <primaryObject hmEntityTypeUid="Organization">
         <subTypeQualifier columnUid="C_PARTY|PARTY_TYPE" filterValue="Organization"/>
         <layout columnsNum="3">
            <column columnUid="C_PARTY|ORGANIZATION_NAME" editStyle="FIELD"
required="true"/>
            …
         </layout>
      </primaryObject>
      <externalLinkAction callback="true" name="organization _callback_action"
            displayName="Org Callback">
         <externalLink      name="org_name_google_search_action_link"
               type="IFRAME" url="http://external/application/url">
            <param name="InternalID" bddParamName="SiperianRowID"/>
            <param name="organization_id" bddParamName="C_PARTY|ROWID_OBJECT"/>
         </externalLink>
      </externalLinkChild>
```

```
        …
</subjectArea>
```

If an IDD application user opens an Organization with ROWID_OBJECT=1222 and then invokes this custom action, IDD opens a modal window displaying the page requested from the following URL:

```
http://external/application/url?InternalID=BASE_OBJECT.C_PARTY|1222&organization_id=1222
```

This page can then call the IDD application's refreshObject JavaScript function with the parameter 'BASE_OBJECT.C_PARTY|1222' (this is internal ID of the opened Organization record), which causes the IDD application to refresh all views opened for this record.

Security for Custom Extensions

Access to custom child tabs and custom actions is controlled through SAM.

When an IDD application is deployed, Custom Resources are created for each custom child tab and custom action defined in the IDD configuration. Privileges for these resources should be configured using the Hub Console.

Custom Child Tabs

For custom child tabs, resources are named as follows:

```
CUSTOM_EXTENSION/CUSTOM_CHILD_TAB:<name>
```

where <*name*> is the unique name of the child tab as specified in the configuration.

A custom child tab is visible if the IDD application user has READ privileges to the corresponding tab resource.

Custom Actions

For custom actions, resources are named as follows:

```
CUSTOM_EXTENSION/CUSTOM_ACTION:<name>
```

where <*name*> is the unique name of the action as specified in the configuration.

A custom action is shown and can be executed if the IDD application user has EXECUTE privilege for the corresponding action resource.

User Exits

User exits provide a means to add custom business logic to standard Informatica Data Director operations. You can use user exits within the Data workspace.

User exits are implemented in Java. For details on the interfaces used to implement user exits, see the Javadoc for `siperian-bdd.jar` that is included in the MDM Hub Resource Kit. Also included in the Resource Kit is a set of example user exits. The set of user exits includes an ant project that you can use as a template to build a user exit JAR file.

User Exits and the Entity 360 Framework

User exits are not supported for use with workspaces that are built on the Entity 360 framework, such as the **Start** workspace and the entity workspace.

With the Entity 360 framework, you can use cleanse functions and server-side validation to replace some of the functionality of user exits. For more information, see the *Multidomain MDM Provisioning Tool Guide*.

Note: For backward compatibility, you can continue to use user exits with the **Data** workspace. To display the **Data** workspace, enable the `cmx.dataview.enabled` property in the `cmxserver.properties` file. For more information, see the *Multidomain MDM Configuration Guide*.

User Exit Operations

User exits have defined operations and entry points.

For each subject area, you can implement user exits to add custom functionality for the following operations:

- Save
- Send for Approval
- Task Operations
- Merge
- Mark Not A Match
- Custom Operations
- HM Save Relationship
- HM Custom Operations
- Open

The following table describes the user exit entry points available for each operation. Save, Send for Approval, and Task Operations are variations on the process of saving changes to the subject area data view and provide the same set of entry points.

Operation	Entry Point	Description
Save, Send for Approval, Task Operations	beforeValidation	**Note:** This entry point is no longer supported. Use the beforeEverything entry point instead.
	afterValidation	**Note:** This entry point is no longer supported. Use the beforeEverything entry point instead.
	beforeEverything	Called before any processing is done. Use this to perform custom validation or augmentation of the data in the subject area. Informatica Data Director saves changes that the user exit makes to the data in the subject area. Can report Errors, Warnings, and Confirmations. Can set start and end dates for a period. Runs outside the save transaction.

Operation	Entry Point	Description
	beforeSave	Called after the search for duplicates, just before performing the composite save. Use this to run custom business logic that augments the data in the subject area. Informatica Data Director saves changes that the user exit makes to data in the subject area. Can report Errors. Runs as part of the composite save transaction. SIF requests to the Operational Reference Store are part of this transaction.
	afterSave	Called after the subject area changes are saved. Use this to perform maintenance of data that is not part of the subject area. Can report Errors that roll back the transaction. Runs as part of the composite save transaction. SIF requests to the Operational Reference Store are part of this transaction.
	afterEverything	Called after the save transaction is committed. Use this to provide user notifications or perform maintenance of data that is not part of the subject area when the changes cannot be executed as part of the transaction. Can report Warnings. Runs outside the save transaction.
Merge	beforeEverything	Called before any processing is done. Use this to perform custom validation or augmentation of the data in the subject area. Can report Errors, Warnings, and Confirmations. Can set start and end dates for a period. Runs outside the save transaction.
	beforeMerge	Called just before the merge is performed. Use this to run custom business logic to provide error or confirmation messages. Can report Errors. Runs as part of the merge transaction. SIF requests to the Operational Reference Store are part of this transaction.
	afterMerge	Called after the merge operation has completed. Use this to perform maintenance for data that is not part of the subject area. Can report Errors that roll back the merge. Runs as part of the merge transaction. SIF requests to the Operational Reference Store are part of this transaction.
	afterEverything	Called after the merge transaction is committed. Use this to provide user notifications or perform maintenance of data that is not part of the subject area when the changes cannot be executed as part of the transaction. Can report Warnings. Runs outside the transaction.

Operation	Entry Point	Description
Mark Not A Match	beforeEverything	Called before any processing is done. Use this to perform custom validation or augmentation of the data in the subject area. Can report Errors, Warnings, and Confirmations. Can set start and end dates for a period. Runs outside the save transaction.
	beforeMarkNotAMatch	Called just before the not a match is performed. Use this to run custom business logic to provide error or confirmation messages. Can report Errors. Runs as part of the not a match transaction. SIF requests to the Operational Reference Store are part of this transaction.
	afterMarkNotAMatch	Called after the not a match operation is completed. Use this to perform maintenance for data that is not part of the subject area. Can report Errors that will roll back the merge. Runs as part of the not a match transaction. SIF requests to the Operational Reference Store will be part of this transaction.
	afterEverything	Called after the not a match transaction has been committed. Use this to provide user notifications or perform maintenance of data that is not part of the subject area when the changes cannot be executed as part of the transaction. Can report Warnings. Runs outside the transaction.
User Operation	processOperation	Called when the Informatica Data Director user invokes the custom operation user exit from the More Actions menu in the data view. Use this to run custom business logic. The user exit can return Error or Warning messages. The data view is refreshed if this completes without error so that any changes to the subject area made by the user exit are reflected in Informatica Data Director.
HM Save Relationship	beforeEverything	Called before any processing is done. Use this to perform custom validation or augmentation of the relationship. Can report Errors, Warnings, and Confirmations. Can set start and end dates for a period. Runs outside the save transaction.
	afterValidation	Called after validation and cleanse function execution are performed. Use this to perform custom validation or augmentation of the relationship. Can report Errors, Warnings, and Confirmations. Runs outside the save transaction.

Operation	Entry Point	Description
	beforeSave	Called just prior to performing the save. Use this to run custom business logic that augments the data associated with the relationship. Can report Errors. Runs as part of the save transaction. SIF requests to the Operational Reference Store are part of this transaction.
	afterSave	Called after the relationship changes are saved. Use this to perform maintenance of data associated with the relationship. Can report Errors which roll back the save. Runs as part of the save transaction. SIF requests to the Operational Reference Store are part of this transaction.
	afterEverything	Called after the save transaction is committed. Use this to provide user notifications or perform maintenance of data associated with the relationship when the changes cannot be executed as part of the transaction. Can report Warnings. Runs outside the save transaction.
HM User Operation	processOperation	Called when the Informatica Data Director user invokes the custom operation user exit from the More Actions menu in the data view. Use this to run custom business logic. The user exit can return Error or Warning messages. The user exit indicates which parts of the graph need to be refreshed as a result of the user exit operation.
Open	beforeOpen	Called before an open operation is performed. Use this to mark columns as read-only in edit mode and to overwrite column values. Can report errors, warnings, confirmations, and custom messages. Runs outside the open transaction
	afterOpen	Called after the open operation is completed. Use this to send various notifications to the data in the subject area. Also, you can use this for custom checking of the data that is loaded to the database. Can report errors, warnings, confirmations, and custom messages. Runs as part of the open transaction. SIF requests to the Operational Reference Store are part of this transaction.

Each user exit is supplied with the following data, which is described in detail in the Javadoc:

- the subject area data that is being operated on
- a SiperianClient object that can be used to perform SIF operations against the Operational Reference Store database, plus the Operational Reference Store ID and user credentials to use in SIF requests
- operation-specific data

Building User Exits

The basic steps for building user exits for an IDD application are:

1. Develop the user exit Java code.
2. Compile and build a jar file containing the user exit classes.
   Use `siperian-bdd.jar` from the MDM Resource Kit. This archive contains all IDD specific classes and interface definitions that are required for building User Exit implementation. For more information, see the *Multidomain MDM Resource Kit Guide*.

   Note: The jar file must be named `UserExitsImplementation.jar`.
3. Use the IDD Configuration Manager to import the JAR file into your IDD application. (You can also include the JAR file in an IDD application ZIP file that is imported.)
4. Register user exit classes with the subject area.
5. Deploy the IDD application.

Configuring a User Exit

User exits are configured by subject area.

A subject area can have user exits defined for each of the user exit operations described earlier in this section.

```
<subjectArea name="Organization" displayName="Organization">
      <primaryObject hmEntityTypeUid="Organization">
         <subTypeQualifier columnUid="C_PARTY|PARTY_TYPE" filterValue="Organization"/>
         <layout columnsNum="3">
            <column columnUid="C_PARTY|ORGANIZATION_NAME" editStyle="FIELD"
required="true"/>
            …
         </layout>
      </primaryObject>
      …
      <userExits className="com.siperian.bdd.userexits.sample.SaveHandler"/>
      <userExits className="com.siperian.bdd.userexits.sample.SendForApprovalHandler"/>
      <userExits className="com.siperian.bdd.userexits.sample.CustomActionProvider"
         actionName="Custom User Exit"/>
   </subjectArea>
```

The following code snippet is an example of configuring ClassName for the HM Save Relationship User Exits in the IDDConfig.xml file.

```
<hmRelationshipTypes>
<hmRelationshipType hmRelationshipUid="HM_RELATIONSHIP_TYPE.contains member">
<layout columnsNum="2">
<column columnUid="C_RL_PARTY_GROUP|HUB_STATE_IND" editStyle="FIELD"
horizontalStyle="MEDIUM"/>
</layout>
<userExit className="com.siperian.bdd.userexits.sample.HMRelationshipSaveHandler"/>
<userExit className="com.siperian.bdd.userexits.sample.HMRelationshipHandler"/>
</hmRelationshipType>
</hmRelationshipTypes>
```

Note: Based on the user roles you cannot configure user exits.

Configuring a User Exit to Set Start Date and End Date for a Period

To set the start date and end of a period in Informatica Data Director, you can use the `IEffectivePeriodSetters` interface in the beforeEverything user exit.

Note: The `IEffectivePeriodSetters` interface only works for Save handler. It does not work for HM relationship Save handler.

The following code snippet is an example of how to set the effective start and end date:

```
// get existing start end date from effective period
IEffectivePeriodSetters epd = ((IEffectivePeriodSetters)
getOperationContext().getValue(OperationContext.EFFECTIVE_PERIOD));
Date stDate = epd.getStartDate();
Date eDate = epd.getEndDate();
// set new effective start and end date
epd.setPeriod(DateUtils.addDays(stDate, 1), DateUtils.addDays(eDate, -1));
```

User Exit Messages

User exits can return a message (as an error, warning or confirmation) to be displayed to the user.

These messages are handled by IDD in the same way that it handles its own messages. Each message has a code that is a key to the resource bundle ErrorCodeBundle.properties. IDD finds the error level (error, warning or confirmation) and the text of the message in this resource bundle.

Note: Be sure to use unique codes for any custom messages.

These message strings can be localized just as other strings can be localized.

Messages can have parameters that are replaced with data specified in the user exit. These parameters are handled using the Java MessageFormat class.

The format for the messages in ErrorCodeBundle.properties is:

```
error code=error level|title|main message[|secondary message]
```

where

Item	Description
Error code	Unique key for the message.
Error level	One of the following values: ERROR, WARNING, or CONFIRMATION.
Title	Title for the dialog box. The title should describe the location and context in which the problem occurred. If not specified, the title will be 'Informatica Data Director'.
main message	Main error message. This text should describe the problem from the IDD application user's point of view - not an internal technical point of view. For example, something like `"Problem saving xxx", not "Put error"`.
Secondary message	Secondary part of the message telling the IDD application user what to do about the problem. In the dialog box, this part will be separated from the main message by at least one blank line. This message should not be too long.

Troubleshooting

When trying to understand why a user exit is not operating properly, use the following standard tools.

Tool	Description
Logs	Exceptions generated in the user exit can be found in the Informatica MDM Hub logs. The user exit can also make entries in the log using log4j, as shown in the sample user exits.
Debugger	Java debugger can be used to step through the execution of the code. This is done as you would debug any Java application deployed in an application server environment.

Localization

Resource bundles contain the strings that display in an Informatica Data Director application.

There are four sets of resource bundles:

- BDDBundle
- ErrorCodeBundle
- MessagesBundle
- MetadataBundle

Each set includes the default file, a placeholder English language file, and localized versions of the file, if any exist.

For example, for the MessagesBundle set, there is the default file `MessagesBundle.properties` and the placeholder English language file `MessagesBundle_en.properties`.

Each resource bundle file is a UTF-8 encoded properties files. Each entry in the file is a name/value pair like <*name*>=<*value*>.

- <*name*> is a fixed value that is referenced by the Informatica Data Director application. You cannot change this.
- <*value*> is the part that you can localize.

A few examples:

```
title=Business Data Director
locale=Locale
search=Search
```

To add message bundle files to the Informatica Data Director application, you can include them in the application .zip file that you import. Alternatively, import message bundle files directly into an existing application in Informatica Data Director.

Note: In the localized `MetadataBundle.properties` file, avoid spaces in the names of Hierarchy Manager relationship types and Hierarchy types. Informatica Data Director replaces spaces with underscores when it displays these localized values.

When you first create an Informatica Data Director application, the Informatica Data Director Configuration Manager generates default resource bundles of each type. These resource bundles have entries for all of the labels used in the Informatica Data Director application.

To change or localize these resource bundles, perform the following steps:

1. Export the Informatica Data Director application.
2. Extract the files from the application .zip file.
3. Create a resource bundle with the appropriate ISO language code suffix of your chosen language.
4. In your chosen language, edit the labels in the resource bundle.

   Note: To localize the labels of subject area groups, subject area and logical menu group names, use the `BDDBundle.properties` file with the appropriate language code suffix.
5. Repeat steps 3 through 4 for each resource bundle you want to localize.

Setting the Login Page and Configuration Manager Default Display Language

The language of your web browser dictates the display language for the Informatica Data Director login page and Configuration Manager. You can run a script to set the language that displays in the login page and the Configuration Manager user interface.

The script does not set the default display language of the Informatica Data Director application. You can set the display language of the Informatica Data Director application from the Change Language menu option under your user name. When you set the login page and Configuration Manager default display language, Informatica Data Director ignores the language setting of your web browser.

1. Run the following script to set the language code for the globalLocale parameter:

```
INSERT
INTO CMX_SYSTEM.C_REPOS_DS_PREF_DETAIL
  (
    ROWID_DS_PREF_DETAIL,
    CREATE_DATE,
    CREATOR,
    LAST_UPDATE_DATE,
    UPDATED_BY,
    ROWID_DS_PREF,
    NAME,
    VALUE
  )
  VALUES
  (
    'MST1.5AB',
    sysdate,
    'admin',
    sysdate,
    'admin',(SELECT ROWID_DS_PREF
FROM CMX_SYSTEM.C_REPOS_DS_PREF where name = '___SYSTEM_PREFERENCES_ROOT___'),
    'globalLocale',
    '<ISO language code>'
  );
```

   The ISO language code is a two-letter code that represents the language. For example, the country code 'ja' represents Japanese. If you do not set a valid ISO language code, the display language is English.

2. Restart the application server.

Custom Error Pages

You can configure Informatica Data Director (IDD) to display custom error pages instead of error messages from the application server. For example, when a user enters an incorrect URL, you can configure IDD to redirect the user to the login page or a more user-friendly error page.

To create custom error pages, edit the `web.xml` file and configure the page to appear when a user encounters an error in an IDD session.

The `web.xml` file is in the following location:

```
<infamdm installation directory>/hub/server/siperian-mrm.ear/zds-gui.war
```

Configuring a Custom Error Page

To create custom error pages, edit the `web.xml` file and configure the page to appear for a particular error code.

1. Extract the files from the `zds-gui.war` directory.

   The directory contains multiple files, including web.xml.

2. Use a text editor to edit the `web.xml` file.
   In the following example, the 404 HTTP response from the application redirects the user to the `error_custom.html` page.

   ```
   <error-page>
   <error-code>404</error-code>
   <location>/error_custom.html</location>
   </error-page>
   ```

   Note: To ensure the custom page appears for users, add the `error_custom.html` page to the zds-gui.war directory.

3. Save the changes to the `web.xml` file and then redeploy the IDD application.

Online Help

By default, an Informatica Data Director (IDD) application includes the User Guide help. You can also add custom help.

User Guide Help

The User Guide help describes the tasks that you can perform with an IDD application. For example, the help tells you how to add a business entity or merge business entities. The IDD application developer can replace the shipped help file with a revised help file. Localized versions of the help file are also available. If you change the locale for an IDD application, the application displays the help in the same language.

Custom Help

Custom help describes the business entities or subject areas that are defined in the application. The IDD application developer creates custom help and adds the custom help to the application.

Data Director User Guide

The User Guide describes the tasks that business users can perform in Data Director (IDD). For example, the guide describes how to add a business entity or merge business entities.

By default, Data Director includes the User Guide as an online help file. The IDD application developer can replace the shipped help file with a revised help file. Revised help files are available from the Informatica Network.

Downloading a Revised User Guide Help File

You can find and download revised User Guide help files from Informatica Network.

1. In a browser, open Informatica Network.
2. Search for "**Informatica Data Director User Guide Help**".

If you see **Informatica Data Director User Guide Help** in the results, a revised help file is available for the specified version.

3. Select the link.
4. Make a note of the Revision Number. You can use this number to verify that the correct help appears.
5. Download the help file.

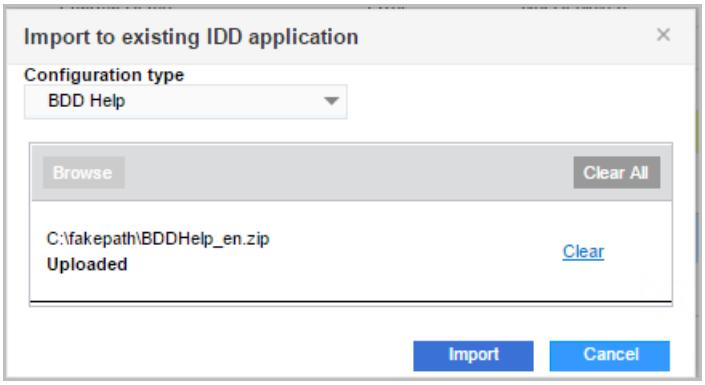
Importing a Revised User Guide Help File

You can import a revised help file into your IDD applications.

The name of the help file is in the form `BDDHelp_xx.zip` file, where `xx` is an ISO language code. If you support multiple languages, import the localized help files for each language you support. The localized help appears when a user selects the locale in the IDD application.

1. Log in to the IDD Configuration Manager.
2. Select an application.
3. Click **Import** > **Import to existing IDD application**.
4. In the **Import to existing IDD application** dialog box, select **BDD Help** from the **Configuration type** list.
5. Click **Browse**.
6. In the **Open** dialog box, select the revised help file, and click **Open**.

   The following image shows the English version of the help file is ready to import:



7. Click **Import**.

   The import process updates the application with the revised User Guide help.

Testing the Revised Help

After you import a revised help file, open the application and verify that the help displays the correct revision number.

1. If the application is open, close the application.
2. Log in to Informatica Data Director.
3. If prompted, select the application that contains the revised help.
4. From the **Help** menu, click **Help**.
5. Verify that the Revision Number at the bottom of the Welcome topic matches the number associated with the help file that you downloaded.

Custom Help

You can create custom help that describes the business entities or subject areas that you defined in the IDD application. After you import the custom help and deploy the application, a **Custom Help** menu item appears in the **Help** menu.

If you support multiple languages, you can create localized help files for each language that you support. When a user selects the locale in the IDD application, the localized custom help is used.

Creating a Custom Help File

You can create a custom help file to document your IDD applications. If you support multiple languages, you can also create localized help files for each language that you support.

1. In an HTML authoring tool, create the custom help topics and generate the help project.
2. Rename the `index.htm` file to `bdd_help_CSH.htm`.
3. Create a directory named `bdd_help`.
4. Copy the generated help directories and files to the `bdd_help` directory.
5. Select the `bdd_help` directory and create a .zip file that preserves the directory structure.
6. Name the .zip file `CustomBDDHelp_xx.zip`, where `xx` is a two-character ISO language code.
7. Verify that the size of the `CustomBDDHelp_xx.zip` file is 20 MB or less.

Importing a Custom Help File

You can import a custom help file into IDD applications. If you localized the custom help, import the localized help files as well.

1. Log in to the IDD Configuration Manager.
2. Select an application.
3. Click **Edit**.
4. In the Edit Application panel, select the **Custom Help** check box, and then click **Save**.

   In the application configuration file, the `help` property is updated to set `customBddHelp` to `true`:

   ```
   <help bddHelp="true" customBddHelp="true"/>
   ```
5. In the navigation tree, click **Applications**.

   The list of applications appears.
6. Select the same application.
7. Click **Import** > **Import to existing IDD application**.
8. In the **Import to existing IDD application** dialog box, select **Custom BDD Help** from the **Configuration type** list.
9. Click **Browse**.
10. In the **Open** dialog box, navigate to and select the `CustomBDDHelp_xx.zip` file, and click **Open**.
11. Click **Import**.

    The import process updates the application with the custom help file.
12. Click **Redeploy**.

CHAPTER 6

IDD Global Properties

This chapter includes the following topics:

Informatica Data Director Global Properties Reference

The following table lists global properties that control runtime behavior of all the Informatica Data Director (IDD) applications on a single Hub server.

The table describes each properties and its default value. These properties are stored in the CMX_SYSTEM.C_REPOS_DS_PREF_DETAIL table. If properties are not defined then the specified default values are used.

Important: The application server must be restarted for changes to the following global properties to take effect.

Property	Default value	Usage
allowDsEmptyChildren	false	Determines if users can view child records when you configure a security filter on a grandchild column, but there are no grandchild records. If `true`, users can view the child records when there are no grandchild records. If `false`, users cannot view the child records when there are no grandchild records.
asyncChildLoading	false	Loads the child data in the Data View, when you explicitly open the child record of the primary object. You can set the property value to true to load the children data when you open the record in the Data View.

Property	Default value	Usage
bulkexportloadsize	500	Maximum size of the load for each thread when you export data to a Microsoft Excel file. Default is `500` records. The maximum is `1000` records. If set to greater than `1000`, the default load size is used.
CompositePagerTotalRecords	500	Maximum number of ActiveVOS tasks that IDD sorts case-insensitively at a time. If tasks exceed the set value, IDD sorts ActiveVOS tasks according to the database type. If the database is Microsoft SQL Server, the sorting is case insensitive. If the database is Oracle or IBM Db2, the sorting is case sensitive.
convert2DigitYearTo4Digit	false	Enables the adjustment of a two-digit year entry to a four-digit year entry. Set to `true` to enable the adjustment of entered dates to 80 years before and 20 years after the current date. For example, if you enter `1/Jan/30` as the date, IDD interprets the entry as January 1, 2030. If you enter `1/Jan/70` as the date, IDD interprets the entry as January 1, 1970.
credentialsAutofillDisabled	false	For security reasons, if you want to control user's browser from saving the login credentials such as username and password, you can set this value to 'true'.
CSVColumnSeparator	Comma (,)	Determines the character to use as a column separator when you export the data to a comma-separated values (CSV) file. You can also use a tab, semicolon, and space as a delimiter.
deleteMovedRelInExplorerView	true	Determines whether to delete the old relationship when you create a new relationship in Hierarchy Manager explorer view. Set to false to end-date the old relationship.
enableCreateBEMenuGrouping	false	Specifies whether you can define logical groups for the **New** window. Required if you have a large number of subject areas. Set to `true` to define logical groups for the **New** window. Set to `false` if you do not want to define logical groups for the **New** window.

Property	Default value	Usage
enableRememberCredentials	true	When true, the Remember Me checkbox is displayed on the login page. Users remain logged in for the period determined by rememberCredentialsPeriod.
enableSaveForPeriodDialogForHmRel	true	Enables the effective period dialog box that appears when you update a Hierarchy Manager-enabled record in IDD. Set to `false` to disable the effective period dialog box.
enableTaskAttachments	false	Specifies whether users can attach files to tasks when using the legacy views with Data Director. Set to `false` to disable attachments and to hide the **File Attachments** section in the **Task Details** dialog box and in the **Create Task** dialog box. Set to `true` to enable attachments. **Important:** To see your change in the IDD application, use the IDD Configuration Manager to clear the IDD application cache.
expandDropDownListShowFullValue	false	Enables expansion of the drop-down list in the **Search** tab for lookup records. Set to `true` to allow the list to adjust to accommodate the longest list item.
exportusingmultithread	false	Enables multithreading for data export to a Microsoft Excel file. Set to `true` to enable multithreading for data export.
handleUserExitBeforeShowingDialog	false	Determines when IDD calls the SendForApprovalHandler user exit. Set to `true` to have IDD call the SendForApprovalHandler user exit when the user clicks **Send for Approval**. Set to `false` to have IDD call the SendForApprovalHandler user exit when the user clicks **Ok** from the **Send for Approval** dialog box.
HeaderBgColor	#000000	Specifies the HTML color code of the background color of the IDD header area.

Property	Default value	Usage
hideSystemColumnsInResult	false	Specifies whether to show system columns in IDD search results. Set to `true` to hide system columns in IDD search results. When `true`, you can still customize the table view to manually select system columns to display.
hmInactiveRelationshipsAvailable	false	Set to `true` so the user can view inactive relationships in Hierarchy Manager.
IDD2COCSConverter.prefixCoNames	false	When the Informatica Data Director configuration is converted to a business entity configuration, determines if a prefixed subject area name is used for the business entity name. Set to `false` to use the subject area name as the business entity name. Set to `true` to use the subject area name prefixed with the Informatica Data Director application name as the business entity name.
isEffectiveDateIncluded	false	Specifies whether to include the Effective Date field for search queries in the Informatica Data Director. Set to `true` to display the current date in the Effective Date field. Set to `false` to hide the Effective Date field.
isFillOnGap	false	Specifies whether to enable the Fill on Gap property for operations in Informatica Data Director. Set to `true` to enable the Fill on Gap property. Set to `false` to disable the Fill on Gap property.
lookupCacheUpdatePeriod	300000 (5 min)	The number of milliseconds that lookup data can be in the IDD cache before it will be reloaded.
minModalWidth	1100	Determines the minimum width in pixels of the **Search** window.
maxCopiedChildrenNumber	10	Determines the maximum number of child records for each child type that are copied when a user copies a subject area.

Property	Default value	Usage
maxCopiedGrandChildrenNumber	10	Determines the maximum number of grandchild records for each child type that are copied when a user copies a subject area.
maxImportThreads	5	Determines the maximum number of threads to use during data import.
maxParallelPromoteThreads	1	Determines the maximum number of threads to use when you approve a task. When maxParallelPromoteThreads is greater than 1 and you promote records from multiple base objects, the promote process runs in parallel. The maximum value of maxParallelPromoteThreads is equal to the number of server CPU cores.
maxParallelISavedQueriesThreads	true	Determines whether queries load through multiple threads. Multi-threaded queries load faster. Set to `true` to enable multi-threading. Set to `false` to disable multi-threading.
maxParallelBvtThreads	1	Determines the maximum number of threads to use when IDD loads a task to view.
maxSearchResultsExportedRows	5000	Maximum number of rows of search result data that will be exported.
maxXrefSearchReturnCount	100	Specifies the maximum number of cross-reference records that a search request returns.
needLoadChildOnOpen	false	Set to `true` to initially display only the parent records in the Matches view. Child records are displayed as you expand the child record tabs. Set to `false` to initially display the parent records and all child records in the Matches view.
openDashboardAfterTaskClose	false	Set to `true` for Informatica Data Director to open the Start workspace after you complete any task. Set to `false` for Informatica Data Director to open the previous tab in Data View after you complete any task.

Property	Default value	Usage
overrideTextAreaColumnOrder	true	By default, if you configure a column as a text area in a subject area, the text area column always appears at the bottom of the layout regardless of column order. Set to `true` to ensure text area columns in a subject area appear at the bottom of the layout regardless of column order. Set to `false` to ensure text area columns appear in their specified order in a layout.
proactiveMatchResultSort	sortbyscorethenaction	Specifies the sort order in which the potential matches appear. Set to `sortbyscorethenaction` to sort by match scores and then by action. Set to `sortbyactionthenscore` to sort by action, such as open and import, and then by match scores.
qrytaskidfromprocessidtotalretry	2	Number of attempts that IDD makes to reload an ActiveVOS task. Set to a higher integer value if you use a user exit to handle ActiveVOS tasks and tasks are not appearing correctly in IDD.
qrytaskidfromprocessidwaitintrvlmillis	1000	Number of milliseconds that Informatica Data Director waits before attempting to reload an ActiveVOS task. Set to a lower integer value if you use a user exit to handle ActiveVOS tasks and tasks are not appearing correctly in IDD.
rememberCredentialsPeriod	24 (hours)	Length of time (in hours) that user credentials are remembered if the 'Remember Me' checkbox is selected.
samCacheUpdatePeriod	600000 (10 min)	Determines how long (in milliseconds) SAM roles (resources with privilege assignments) can be in the IDD cache before they will be reloaded.
serverPageSize	100	Affects paging of search results and child data. IDD displays to the user a page of 10 records. However, the number of records it fetches from the MDM Hub is determined by this property. With the default setting, IDD will not request additional data until the user goes to the 11th page of data.

Property	Default value	Usage
search_empty_date	false	Determines if the effective date field in the search dialog box is empty or contains the Data view effective date when you create a child record. Set to `true` to have an empty effective date field. Set to `false` to have the Data view effective date populate the effective date field.
searchForDuplicatesBeforeTaskDialog	false	Determines if the **Potential Duplicates** dialog box appears before or after you send a task for approval. Set to `true` to have the **Potential Duplicates** dialog box appear before the **Create Task** dialog box appears. Set to `false` to have the **Potential Duplicates** dialog box appear after you click **Ok** from the **Send for Approval** dialog box.
shouldDisableSearchFieldIfDependentFieldAbsence	false	Enables or disables the dependent lookup field on the search form if the parent lookup field is not present on the search form or the parent lookup field has no value. Set to `true` to enable the dependent lookup field on the search form. Set to `false` to disable the dependent lookup field on the search form.
showMatchedColumns	#DBF5EC	Specifies the HTML color code of the color that identifies matched columns.
showShadowColumns	true	Specifies whether to show shadow columns in the Cross-reference view. Set to `true` to show shadow columns. Set to `false` to hide the shadow columns.
subjectAreaCopyDisabled	false	Determines if users can select Copy from a subject area Actions menu to copy a subject area. Set to `true` disable the option to copy a subject area. Set to `false` to allow the option to copy a subject area.
table_default_width_key	-1	Determines the percentage minimum width of search result columns.

Property	Default value	Usage
tableMaxColumns	25	Determines the number of columns that are visible in the table view of child records and grandchild records. The default value allows for 20 visible columns and 5 hidden columns. To ensure that you have visible columns, specify an integer >5.
tabsExpandByDefault	n/a	Determines which child records are expanded by default in the data view. To expand child records by default in the data view, specify the name of each subject area separated by a comma. To expand the **XREF** tab by default, specify `xref`. To expand the **Relationships** tab by default, specify `hm_relationship`. For example, to expand the XREF tab, the Ship Address tab, and the Organization tab by default, specify `xref, ShipAddress, Organization`. If you do not set a value for tabsExpandedByDefault, no child records are expanded by default in the data view.
threadSchedulerIdleTime	5000 (seconds)	Determines the maximum thread scheduler idle time.
transactionTimeout	30 (seconds)	The number of seconds that transactions have to complete execution before timing out.
updateExistingPeriodByDefault	false	Determines if the Update existing period checkbox is enabled by default. Set to `true` to enable by default. Set to `false` to disable by default.
writeBOM	false	Exports Informatica Data Director search results as a CSV file using UTF-8 encoding with a byte order mark. If the search contains extended ASCII characters, set writeBOM to `true` to see valid data when you open the CSV file.

Related Topics:

- "Informatica Data Director Metadata Has Not Updated" on page 165

Updating the Global Properties

To update the global properties you can run the following SQL script against the CMX_SYSTEM schema.

The following SQL script, when applied to CMX_SYSTEM, initializes the global properties using their default values. Update the VALUE field in this script to modify these values.

```
insert into C_REPOS_DS_PREF_DETAIL
 (ROWID_DS_PREF_DETAIL, ROWID_DS_PREF, NAME, VALUE)
select
 'BDDGP.1', rowid_ds_pref, ' asyncChildLoading', 'true'
from C_REPOS_DS_PREF where name = '___SYSTEM_PREFERENCES_ROOT___';

insert into C_REPOS_DS_PREF_DETAIL
 (ROWID_DS_PREF_DETAIL, ROWID_DS_PREF, NAME, VALUE)
select
 'BDDGP.2', rowid_ds_pref, 'bulkexportloadsize', '1000'
from C_REPOS_DS_PREF where name = '___SYSTEM_PREFERENCES_ROOT___';

insert into C_REPOS_DS_PREF_DETAIL
 (ROWID_DS_PREF_DETAIL, ROWID_DS_PREF, NAME, VALUE)
select
 'BDDGP.3', rowid_ds_pref, 'CompositePagerTotalRecords', '5000'
from C_REPOS_DS_PREF where name = '___SYSTEM_PREFERENCES_ROOT___';
insert into C_REPOS_DS_PREF_DETAIL
 (ROWID_DS_PREF_DETAIL, ROWID_DS_PREF, NAME, VALUE)
select
 'BDDGP.4', rowid_ds_pref, 'convert2DigitYearTo4Digit', 'true'
from C_REPOS_DS_PREF where name = '___SYSTEM_PREFERENCES_ROOT___';
insert into C_REPOS_DS_PREF_DETAIL
 (ROWID_DS_PREF_DETAIL, ROWID_DS_PREF, NAME, VALUE)
select
 'BDDGP.5', rowid_ds_pref, 'credentialsAutofillDisabled', 'true'
from C_REPOS_DS_PREF where name = '___SYSTEM_PREFERENCES_ROOT___';

insert into C_REPOS_DS_PREF_DETAIL
 (ROWID_DS_PREF_DETAIL, ROWID_DS_PREF, NAME, VALUE)
select
 'BDDGP.6', rowid_ds_pref, 'CSVColumnSeparator', ','
from C_REPOS_DS_PREF where name = '___SYSTEM_PREFERENCES_ROOT___';

insert into C_REPOS_DS_PREF_DETAIL
 (ROWID_DS_PREF_DETAIL, Create_Date, creator, Last_Update_Date, Updated_By,
ROWID_DS_PREF, NAME, VALUE)
select
 'PREF_DET_4', sysdate, 'CMX', sysdate, 'admin', rowid_ds_pref,
'enableCreateBeMenuGrouping', 'true'
from C_REPOS_DS_PREF where name = '___SYSTEM_PREFERENCES_ROOT___';

insert into C_REPOS_DS_PREF_DETAIL
 (ROWID_DS_PREF_DETAIL, ROWID_DS_PREF, NAME, VALUE)
select
 'BDDGP.7', rowid_ds_pref, 'enableRememberCredentials', 'true'
from C_REPOS_DS_PREF where name = '___SYSTEM_PREFERENCES_ROOT___';

insert into C_REPOS_DS_PREF_DETAIL
 (ROWID_DS_PREF_DETAIL, ROWID_DS_PREF, NAME, VALUE)
select
 'IDDATT.0', rowid_ds_pref, 'enableTaskAttachments', 'true'
from C_REPOS_DS_PREF where name = '___SYSTEM_PREFERENCES_ROOT___';
```

```
insert into C_REPOS_DS_PREF_DETAIL
 (ROWID_DS_PREF_DETAIL, ROWID_DS_PREF, NAME, VALUE)
select
 'BDDGP.8', rowid_ds_pref, 'expandDropDownListShowFullValue', 'true'
from C_REPOS_DS_PREF where name = '___SYSTEM_PREFERENCES_ROOT___';
insert into C_REPOS_DS_PREF_DETAIL
 (ROWID_DS_PREF_DETAIL, ROWID_DS_PREF, NAME, VALUE)
select
 'BDDGP.9', rowid_ds_pref, 'exportusingmultithread', 'true'
from C_REPOS_DS_PREF where name = '___SYSTEM_PREFERENCES_ROOT___';

insert into C_REPOS_DS_PREF_DETAIL
 (ROWID_DS_PREF_DETAIL, ROWID_DS_PREF, NAME, VALUE)
select
 'BDDGP.10', rowid_ds_pref, 'handleUserExitBeforeShowingDialog', 'true'
from C_REPOS_DS_PREF where name = '___SYSTEM_PREFERENCES_ROOT___';

insert into C_REPOS_DS_PREF_DETAIL
 (ROWID_DS_PREF_DETAIL, ROWID_DS_PREF, NAME, VALUE)
select
 'BDDGP.11', rowid_ds_pref, 'HeaderBgColor', '#000000'
from C_REPOS_DS_PREF where name = '___SYSTEM_PREFERENCES_ROOT___';

insert into C_REPOS_DS_PREF_DETAIL
 (ROWID_DS_PREF_DETAIL, ROWID_DS_PREF, NAME, VALUE)
select
 'BDDGP.12', rowid_ds_pref, 'hmInactiveRelationshipsAvailable', 'false'
from C_REPOS_DS_PREF where name = '___SYSTEM_PREFERENCES_ROOT___';

insert into C_REPOS_DS_PREF_DETAIL
 (ROWID_DS_PREF_DETAIL, ROWID_DS_PREF, NAME, VALUE)
select
 'BDDGP.13', rowid_ds_pref, 'IDD2COCSConverter.prefixCoNames', 'true'
from C_REPOS_DS_PREF where name = '___SYSTEM_PREFERENCES_ROOT___';

insert into C_REPOS_DS_PREF_DETAIL
 (ROWID_DS_PREF_DETAIL, ROWID_DS_PREF, NAME, VALUE)
select
 'BDDGP.14', rowid_ds_pref, 'lookupCacheUpdatePeriod', '300000'
from C_REPOS_DS_PREF where name = '___SYSTEM_PREFERENCES_ROOT___';

insert into C_REPOS_DS_PREF_DETAIL
 (ROWID_DS_PREF_DETAIL, ROWID_DS_PREF, NAME, VALUE)
select
 'BDDGP.15', rowid_ds_pref, 'maxCopiedChildrenNumber', '10'
from C_REPOS_DS_PREF where name = '___SYSTEM_PREFERENCES_ROOT___';

insert into C_REPOS_DS_PREF_DETAIL
 (ROWID_DS_PREF_DETAIL, ROWID_DS_PREF, NAME, VALUE)
select
 'BDDGP.16', rowid_ds_pref, 'maxCopiedGrandChildrenNumber', '10'
from C_REPOS_DS_PREF where name = '___SYSTEM_PREFERENCES_ROOT___';

insert into C_REPOS_DS_PREF_DETAIL
 (ROWID_DS_PREF_DETAIL, ROWID_DS_PREF, NAME, VALUE)
select
 'BDDGP.17', rowid_ds_pref, 'maxImportThreads', '5'
from C_REPOS_DS_PREF where name = '___SYSTEM_PREFERENCES_ROOT___';

insert into C_REPOS_DS_PREF_DETAIL
 (ROWID_DS_PREF_DETAIL, ROWID_DS_PREF, NAME, VALUE)
select
 'BDDGP.18', rowid_ds_pref, 'maxParallelPromoteThreads', '1'
from C_REPOS_DS_PREF where name = '___SYSTEM_PREFERENCES_ROOT___';

insert into C_REPOS_DS_PREF_DETAIL
 (ROWID_DS_PREF_DETAIL, ROWID_DS_PREF, NAME, VALUE)
select
 'BDDGP.19', rowid_ds_pref, 'maxParallelBvtThreads', '1'
from C_REPOS_DS_PREF where name = '___SYSTEM_PREFERENCES_ROOT___';
```

```
insert into C_REPOS_DS_PREF_DETAIL
 (ROWID_DS_PREF_DETAIL, ROWID_DS_PREF, NAME, VALUE)
select
 'BDDGP.20', rowid_ds_pref, 'maxSearchResultsExportedRows', '5000'
from C_REPOS_DS_PREF where name = '___SYSTEM_PREFERENCES_ROOT___';

insert into C_REPOS_DS_PREF_DETAIL
 (ROWID_DS_PREF_DETAIL, ROWID_DS_PREF, NAME, VALUE)
select
 'BDDGP.21', rowid_ds_pref, 'maxXrefSearchReturnCount', '100'
from C_REPOS_DS_PREF where name = '___SYSTEM_PREFERENCES_ROOT___';

insert into C_REPOS_DS_PREF_DETAIL
 (ROWID_DS_PREF_DETAIL, ROWID_DS_PREF, NAME, VALUE)
select
 'BDDGP.22', rowid_ds_pref, 'openDashboardAfterTaskClose', 'false'
from C_REPOS_DS_PREF where name = '___SYSTEM_PREFERENCES_ROOT___';

insert into C_REPOS_DS_PREF_DETAIL
 (ROWID_DS_PREF_DETAIL, ROWID_DS_PREF, NAME, VALUE)
select
 'BDDGP.23', rowid_ds_pref, 'proactiveMatchResultSort', 'sortbyscorethenaction'
from C_REPOS_DS_PREF where name = '__SYSTEM_PREFERENCES_ROOT__';

insert into C_REPOS_DS_PREF_DETAIL
 (ROWID_DS_PREF_DETAIL, ROWID_DS_PREF, NAME, VALUE)
select
 'BDDGP.24', rowid_ds_pref, 'rememberCredentialsPeriod', '24'
from C_REPOS_DS_PREF where name = '___SYSTEM_PREFERENCES_ROOT___';

insert into C_REPOS_DS_PREF_DETAIL
 (ROWID_DS_PREF_DETAIL, ROWID_DS_PREF, NAME, VALUE)
select
 'BDDGP.25', rowid_ds_pref, 'samCacheUpdatePeriod', '600000'
from C_REPOS_DS_PREF where name = '___SYSTEM_PREFERENCES_ROOT___';

insert into C_REPOS_DS_PREF_DETAIL
 (ROWID_DS_PREF_DETAIL, ROWID_DS_PREF, NAME, VALUE)
select
 'BDDGP.26', rowid_ds_pref, 'search_empty_date', 'true'
from C_REPOS_DS_PREF where name = '___SYSTEM_PREFERENCES_ROOT___';

insert into C_REPOS_DS_PREF_DETAIL
 (ROWID_DS_PREF_DETAIL, ROWID_DS_PREF, NAME, VALUE)
select
 'BDDGP.27', rowid_ds_pref, 'searchForDuplicatesBeforeTaskDialog', 'true'
from C_REPOS_DS_PREF where name = '___SYSTEM_PREFERENCES_ROOT___';

insert into C_REPOS_DS_PREF_DETAIL
 (ROWID_DS_PREF_DETAIL, ROWID_DS_PREF, NAME, VALUE)
select
 'BDDGP.28', rowid_ds_pref, 'serverPageSize', '100'
from C_REPOS_DS_PREF where name = '___SYSTEM_PREFERENCES_ROOT___';

insert into C_REPOS_DS_PREF_DETAIL
 (ROWID_DS_PREF_DETAIL, ROWID_DS_PREF, NAME, VALUE)
select
 'BDDGP.29', rowid_ds_pref, 'shouldDisableSearchFieldIfDependentFieldAbsence', 'true'
from C_REPOS_DS_PREF where name = '___SYSTEM_PREFERENCES_ROOT___';

insert into C_REPOS_DS_PREF_DETAIL
 (ROWID_DS_PREF_DETAIL, ROWID_DS_PREF, NAME, VALUE)
select
 'BDDGP.30', rowid_ds_pref, 'showMatchedColumns', '#DBF58C'
from C_REPOS_DS_PREF where name = '___SYSTEM_PREFERENCES_ROOT___';

insert into C_REPOS_DS_PREF_DETAIL
 (ROWID_DS_PREF_DETAIL, ROWID_DS_PREF, NAME, VALUE)
select
 'BDDGP.31', rowid_ds_pref, 'subjectAreaCopyDisabled', 'true'
```

```
from C_REPOS_DS_PREF where name = '___SYSTEM_PREFERENCES_ROOT___';

insert into C_REPOS_DS_PREF_DETAIL
 (ROWID_DS_PREF_DETAIL, ROWID_DS_PREF, NAME, VALUE)
select
 'BDDGP.32', rowid_ds_pref, 'table_default_width_key', '20'
from C_REPOS_DS_PREF where name = '___SYSTEM_PREFERENCES_ROOT___';

insert into C_REPOS_DS_PREF_DETAIL
 (ROWID_DS_PREF_DETAIL, ROWID_DS_PREF, NAME, VALUE)
select
 'BDDGP.33', rowid_ds_pref, 'threadSchedulerIdleTime', '5000'
from C_REPOS_DS_PREF where name = '___SYSTEM_PREFERENCES_ROOT___';

insert into C_REPOS_DS_PREF_DETAIL
 (ROWID_DS_PREF_DETAIL, ROWID_DS_PREF, NAME, VALUE)
select
 'BDDGP.34', rowid_ds_pref, 'transactionTimeout', 300
from C_REPOS_DS_PREF where name = '___SYSTEM_PREFERENCES_ROOT___';

insert into C_REPOS_DS_PREF_DETAIL
 (ROWID_DS_PREF_DETAIL, ROWID_DS_PREF, NAME, VALUE)
select
 'BDDGP.35', rowid_ds_pref, 'updateExistingPeriodByDefault', 'true'
from C_REPOS_DS_PREF where name = '___SYSTEM_PREFERENCES_ROOT___';

insert into C_REPOS_DS_PREF_DETAIL
 (ROWID_DS_PREF_DETAIL, ROWID_DS_PREF, NAME, VALUE)
select
 'BDDGP.36', rowid_ds_pref, 'writeBOM', 'false'
from C_REPOS_DS_PREF where name = '___SYSTEM_PREFERENCES_ROOT___';

insert into C_REPOS_DS_PREF_DETAIL
 (ROWID_DS_PREF_DETAIL, ROWID_DS_PREF, NAME, VALUE)
select
 'BDDGP.37', rowid_ds_pref, 'isFillOnGap', 'false'
from C_REPOS_DS_PREF where name = '___SYSTEM_PREFERENCES_ROOT___';

insert into C_REPOS_DS_PREF_DETAIL
 (ROWID_DS_PREF_DETAIL, ROWID_DS_PREF, NAME, VALUE)
select
 'BDDGP.38', rowid_ds_pref, 'maxXrefSearchReturnCount', '1000'
from C_REPOS_DS_PREF where name = '___SYSTEM_PREFERENCES_ROOT___';
insert into C_REPOS_DS_PREF_DETAIL
 (ROWID_DS_PREF_DETAIL, ROWID_DS_PREF, NAME, VALUE)
select
 'BDDGP.39', rowid_ds_pref, 'deleteMovedRelInExplorerView', 'true'
from C_REPOS_DS_PREF where name = '___SYSTEM_PREFERENCES_ROOT___';

commit;
/
```

APPENDIX A

Sizing and Platform Requirements

This appendix includes the following topics:

Database Server Sizing

IDD deployments do not have direct impact on the database server sizing.

IDD transaction requirements should be considered in defining the API section of the sizing model.

Application Server Sizing

An IDD application runs on the application server and is co-located with the other Informatica MDM Hub Server components.

The application servers should be sized to allow 1 CPU core / 1 GB of memory for every 10 concurrent IDD "heavy user" sessions. The heavy user, for the purpose of the sizing model, is defined as an IDD application user producing a constant load of 5-6 IDD operations per minute.

Client and Network Sizing

Here are the minimum and the recommended configurations for the client machines accessing Informatica Data Director:

Note: The screen resolution configured for Informatica Data Director is 1280 x 1024.

Parameter	Value
CPU	Minimum: 1.6 GHz Recommended: 2 GHz
Memory	Minimum: 1 GB Recommended: 2GB
Effective network bandwidth to the Application Server	Minimum: 10 Mbps Recommended: 100 Mbps

For more information about product requirements and supported platforms, see the Product Availability Matrix on Informatica Network:
https://network.informatica.com/community/informatica-network/product-availability-matrices

Browser Configuration Requirements

You must enable the browser on the client machines to allow cookies.

Disable the pop-up blocker if you run Informatica Data Director on the Google Chrome browser.

APPENDIX B

Application Components

This appendix includes the following topic:

Application Components Reference

An IDD application is stored in the system database (CMX_SYSTEM.C_REPOS_DS_CONFIG) as a ZIP file containing component files.

This ZIP file can be exported from or imported to the IDD Configuration Manager.

File name	Usage
IDDConfig.xml	Main configuration file for the application. It must conform to the `siperian-bdd-config-6.xsd` XML schema.
BDDBundle.properties BDDBundle_XX.properties	Resource bundles with labels for objects defined in the IDD application (such as subject areas and child objects).
MetadataBundle.properties MetadataBundle_XX.properties	Resource bundles with labels for objects defined in the ORS (such as base objects, columns, and so on).
ErrorCodeBundle.properties ErrorCodeBundle_XX.properties	Resource bundles with the text for error messages generated by an IDD application.
MessageBundle.properties MessageBundle_XX.properties	Resource bundles with text displayed in the IDD application.
BDDHelp.zip BDDHelp_XX.zip	Generic IDD help files. Help that generically describes the features of an IDD application.
CustomBDDHelp.zip CustomBDDHelp_XX.zip	Custom IDD help files. Help that has been developed so that it is specific and unique to a particular IDD application. In addition to providing implementation-specific usage instructions, this help file can provide any relevant information, such as an organization's procedures and policies.
logo.gif, logo.png, logo.jpg or logo.jpeg	A replacement for the logo that the IDD application displays in the upper left of the screen. The size for the Informatica logo is 147 pixels wide by 31 pixels high. For best results, the replacement logo should have similar dimensions.

APPENDIX C

IDD Security Configuration

This appendix includes the following topic:

IDD Security Configuration Reference

The following tables show the IDD security configuration settings. You set permissions in the Hub Console by using the Security Access Manager.

Tip: The Security Access Manager includes the following Resource Groups: ALL_GLOBAL_RESOURCES, ALL_XREF, and ALL_XREF_HISTORY. Use these groups when you want to assign the same permission to all the specified resources. For example, you can set the DELETE permission on all cross-references by selecting the DELETE check box in the ALL_XREF row.

Table 1. General

Use-case	Resource Group	Name	Sub-name	Special Requirements/ Comments	C	R	U	D	E	M
Toolbar New Subject Area	CUSTOM_RESOURCE	BDD_NAME	SUBJECT_AREA	Primary BO and all Logical one-to-ones	-	Y	-	-	-	-
	BASE_OBJECT	NAME	-		Y	Y	-	-	-	-
	CLEANSE_FUNCTION	LIB_NAME	FUNCTION_NAME		-	-	-	-	Y	-

Table 2. Data View

Use-case	Resource Group	Name	Sub-name	Special Requirements/ Comments	C	R	U	D	E	M
Create Subject Area	CUSTOM_RESOURCE	BDD_NAME	SUBJECT_AREA	Primary BO and all Logical one-to-ones	Y	-	-	-	-	-
	BASE_OBJECT	NAME	-		-	Y	-	-	-	-
	CLEANSE_FUNCTION	LIB_NAME	FUNCTION_NAME		-	Y	-	-	-	-

Use-case	Resource Group	Name	Sub-name	Special Requirements/ Comments	C	R	U	D	E	M
Read Subject Area	CUSTOM_RESOURCE	BDD_NAME	SUBJECT_AREA	Primary BO and all Logical one-to-ones	-	Y	-	-	-	-
	BASE_OBJECT	NAME	-		-	Y	-	-	-	-
Update Subject Area	CUSTOM_RESOURCE	BDD_NAME	SUBJECT_AREA	Primary BO and all Logical one-to-ones	-	Y	Y	-	-	-
	BASE_OBJECT	NAME	-		-	Y	Y	-	-	-
	CLEANSE_FUNCTION	LIB_NAME	FUNCTION_NAME		-	-	-	-	Y	-
Delete Subject Area	CUSTOM_RESOURCE	BDD_NAME	SUBJECT_AREA	Primary BO, State management is enabled	-	Y	-	-	-	-
	BASE_OBJECT	NAME	-		-	-	-	Y	-	-
Copy Subject Area	CUSTOM_RESOURCE	BDD_NAME	SUBJECT_AREA	Primary BO and all Logical one-to-ones	-	Y	-	-	-	-
	BASE_OBJECT	NAME	-		Y	Y	-	-	-	-
	CLEANSE_FUNCTION	LIB_NAME	FUNCTION_NAME		-	-	-	-	Y	-
Show BO's System columns	CUSTOM_RESOURCE	BDD_NAME	SUBJECT_AREA	BO is not new.	-	Y	-	-	-	-
	BASE_OBJECT	NAME	-		-	Y	-	-	-	-
Create Child Object	BASE_OBJECT	NAME	-	For one-to-many children just BO itself, for many-to-many children both BO and its relation BO is checked.	Y	Y	-	-	-	-
	CLEANSE_FUNCTION	LIB_NAME	FUNCTION_NAME		-	-	-	-	Y	-
Read Child Object	BASE_OBJECT	NAME	-	-	-	Y	-	-	-	-
Update Child Object	BASE_OBJECT	NAME	-	For one-to-many children just BO itself, for many-to-many children both BO and its relation BO is checked.	-	-	Y	-	-	-
	CLEANSE_FUNCTION	LIB_NAME	FUNCTION_NAME		-	-	-	-	Y	-

Use-case	Resource Group	Name	Sub-name	Special Requirements/ Comments	C	R	U	D	E	M
Delete Child Object	BASE_OBJECT	NAME	-	State management enabled. For one-to-many children just BO itself, for many-to-many children both BO and its relation BO is checked.	-	-	-	Y	-	-
	BASE_OBJECT	NAME	XREF	Cross-references for the child object must be selected.	-	-	-	Y	-	-
	CUSTOM_RESOURCE	BDD_NAME	SUBJECT_AREA	Required setting when you use the Entity View.	-	Y	-	-	Y	-

Table 3. CM

Use-case	Resource Group	Name	Sub-name	Special Requirements/ Comments	C	R	U	D	E	M
View Xref	CUSTOM_RESOURCE	BDD_NAME	SUBJECT_AREA	BO is not new.	-	Y	-	-	-	-
	BASE_OBJECT	NAME	-		-	Y	-	-	-	-
	BASE_OBJECT	NAME	XREF	Primary BO and all Logical one-to-ones. For one-to-many children just child BO. For many-to-many children child BO and relation BO.	-	Y	-	-	-	-
Find duplicates	CUSTOM_RESOURCE	BDD_NAME	SUBJECT_AREA	-	-	Y	-	-	-	-
	BASE_OBJECT	NAME	-		-	Y	-	-	-	-
Merge	CUSTOM_RESOURCE	BDD_NAME	SUBJECT_AREA	-	-	Y	-	-	-	Y
	BASE_OBJECT	NAME	-		-	-	-	-	-	Y
Unmerge	CUSTOM_RESOURCE	BDD_NAME	SUBJECT_AREA	-	-	Y	-	-	-	-

Use-case	Resource Group	Name	Sub-name	Special Requirements/ Comments	C	R	U	D	E	M
	BASE_OBJECT	NAME	-		-	-	-	-	-	Y
View Raw Data	BASE_OBJECT	NAME	RAW	-	-	Y	-	-	-	-

Table 4. Tasks

Use-case	Resource Group	Name	Sub-name	Special Requirements/ Comments	C	R	U	D	E	M
Send for approval (New Primary Object)	CUSTOM_RESOURCE	BDD_NAME	SUBJECT_AREA	Primary BO and all Logical one-to-ones, State management is enabled	-	Y	-	-	-	-
	BASE_OBJECT	NAME	-		Y	Y	-	-	-	-
	BASE_OBJECT	NAME	-	Many-to-many children, State management is enabled	Y	Y	-	-	-	-
	CLEANSE_FUNCTION	LIB_NAME	FUNCTION_NAME	Primary Object and all logical one-to-ones	-	-	-	-	Y	-
	CUSTOM_RESOURCE	BDD_NAME	TASK_TYPE:SA	Default for approval	Y	-	-	-	-	-
Send for approval (Existing Primary Object)	CUSTOM_RESOURCE	BDD_NAME	SUBJECT_AREA	Primary BO and all Logical one-to-ones, State management is enabled	-	Y	-	-	-	-
	BASE_OBJECT	NAME	-		-	Y	Y	-	-	-
	BASE_OBJECT	NAME	-	Many-to-many children, State management is enabled	-	Y	Y	-	-	-
	CLEANSE_FUNCTION	LIB_NAME	FUNCTION_NAME	Primary Object and all logical one-to-ones	-	-	-	-	Y	-
	CUSTOM_RESOURCE	BDD_NAME	TASK_TYPE:SA	Default for approval	Y	-	-	-	-	-
Send for Approval Task	CUSTOM_RESOURCE	BDD_NAME	SUBJECT_AREA	Send for Approval and Edit buttons are enabled for a new created record. Save	-	Y	-	-	-	-
	BASE_OBJECT	NAME	-		Y	Y	Y	-	-	-

Use-case	Resource Group	Name	Sub-name	Special Requirements/ Comments	C	R	U	D	E	M
	CUSTOM_RESOURCE	BDD_NAME	TASK_TYPE:SA/ ReviewNoApprove	button is disabled.	Y	-	-	-	-	-
Open task from Start workspace	CUSTOM_RESOURCE	BDD_NAME	SUBJECT_AREA	-	-	Y	-	-	-	-
	BASE_OBJECT	NAME	-		-	Y	-	-	-	-
	CUSTOM_RESOURCE	BDD_NAME	TASK_TYPE:SA		-	-	-	-	Y	-
Create Task	CUSTOM_RESOURCE	BDD_NAME	SUBJECT_AREA	Primary BO and all Logical one-to-ones, State management is enabled	-	Y	-	-	-	-
	BASE_OBJECT	NAME	-		-	Y	-	-	-	-
	BASE_OBJECT	NAME	-	Many-to-many children, State management is enabled	-	-	-	-	-	-
	CLEANSE_FUNCTION	LIB_NAME	FUNCTION_NAME	Primary Object and all logical one-to-ones	-	-	-	-	Y	-
	CUSTOM_RESOURCE	BDD_NAME	TASK_TYPE:SA	Any creational task type	Y	-	-	-	-	-
View Task Details	CUSTOM_RESOURCE	BDD_NAME	SUBJECT_AREA	-	-	Y	-	-	-	-
	CUSTOM_RESOURCE	BDD_NAME	TASK_TYPE:SA	-	-	-	-	-	Y	-
Merge Task	CUSTOM_RESOURCE	BDD_NAME	SUBJECT_AREA	-	-	Y	-	-	-	-
	CUSTOM_RESOURCE	BDD_NAME	TASK_TYPE:SA/ Merge		Y	-	-	-	-	-
Unmerge Task	CUSTOM_RESOURCE	BDD_NAME	SUBJECT_AREA	-	-	Y	-	-	-	-
	CUSTOM_RESOURCE	BDD_NAME	TASK_TYPE:SA/ Unmerge		Y	-	-	-	-	-
Queue for merge	CUSTOM_RESOURCE	BDD_NAME	SUBJECT_AREA	Queue for Merge button is enabled.	-	Y	Y	-	-	Y
	BASE_OBJECT	NAME	-		-	Y	-	-	-	Y
	CUSTOM_RESOURCE	BDD_NAME	TASK_TYPE:SA/ Merge		-	-	-	-	Y	-

Use-case	Resource Group	Name	Sub-name	Special Requirements/ Comments	C	R	U	D	E	M
Execute Task's action	CUSTOM_RESOURCE	BDD_NAME	SUBJECT_AREA	-	-	Y	-	-	-	-
	CUSTOM_RESOURCE	BDD_NAME	TASK_TYPE:SA		-	-	-	-	Y	-

Table 5. History View

Use-case	Resource Group	Name	Sub-name	Special Requirements/ Comments	C	R	U	D	E	M
View Subject Area History	CUSTOM_RESOURCE	BDD_NAME	SUBJECT_AREA	Primary BO is persisted, history is enabled for primary BO.	-	Y	-	-	-	-
History View for Primary Object	CUSTOM_RESOURCE	BDD_NAME	SUBJECT_AREA	Primary BO and all Logical one-to-ones.	-	Y	-	-	-	-
	BASE_OBJECT	NAME	HISTORY		-	Y	-	-	-	-
	BASE_OBJECT	NAME	-	History should be enabled for BO.	-	Y	-	-	-	-
History View for Child BO	CUSTOM_RESOURCE	BDD_NAME	SUBJECT_AREA	For many-to-many children relationship privileges are taken in account.	-	Y	-	-	-	-
	BASE_OBJECT	NAME	HISTORY		-	Y	-	-	-	-
	BASE_OBJECT	NAME	-	History should be enabled for BO.	-	Y	-	-	-	-
View BO Xref History	CUSTOM_RESOURCE	BDD_NAME	SUBJECT_AREA	History should be enabled for BO.	-	Y	-	-	-	-
	BASE_OBJECT	NAME	XREF_HISTORY		-	Y	-	-	-	-
	BASE_OBJECT	NAME	-		-	Y	-	-	-	-
View BO Merge History	CUSTOM_RESOURCE	BDD_NAME	SUBJECT_AREA	-	-	Y	-	-	-	-
	BASE_OBJECT	NAME	-		-	Y	-	-	-	-

Table 6. Charts

Use-case	Resource Group	Name	Sub-name	Special Requirements/ Comments	C	R	U	D	E	M
View Chart	CUSTOM_RESOURCE	BDD_NAME	CHART/ View	-	-	Y	-	-	-	-

APPENDIX D

Data Security

This appendix includes the following topics:

Data Security Overview

Data security is the protection of data from either accidental or unauthorized access, modification, corruption, destruction, duplication or disclosure during operations such as input, processing, storage, transmission, and output and that access to data is suitably controlled.

IDD data security ensures that data is accessible to users based on the following criteria:

- User role
- Data security configuration
- Data stored in the hub

Data Security Using Filters

Data security in Informatica Data Director is configured using the **Subject Area** dialog in the Informatica Data Director Configuration Manager. You can define filters on the subject area column to restrict and secure subject area data that individual users can access. Filters can be defined on primary object column, child column and grandchild column. You can set up any number of filters for a subject area and subject area group column.

Informatica Data Director data security supports the following types of values for security filters in table column type in the database:

- String
- Integer
- Float

Note: Table column value of type Date is not supported by data security filters of Informatica Data Director.

Consider the following rules and guidelines when you work with filters:

- Each security filter is defined on columns in the subject area and consists of a filter value to be applied to a list of roles.
- Security filters are based on exact values and not on wild card comparisons or ranges.

- Filters must be defined on the match columns to apply security filters consistently across basic, extended, and advanced searches.
- Filters can be combined. A user with multiple roles can have combinations of filters applied. The result is that a user has access to all the data available in each assigned role through a union of the filter assignments.
- Filters on different columns can be combined to create multi-dimensional data security.
- Multiple filters on a single column for a single role. A user has access to a union of all data that meet each filter.
- Filters on multiple columns for a single role. A user has access to intersection of all data that meet each filter.
- In IBM Db2 environments, filters on columns with a datatype of float do not filter beyond the scale of the column. For example, if the scale of the column is 1 and you set the filter to 1.2, values that are beyond the scale of the column, such as 1.21, are also accessible.

For more information, see the *Configuration Manager* online help.

Data Security Parameters

To restrict the data that users that belong to a particular role can access, you can configure the data security parameters in the `BDDConfig.xml` file.

You can configure the following data security parameters:

securityFilter

Specifies the column that Informatica Data Director (IDD) bases filtering on. The 'columnUid' attribute specifies the column ID or match path.

securityValue

Specifies the value that the securityFilter column must have to allow the user to view the data in a record.

securityRole

Specifies the role to which the security filter applies. The 'roleID' attribute specifies the role ID of the role whose access is restricted by the data security filter.

Data Security Parent Object Configuration Example

You need to configure security in the `BDDConfig.xml` file so data stewards see content that applies to their country. Data stewards in France see parent records with a country value of 'FR', and data stewards in Japan see parent records with a country value of 'JA'.

To filter based on the location of a data steward, create a role in the MDM Hub for each region. In this example, you assign the data stewards in France the 'DSFrance' role, and you assign the data steward in Japan the 'DSJapan' role.

The following excerpt from the `BDDConfig.xml` file shows how to configure data security for this example:

```
<dataSecurity>
  <securityFilter columnUid="COUNTRY">
    <securityValue value="FR">
      <securityRole roleUid="DSFrance"/>
    </securityValue>
    <securityValue value="JA">
      <securityRole roleUid="DSJapan"/>
    </securityValue>
```

```
  </securityFilter>
</dataSecurity>
```

Data Security Grandchild Object Configuration Example

You want the data stewards in France to view child records and grandchild records when the 'Country' column of the C_MT_ADDRESS grandchild record has a value of 'FR'.

To filter based on the location of a data steward, create a role in the MDM Hub for data stewards in France. In this example, you assign the data stewards in France the 'DSFrance' role. Use the match path component for the grandchild object when you specify the 'columnUid' value.

The following excerpt from the `BDDConfig.xml` file shows how to configure data security for this example:

```
<subjectArea name=”Organization”>
  <one2ManyChild name="Employee">
    <dataSecurity>
      <securityFilter columnUid="MATCH_PATH_COMPONENT.C_MT_ADDRESS|COUNTRY">
        <securityValue value="RUS">
          <securityRole roleUid="DSFrance"/>
        </securityValue>
      </securityFilter>
    </dataSecurity>
    <one2ManyChild name="Address" mpcUid="C_MT_ADDRESS">
    </one2ManyChild>
  </one2ManyChild>
</subjectArea>
```

By default, users cannot view the child record if you configure a filter for a grandchild column, but the child record does not have grandchild records. To allow users to view child records with no grandchild records, set the 'allowDsEmptyChildren' global property to `true`.

Apply Data Security

Data security provides the solution to protect organizational data such as transactional, historical, dynamic, hierarchical, and static data that organizations obtain, store, create, delete and update in order to conduct business processes.

In an IDD application, data security defined on a subject area is applied to the following content types:

- Search data
- Entity data
- Hierarchical data
- Historical data
- Task data.

Data Security in Search Data

IDD search allows a user to search records by subject area and subject area group. If a subject area has any data security filters for the user, then search results should contain only those records that meet the data security. Data security is applied for both basic and fuzzy searches. For example, when a user performs a search and has access only to people in CA, the search result displays only records of people in CA.

Note:

- If a user with data security performs a search using a search term, the search result is an intersection of the records that meet data security and what is returned by the search.
- If the search de-duplication for a child record is not enabled and the user with data security on the primary objects is more than one child record, then the search result will have all the records related to the primary object.
- When search is performed over subject area group, different data security filters are used.
- IDD collapses all duplicates in case of amount of found records is less than configured server page size, for example, all results are fetched after first request.

Data Security in Entity Data

IDD allows a user to access Primary Object (PO) record, child record, grandchild record and subject area links by subject area and subject area group. If a subject area has data security filters for a user, then the user can access only those records that meet the data security. The following sections describe how data security is applied for different operations in the data view.

Open a Record

Data security filters ensure that only authorized users can open records in the data view.

Open a Record Using a Single Role

Users with a single role can open the primary object records if the following conditions are satisfied:

- Primary object must satisfy all the data security filters that exist on the primary object's column.
- Primary object must have at least one record passing the security restrictions enabled on each child tab with data security.

For example, consider a data security model in which a user has the role, Sales Manager- NY and has the following security filters configured:

- Filter 1: State code is NY.
- Filter 2: Phone type is Business and Home.
- Filter 3: Person salutation code is MR.

Using this data security model, consider a scenario where the database has a primary object record, Mr. Steve Nash, who has the billing address in NY state and Business as phone type. User with Sales Manager-NY role can open the Mr. Steve Nash record on the data view as the primary object satisfies filter 3 and its children satisfies filter 1 and filter 2.

Using the same data security model, consider another scenario, where the database has a primary object record, Mr. Carlos Booser, who has the Bill Address in NY state and Mobile as phone type. User with Sales Manager- NY role cannot open the Mr. Carlos Booser record on the data view as it does not pass the restriction enabled on the child tab of Phone type.

Filter Record Using a Single Role

Users with a single role can access the child or grandchild object details only if it satisfies all the data security filters that exist on the primary object's child or grandchild column.

Consider, for example, a data security model in which the user has the role, SalesManager- NY and has the following security filters configured:

- Filter 1: State code is NY.
- Filter 2: Phone type is Business and Home.
- Filter 3: Person salutation code is MR.

Using the data security model mentioned above, consider a scenario where the database has a primary object record: Mr. Robin Cameron, who has the Billing address in CA, TX and NY State and Business and facsimile as phone type. User with Sales Manager- NY role can see only address in NY State on billing address tab and only Business phone on Telephones tab, all other records on both tabs are filtered out.

Filter Records Using Multiple Roles

By default, a user that belongs to multiple roles can access child records or grandchild records based on the combined data security filters.

For example, consider a data security model in which the user belongs to the role 'Sales Manager NY' and the role 'Car Sales Manager NJ'.

The role 'Sales Manager NY' has the following data security filters:

- Filter 1: State Code is 'NY'.
- Filter 2: Phone Type is 'Business' or 'Home'.

The role 'Car Sales Manager NJ' has the following data security filters:

- Filter 1: State Code is 'NJ'.
- Filter 2: Car Year is '2009'.

Consider a scenario where the database has a primary object record for John Smith. John has billing addresses with state code values of 'NY', 'NJ', and 'TX'. John has phone numbers with phone type values of 'Business' and 'Facsimile'. John has a car produced in the year 2009, and a car produced in the year 2001. The user with the role 'Sales Manager NY' and the role 'Car Sales Manager NJ' sees the following information:

- The user sees the NY and NJ billing addresses because the State Code filter is configured for both roles.
- If the attribute 'affectFilter' for 'securityValue' is `false`, the user sees phone numbers for all phone types, and car records for all years. Informatica Data Director (IDD) does not apply the data security filters for the phone type or the car year because these filters are not configured for both roles.
- If the attribute 'affectFilter' for 'securityValue' is `true`, the user sees the phone numbers for the 'Business' phone type and car data for the year 2009. IDD applies all data security filters that are configured for each role. The default of the attribute 'affectFilter' is `true`.

Data Security Filters for Inherited Roles

You can configure data security filters for inherited roles that descend from a parent role. To configure the data security filters for inherited roles, set the affectFilter attribute for the securityFilter parameter in the `BDDConfig.xml` file.

For example, consider a role hierarchy with a DataSteward_NY role that is a descendant of a DataSteward role. A user that belongs to the DataSteward_NY role also belongs to the DataSteward role.

You want to configure a data security filter that only affects the users who belong to the DataSteward_NY role. You want the users who belong to the DataSteward_NY role to see records that have a STATE_CD value of `NY`. You must set the affectFilter attribute to `false` to filter data for the DataSteward_NY role. When the affectFilter attribute is `false`, Informatica Data Director filters data for the DataSteward_NY role independently of the data security filters for the DataSteward role.

The following excerpt from the `BDDConfig.xml` file shows how to configure data security filters for this example:

```
<securityFilter columnUid="MATCH_PATH_COMPONENT.C_MT_ADDRESS|STATE_CD">
  <securityValue value="NY">
    <securityRole roleUid="DataSteward_NY"/>
  </securityValue>
  <securityValue affectFilter="false">
    <securityRole roleUid="DataSteward"/>
  </securityValue>
</securityFilter>
```

View Relationships

In IDD, a relationship describes the affiliation between two specific entities. For example, a customer entity can be logically linked to an address entity.

Relationship tab in Data View contains information about Hierarchy Manager relations of the primary object with some other Hierarchy Manager entities. Some of the hierarchy manager entities can be transformed to primary objects that might be affected by data security.

Relationship tab should contain only those relationships that connect Hierarchy Manager entities associated with the primary objects satisfying the data security settings.

Merge Data

Merging is the process of combining two or more records because they are identical or sufficiently similar to be considered duplicates. You merge records to consolidate duplicate data into a single entity (master entity) that represents the best version of the truth (BVT). Where attribute values differ, the retained values could be determined by different factors. For example, retained values could be determined based on the trust configuration for these records, or based on values supplied by a user who chose to edit the override value instead.

In the IDD application, the **Find Merge Candidates Search** dialog should display only those records that are valid by primary object's subject area data security.

Export Data and Export Profiles

All data security filters and data masking are applicable for exported data as well as for data displayed to the user.

Save a Record

A user can save a record only after validation is completed and all the data security filters of the subject area are applied. If a record does not meet the requirements of data security filters, a warning message is displayed to the user.

In the warning message dialog box, if you choose **Yes**, primary object is saved and the tab is closed. If you choose **No**, primary object is not saved, yet user can continue filling the primary object details.

Find Duplicates (Potential Matches)

Duplicates are entities in which the data in certain columns such as name, address, or organization data is identical or sufficiently similar to be considered nearly identical. IDD uses special matching logic and match-enabled attributes to determine whether two entities are sufficiently similar to be considered matches. Duplicates are entities that you consider for a merge.

To search for potential matches, click **More Actions** and choose **Find Duplicates**. If a subject area has any data security filters for the user, then the results for Find Duplicates should contain only those PO records that meet the data security.

For example, consider a data security model in which the user has a single role, SalesManager- CA and the user runs a search for duplicates for a person. The search result will have individuals having at least one billing address in CA State and all the other duplicates are filtered.

Note: If a user has more than one role and runs a search for duplicates, the user is allowed to see a union of results that each role is able to see.

Data Security in Hierarchical Data

In IDD, a hierarchy is a set of relationship types. They are merely relationship types that are grouped together for the ease of classification and identification. When Hierarchy Manager View is opened, it first checks if the hierarchy manager anchor entity can be transformed to primary object and if transformed to primary object is visible by data security.

Add Hierarchy Manager Entities

Hierarchy manager entity can be added to canvas using Search and Create operations.

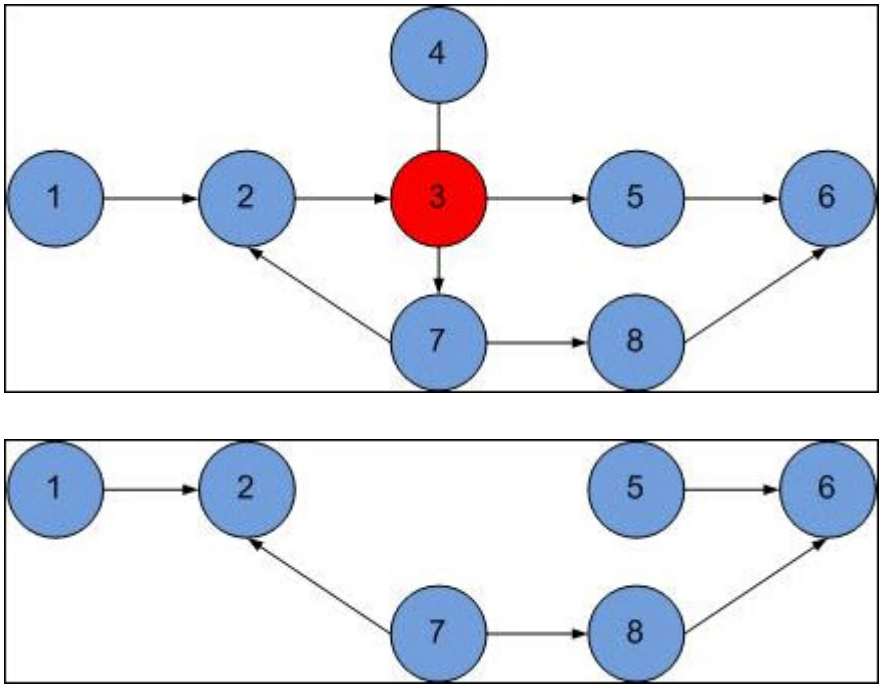
In the data search results, only records that are allowed by the primary object of the subject area data security are shown. Hence, using the search option only valid objects can be added.

When a user creates HM entity, user can save PO in the data view that is not valid by Data Security. If user confirms to save PO that is not visible by Data Security, this HM entity is not added to canvas.

Represent Hierarchy Manager Graphs

Hierarchy manager entities can be transformed to primary objects. Primary objects that are not visible as a result of data security are not represented in hierarchy manager graph as hierarchy manager entities. If a hierarchy manager entity is not visible to a user, then hierarchy manager graph must not display this entity and its sub-tree.

For example, consider the following hierarchy manager graph, where a user is not allowed to view entity 3 because of the data security. In this case, the graph must be visible to the user without entity 3 and its sub-tree component, entity 4.



Note: Users with multiple roles can access the union of all objects that can be accessed by each of the multiple roles.

Data Security in Historical Data

IDD allows you to view the history of data processing events like updates, deletes, and merges that occurred to the selected entity. If a base object meets the data security requirements, then the history of this base object is shown, even if in the past it was hidden due to data security.

Data Security in History impacts the following areas:

- History for primary objects
- History events
- Composite History of primary objects.

Open History Details

A user can open the history of data using deep link or as a history component in IDC. In this case, data security is applied by IDD to ensure that the primary object for which the history is built is visible to authorized users.

View History Events

Events in the History view refers to subject area objects, primary object or primary object children. If primary object is not visible due to data security, then History view is not shown at all. If a child record is not visible due to data security, History view is shown for the primary object but history events of the child record are not added to the timeline.

Data Security in Deep Links

Deep linking feature in IDD application enables you to manage application state using URL parameters. It allows defining of some internal navigation path in URL, which can be opened in the IDD application.

This feature is also used to:

- Provide navigation between IDC component and IDD application.
- Provide bookmarks to specific parts of application.

IDD data security impacts the following deep link areas:

- **Opening a record**: Primary object and its children data are checked against the data security settings, before a new tab displays the record's data in the data view.
- **Opening a Task**: All the data security settings mentioned in the section "Data Security in Task Data" on page 146 is applicable. These data security settings determine whether a user is able to open a task or not.

APPENDIX E

Example Role-Based Security Configuration

This appendix includes the following topics:

Example Role-Based Security Configuration Overview

This appendix describes a simple scenario for setting up role-based access to secure resources in Informatica Data Director (IDD).

It introduces key concepts and walks through the security configuration tasks required to implement a sample scenario. The purpose of this appendix is to provide IDD implementers with some basic knowledge about what might be required to set up security in their IDD implementation projects.

Note: This is not a lab tutorial for building a working sample application. It is simply a narrative walkthrough of the tools and tasks involved to support a given scenario.

Key Concepts

This section describes key concepts that you need to understand before implementing security for IDD.

IDD, Security Access Manager (SAM), and Services Integration Framework (SIF)

Most IDD functionality is implemented using SIF calls.

SIF requires SAM configuration, which is highly granular, to provide the rights and privileges needed to run the SIF calls. SAM configuration involves defining the users, roles, secure resources, and privileges required to support role-based access to data and operations.

Tools for Setting Up IDD Security

You use the following tools in the Informatica MDM Hub Console to configure SAM: Users, Users and Groups, Roles, and Secure Resources / Resource Groups (including packages and cleanse functions).

In addition, you use the IDD Configuration Manager to bind your SAM configuration to IDD objects.

Related Reading

The following Informatica documentation provides important reference information about the Security Access Manager, Services Integration Framework, and Data Director security:

- *Multidomain MDM Security Guide*
- *Multidomain MDM Services Integration Framework Guide*, "Using the Security Access Manager with SIF API"

Related Topics:

- "IDD Security Configuration Reference" on page 105

Object and Task Security

It is useful to think of IDD security in two broad categories:

- Object security: Access to subject area data, and the ability to perform operations on that data (such as view, create, update, and merge) in IDD.
- Task security (workflow). Access to tasks and actions based on roles that are defined in the workflow.

Note: Although this example scenario focuses on object security only, many of the ideas are also applicable to task security in IDD, as task security depends on SAM as well.

Related Topics:

- "Workflows and Tasks" on page 20

Tips for Designing Security for IDD Usage

Implementing IDD security is an iterative and ongoing process.

To get started, you need to understand the various types of access to resources (objects and operations) that IDD users will need in your IDD application.

In SAM, the *role* is the core mechanism that determines how much access a user has to IDD resources. SAM is highly configurable and provides granular control over resources. Consider creating a separate role for each unique combination of objects/operations access, and assign privileges to that role. Roles can be based on other roles to create layers of expanding privileges. Once configured, you assign users to the role best suited to their job responsibilities.

This example scenario follows the principle of *least privilege* - access to resources are granted on an as-needed basis. By default, users have no permissions. You then selectively grant users only those permissions that are required to complete the operations for which they are responsible.

Important: SAM configuration must match the IDD configuration. Whatever you configure in the IDD application, you need to configure SAM to provide sufficient privileges to support the configured IDD functionality.

Other Considerations

When planning security for your IDD application, consider the following issues:

- For IDD to access Informatica MDM Hub resources, the resources must be configured as SECURE (not private) in the Secure Resources tool in the Hub Console.
- SAM is configured on a per-ORS basis. When you add IDD users, you need to set the IDD schema as the default database for these users.
- In the Data workspace, IDD users typically will not see explicit error messages for insufficient privileges. For example, a certain resource might simply be hidden from the user because they are not configured to access it. When testing your security configuration, refer to the server log for debugging information.
- In an entity workspace, IDD displays all resources regardless of user role. When users perform actions for which they do not have the required security permission, IDD displays error messages.
- The security configuration is stored in two places: in the Hub Server cache and the IDD cache. There is a slight lag time (1 minute) to synchronize changes. In a development environment, you can restart the server to refresh the cache.

IDD Security Configuration Tasks

This section walks through the series of tasks to implement a sample, role-based scenario: provide IDD users with four different privilege levels (no permissions, read-only, create, and update) to access a Party base object and affiliated resources.

For example, consider a scenario with two subject areas such as Party and Organization and the Party subject area has as a logical one-to-one relationship with the Organization subject area. In the data view, to edit any attributes of the record, you must have the CREATE and UPDATE privilege on both the subject areas that is C_PARTY and C_ORGANIZATION. If some fields in the primary object or the object with a logical one-to-one relationship with the primary object are READ-ONLY, you can still edit the primary object. The fields that are READ-ONLY are visible in Data view, but cannot be edited. If all the fields in the primary object and the object with a logical one-to-one relationship with the primary object have READ-ONLY permissions, you cannot edit the primary object in Data view.

Configure Design Objects in the Hub Console

Before you begin, you must configure all design objects in the Hub Console that will be used by the IDD application.

In this scenario, the following objects are required:

- Party base object (Schema Manager)
- Packages (Queries tool and Packages tool), which affect Search
- Match rules (Schema Manager), which affect Find Duplicates (possible matches)
- Cleanse functions (Cleanse Functions tool), which affect data entry (inline cleansing of data upon save)

For more information about the objects and Hub Console tools, see the *Multidomain MDM Configuration Guide*.

Note: Although this scenario describes configuring just a single base object, customer data models involve a web of various relationships among base objects. What is important is that you configure the entire constellation of base objects and other design objects that will be accessed by users of the IDD application.

Configure IDD Application Users (Users Tool)

Begin configuring SAM by adding IDD user accounts to the Master Database of your Informatica MDM Hub implementation.

For example, in the Hub Console, you could run the Users tool and add the following user accounts:

User Account	Assigned a Role that Grants
user_1	No permissions (default).
user_2	Read-only permission to the Party base object.
user_3	Create permission to the Party base object.
user_4	Update permission to the Party base object.

Note: For each user, make sure that they have access to all Operation Record Stores (ORS) associated with the IDD application. You can also do this on the Users Assigned to Database tab in the Users and Groups tool.

Configure Secure Resources (Secure Resources Tool)

For IDD to have access to a resource, it must be flagged as SECURE in the Secure Resources tool.

You must make sure that all of the design objects configured earlier are configured as SECURE resources.

- Party base object, including associated:
  - match rule sets, which are used in IDD for Search for Duplicates (potential matches)
  - content metadata (HISTORY, RAW, and XREF), which are used in IDD to display the change history, cross references, and raw records
- cleanse functions used for data entry
- packages used for Search results

Note:

- Consider creating resource groups to organize IDD-accessible resources and to expedite security configuration.
- If you want to prevent all IDD users from having access to a certain resource, make it PRIVATE. For example, you could globally hide IDD access to RAW records in this way.

Create and Configure a New IDD Application (IDD Configuration Manager)

In the IDD Configuration Manager, create a new IDD application, and then configure it. Add a subject area group (such as Party Group), and then the Party subject area.

In this scenario, you would specify all Party columns, the Party match rule set to use for duplicate checks (must be configured as SECURE), and a cleanse function (must be configured as SECURE). When you are finished, save changes and deploy the IDD application.

Note: One way to restrict user access to information is to specify only a subset of columns to display in the IDD GUI. Later, you can configure permissions for roles at the column level, allowing some users to see a column while others cannot.

View Custom Resources (Secure Resources Tool)

When you first deploy an IDD application in the IDD Configuration Manager, it automatically adds a node for the application under the Custom Resources node.

When you redeploy the application, the IDD Configuration Manager adds/updates any special support design objects as SECURE resources. These support objects are necessary for IDD integration with SAM. After you save changes to the subject area and redeploy the application, revisit the Secure Resources tool and note the custom resources that were automatically added by the IDD configuration file.

Note: Remember that there can be a slight delay between the time you save your application configuration and the time in which it shows up in the Secure Resources tool.

Here is a brief description of these resources:

Custom Resource	Exposes the Ability To
REPORT/View	View reports on the Start workspace.
SEARCH_QUERY/Create	Create private queries.
SEARCH_QUERY/CreatePublic	Create public queries.
SUBJECT_AREA/ *BaseObject*	Access the Subject Area in IDD. You might see multiple SUBJECT_AREA resources getting their data from the same base object, but representing different views. Even if the role has access to the base object, you can additionally restrict privileges on these resources to limit which SUBJECT_AREA the role can search, view, and so on.
TASK_TYPE/ *SubjectArea*:*TaskType*	Access the specified task for the associated subject area.

Configure Roles and Resource Privileges (Roles Tool)

Roles provide highly-granular control over what privileges are assigned to which resources.

To expedite your security configuration, you can even create a hierarchy of roles by assigning roles to other roles. In the Hub Console, use the Roles tool to configure the permissions needed for the IDD operations performed by this role.

Create Roles

You begin by creating the roles you want, such as:

Role Name	Description
party_no_privileges_role	Initial default. No permissions to access anything (comparable to a user with no assigned role). This is not a real-world scenario - it is provided to show what happens as privileges are added with other roles.
party_read_only_role	Read-only permission to the Party base object.
party_create_role	Create permission to the Party base object.
party_update_role	Update permission to the Party base object.

Configure Resource Privileges for Base Objects and Affiliated Objects

Next, for each role, you configure the resource privileges for base objects and affiliated objects.

To configure base object permissions in the Roles tool, select the role you want to configure, expand the Base Objects node, expand the Party node, and configure privileges for the base object, content metadata, and match rule sets.

The following table shows the privileges that you should configure for this scenario.

Role Name	Resource Privileges
party_no_privileges_role	No permissions.
party_read_only_role	- READ privileges to all columns in the PARTY base object - READ privileges to an applicable match rule set - READ privileges to content metadata (HISTORY, RAW, and XREF).
party_create_role	- READ privileges to all columns in the PARTY base object. - READ privileges to an applicable match rule set - READ privileges to content metadata (HISTORY, RAW, and XREF) - CREATE privileges to all columns in the PARTY base object (required for creating a new record) - UPDATE privileges to all columns in the PARTY base object (if you want to allow this role to update an existing record as well)
party_update_role	- READ privileges to all columns in the PARTY base object. - READ privileges to an applicable match rule set - READ privileges to content metadata (HISTORY, RAW, and XREF) - UPDATE privileges to all columns in the PARTY base object (required for saving changes to a record)

Tips:

- If your base object has relationships with other base objects (for example, parent-child relationships, foreign key lookups, or one-to-one relationships), you need to configure access to all of these resources as well. Lookups require READ access, while related base objects require permissions that are comparable to the core base object).
- You can selectively disable READ privileges on certain columns so that users cannot see them in the IDD application. Similarly, you can enable READ and disable UPDATE privileges so that users can see the columns but not change any data.
- You must configure READ access to a match rule set in order for Find Duplicates to work.
- You can control whether a role can view history (requires READ privileges to HISTORY), view cross references (requires READ privileges to XREF), and view raw records (requires READ privileges to RAW).
- Select (check) **Show Only Resources for this Role** to quickly see what resources are assigned to the current role.

Configure Resource Privileges for Packages

IDD applications use packages to display search results when executing queries on the Search tab.

Roles must be configured to have READ access to packages associated with the base object. To configure package permissions in the Roles tool, select the role you want to configure, expand the Packages node, and configure privileges for the applicable packages.

Role Name	Resource Privileges
party_no_privileges_role	No privileges.
party_read_only_role	READ privileges on the Party package.
party_create_role	READ privileges on the Party package.
party_update_role	READ privileges on the Party package.

Configure Resource Privileges for Cleanse Functions

If a subject area is configured to use an inline cleanse function (configured in the IDD configuration file), the role must have EXECUTE privileges on that cleanse function in order for the cleanse function to fire upon save.

Configure Resource Privileges for Custom Resources

Next, for each role (except the party_no_privileges_role), expand the Custom Resources node, expand the IDD application node, and assign the following privileges:

Role Name	Resource Privileges
party_no_privileges_role	No permissions.
party_read_only_role	- READ privileges to the CHART/View resource so that users can see charts in the Start workspace. - CREATE privileges to the SEARCH_QUERY/Create and SEARCH_QUERY/CreatePublic resource (or READ if you want users to run existing queries only and not create new queries). - READ privileges to the SUBJECT_AREA/Party resource.

Role Name	Resource Privileges
party_create_role	- READ privileges to the CHART/View resource so that users can see charts in the Start workspace. - READ and CREATE privileges to the SEARCH_QUERY/Create and SEARCH_QUERY/CreatePublic resources. - READ and UPDATE privileges on the SUBJECT_AREA/Party (only if you want to allow the role to bypass workflow altogether). Normally, users have READ and CREATE privileges on TASK_TYPE/Party: ReviewNoApprove, which gives users access to the **Send for Approval** button. - READ and UPDATE privileges to the SUBJECT_AREA/Party resource.
party_update_role	- READ privileges to the CHART/View resource so that users can see charts in the Start workspace. - READ and CREATE privileges to the SEARCH_QUERY/Create and SEARCH_QUERY/CreatePublic resources. - READ and UPDATE privileges on the SUBJECT_AREA/Party resource (only if you want to allow the role to bypass workflow altogether). Normally, users have READ and UPDATE privileges on TASK_TYPE/Party:ReviewNoApprove, which gives users access to the **Send for Approval** button.

How you configure access to these custom resources affects what users see in the IDD application. For example:

- If a user does not have CREATE privileges to SEARCH_QUERY/Create, they will not have the option to create or save a new query in IDD.
- If a user does not have CREATE privileges to SEARCH_QUERY/CreatePublic, they will not see the Public Query option in the Save Query As dialog.
- In general, users need to have READ and EXECUTE privileges on tasks that will be assigned to them. If a user does not have CREATE privileges to a given TASK_TYPE, they will not be able to create that task in IDD.

Additional Configuration Tips

- If you want to allow a role to merge and/or unmerge data, you need to grant that role MERGE privileges on the base object.
- If you want to allow a role to open records on the Hierarchy View tab, you need to grant them READ access on the HM_PROFILE resource (the Default profile or other applicable HM_PROFILE resource).

  Also grant the appropriate READ, CREATE, UPDATE and/or DELETE privileges on the HM_RELATIONSHIP_TYPE and HM_HIERARCHY_TYPE resource.

  To add an entity (Add Entity), the role must have CREATE privileges on the subject area. To add a relationship (Add Relationship), the role must have CREATE privileges on the REL table, and READ and CREATE privileges on the HM_PROFILE and READ and CREATE privileges on the HM_RELATIONSHIP_TYPE and HM_HIERARCHY_TYPE.

Assign Roles to Users (Users and Groups Tool)

In the Hub Console, use the Users and Groups tool to assign the IDD users to the roles you have defined.

User Account	Assign to Role
user_1	party_no_privileges_role
user_2	party_read_only_role
user_3	party_create_role
user_4	party_update_role

What Sample IDD Users Might Be Able To See and Do

Once roles have been assigned resource privileges to SECURE resources, and users have been assigned to roles, users can log into the IDD application and see what is available to them.

In this example, users might be able to see and do:

Role Name	What the User Can See and Do
user_1 (no privileges)	- On the Start workspace, the user cannot view charts. - On the Data tab, the user can see the Search tab but cannot actually view public queries or create queries. - On the Data tab, the user can see the various subject areas on the Data tab, but they cannot do anything with them.
user_2 (read-only privileges)	- On the Start workspace, the user can view charts. - On the Data tab (Search tab), the user can run a query, view public queries, and view the search results (including all fields for individual records), but they cannot create or update a query. - On the Data tab (Party subject area), the user cannot create a new record.
user_3 (create privileges)	- On the Start workspace, the user can view charts. - On the Data tab (Search tab), the user can run, create, and update a query. - On the Data tab (Party subject area), the user can create a new Party record, add data, and save changes.
user_4 (update privileges)	- On the Start workspace, the user can view charts. - On the Data tab (Search tab), the user can run, create, and update a query. - On the Data tab (Party subject area), the user can edit an existing Party record and save changes, but not create a new Party record.

APPENDIX F

Data Masking

This appendix includes the following topics:

Related Topics:

- "Data Masking" on page 22

Data Masking Overview

This appendix describes the Data Masking mechanism.

This mechanism is used for hiding critical information from IDD users that are not authorized to access that information. For masked fields, IDD replaces part of characters (or all field value) with asterisk (*).

Mask pattern is described in terms of regular expressions. Parts of expression that must be masked are in parenthesis.

Expressions

Mask pattern is described in terms of regular expressions.

Parts of expression that must be masked are in parenthesis.

.

Dot means any character.

.*

Dot followed by asterisk mean sequence of characters or empty sequence.

.+

Dot followed by plus sign mean one or more characters. Empty sequence is not matched by this expression.

.{n}

Dot followed by an integer number in curly brackets means up to n characters.

[.]

Dot in square brackets means dot character.

Sample Patterns

The following examples show sample patterns.

Mask whole field value:

```
(.+)
```

Mask all but last three characters:

```
(.+)…
```

Leave unmasked first four characters:

```
….(.+)
```

Pattern that hides five first characters then leave unmasked three, then hides rest of value except last four characters:

```
(.{5})…(.+)….
```

If the specified pattern does not matched field value, then the whole value is masked. For example string "ABS" doesn't match pattern `(.+)…` because it expects at least four characters (one at the beginning to be masked and three at the end to be left unmasked). In this case "ABS" is replaced with `"***"`.

Sample Mask Definition

Mask definitions can appear in XML configuration file in any Layout section.

```
<layout columnsNum="3">
   <column columnUid="C_PRODUCT|PRODUCT_NUMBER" editStyle="FIELD"
horizontalStyle="MEDIUM"
           required="true" showInHMCompactView="true">
      <dataMask value="...(.+)">
         <securityRole roleUid="Customer-CA"/>
      </dataMask>
   </column>
   <column 1columnUid="C_PRODUCT|PRODUCT_NAME" editStyle="FIELD" horizontalStyle="MEDIUM"
           Requi1red="true" showInHMCompactView="true"/>
   <column columnUid="C_PRODUCT|PRODUCT_DESC" editStyle="TEXT_AREA"
horizontalStyle="MEDIUM"/>
...
</layout>
```

The preceding example shows mask definition for column Product Number. Mask is applied to users with security role Customer-CA.

Note: If no security roles are defined for Data Mask definition, mask is applied for all non-admin users.

APPENDIX G

Siperian BPM Workflow Engine

This appendix includes the following topics:

Siperian BPM is Deprecated

Effective in version 10.0.0, the Siperian BPM workflow engine is deprecated and will be removed in a future release. Previously, the Siperian BPM workflow engine was the default workflow engine in the MDM Hub.

Informatica recommends that you update Data Director (IDD) applications to use the ActiveVOS Server.

For more information, see *Multidomain MDM Data Director Migration Guide*.

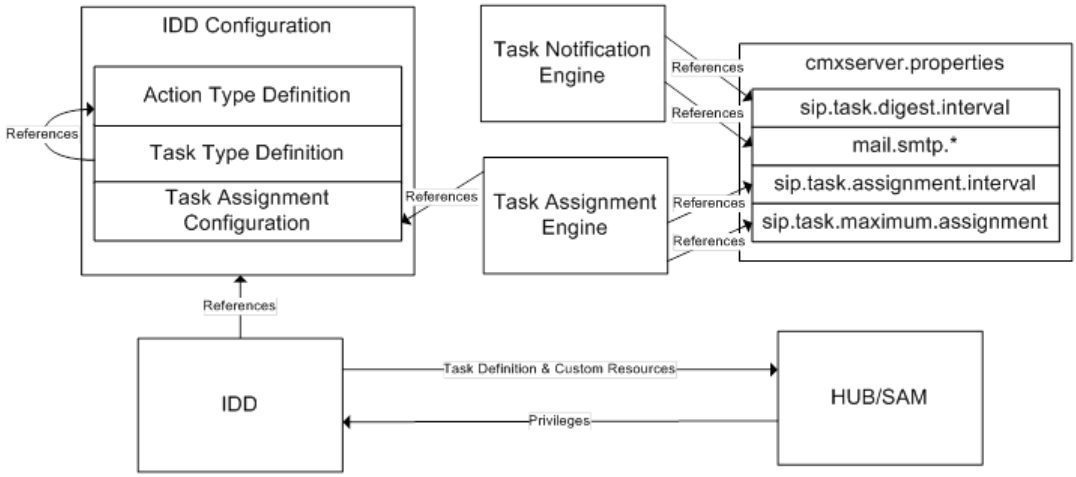
Workflows and Tasks

When using the legacy Siperian BPM tool or third-party BPM tools, you need to configure tasks and task management for your IDD application.

Note: This section does not apply to Informatica ActiveVOS, either embedded or standalone. The embedded version uses predefined tasks. The standalone version requires that you define tasks in Informatica ActiveVOS.

Workflow and Task Configuration Components Diagram

The following diagram shows the components of workflow and task configuration and their relationships.



Workflow and Task Configuration Component Descriptions

Component	Description
Action Type Definition	*Action types* are the reusable building blocks for tasks in a workflow. These define what a task will do when an action is performed in the context of the task. These are reusable because many tasks will provide similar subsets of actions that can be performed. **Note:** The Action Type Definition can be customized only in a very limited way in this release. However, additional customizability is planned for future releases.
Task Type Definition	*Task types* define the types of tasks that can be used to create workflows in an IDD application. This section of the configuration allows for the customization of which tasks are available, as well as of their general behavior. For more information, see "Task Types" later in this document.
Task Assignment Configuration	Used to specify the behavior of automatic and manual task assignment mechanisms. This is configured through the IDD Configuration Manager (see "IDD Configuration Manager" earlier in this document).

Component	Description
Task Notification Engine	Runs in the Informatica MDM Hub and sends email notifications to users on a configured interval.
Task Assignment Engine	Runs in the Informatica MDM Hub and periodically assigns any unassigned tasks to configured users.
cmxserver.properties file	Specifies multiple properties that can be set to configure task behavior. These properties are described in detail in the applicable sections later in this document.
IDD	The main application loads the configuration at startup (and deployment). The IDD also synchronizes the task configuration with SAM by creating task metadata and custom secure resources in the Informatica MDM Hub.
SAM	Provides information to the IDD on what privileges are granted to users for task types.

Note: When using workflows and tasks with an IDD application, task capabilities are only available if all base objects involved in a subject area have state management enabled in the Schema Manager in the Hub Console. This is required because certain tasks use pending records, which are available only when state management is enabled.

Task Configuration

Each IDD application is initialized with a default workflow and task definition.

Task assignments are configured in the IDD Configuration Manager. In many cases, the default definition will be adequate. However, task assignment configuration will always be required. Each of the following subsections focus on a part of this configuration.
Note: By default, the task configuration for IDD is a two-step approval process.

Task Types

This section of the IDD configuration file specifies the types of tasks that are available for use in an IDD application.

Task types are the most configurable task component. This section determines the behavior of tasks within Informatica MDM Hub, along with the flow from one task to the next.

The IDD default configuration includes seven predefined tasks:

Predefined Tasks	Description
UpdateWithApproval	Update a record and the next step requires the user to go through an approval process before completing the task.
UpdateWithOptionalApproval	Update a record and the next step does not require the user to go through an approval process before completing the task. The approval step is optional.

Predefined Tasks	Description
ReviewNoApprove	Review a change and either escalate or reject it. This task does not provide an Approve option and requires at least one other person to review the changes as well.
FinalReview	Review a change and approve, reject, or escalate.
Merge	Merge records together.
Unmerge	Unmerge an XREF record from a Base Object record.
UpdateRejectedRecord	Update a record that was rejected in an approval process.

Task Types - Sample XML

The following sample excerpt from the IDD configuration file pertains to task types (and will be referenced later in this subsection).

```
<!-- Task Definitions -->
        <taskType name="UpdateWithApproval" displayName="Update With Approval"
                  creationType="create">
            <description>Update a record and require the user to go through
                         an approval process before completing the task.
            </description>
            <action name="SubmitForApproval">
                <targetTask>ReviewNoApprove</targetTask>
            </action>
            <action name="Augment">
                <targetTask>UpdateWithApproval</targetTask>
            </action>
            <action name="CancelTask"/>
        </taskType>
         <taskType name="UpdateWithOptionalApproval" displayName="Update With Optional
Approval"
                   creationType="create">
            <description>Update a record and do not require the user to go through
                         an approval process before completing the task. The approval
step
                         is optional.
            </description>
            <action name="CompleteUpdate"/>
            <action name="SubmitForApproval">
                <targetTask>ReviewNoApprove</targetTask>
            </action>
            <action name="Augment">
                <targetTask>UpdateWithOptionalApproval</targetTask>
            </action>
            <action name="CancelTask"/>
        </taskType>
        <taskType name="ReviewNoApprove" displayName="Review" defaultApproval="true"
                  creationType="none" pendingBVT="true">
            <description>Review a change and either escalate or reject it. This task
                         does not provide an Approve option and requires at least one
                         other person to review the changes as well.
            </description>
            <action name="Reject">
                <targetTask>UpdateWithApproval</targetTask>
            </action>
            <action name="Escalate">
                <targetTask>FinalReview</targetTask>
            </action>
```

```
            <action name="Reassign">
                <targetTask>ReviewNoApprove</targetTask>
            </action>
            <action name="CancelTask"/>
        </taskType>
        <taskType name="FinalReview" displayName="Final Review" creationType="none"
                  pendingBVT="true">
            <description>Review a change and approve, reject or escalate it.</
description>
            <action name="Approve"/>
            <action name="Reject">
                <targetTask>UpdateWithApproval</targetTask>
            </action>
            <action name="Escalate">
                <targetTask>FinalReview</targetTask>
            </action>
            <action name="Reassign">
                <targetTask>FinalReview</targetTask>
            </action>
            <action name="CancelTask"/>
        </taskType>
        <taskType name="Merge" displayName="Merge" creationType="merge"
displayType=”merge”>
            <description>Merge two records together.</description>
            <action name="Reassign">
                <targetTask>Merge</targetTask>
            </action>
            <action name="CancelTask"/>
        </taskType>
        <taskType name="Unmerge" displayName="Unmerge" creationType="unmerge"
                  displayType=”unmerge”>
            <description>Unmerge an XREF record from a Base Object record.</description>
            <action name="Unmerge"/>
            <action name="Reassign">
                <targetTask>Unmerge</targetTask>
            </action>
            <action name="CancelTask"/>
        </taskType>
```

You can customize workflows and tasks by changing task type properties. Care should be used whenever the task definition is modified, because errors here can render tasks unusable in an IDD application. Task definition includes the following properties.

TaskType Attributes and Tags

name

The name attribute is the identifier of the task type. Do not use spaces and non-ASCII characters in the name attribute.

The name attribute is for internal use by an IDD application and Informatica MDM Hub, so there is no need to change these settings. If you introduce a new task type, specify any name as it will not be significant.

displayName

The displayName atribute specifies the name of the task that must appear in an IDD application.

However, the actual name shown in an IDD application is taken from a resource bundle, so changes to the displayName value may not result in corresponding visible changes in the deployed IDD application. Display name is used as the default value when IDD retrieves the localized display name from a resource bundle.

creationType

This attribute should not be modified for existing tasks.

This is used to determine where a task can be created in an IDD application. The possible values are:

creationType	Description
create	Tasks are created when the IDD application user chooses Create Task from a menu in an IDD application. **Note:** When creating a task using **More Actions** > **Create Task**, in the **Create Task** window, only those tasks configured as **CREATE** for the **Creation Type** option will be listed in the **Type** drop-down field.
merge	A task is created when the IDD application user chooses the command to create a task in the Potential Matches view. **Note:** Only one task type must have this designation.
unmerge	Tasks are created when the IDD application user chooses the command to create a task in the Cross References dialog. **Note:** Only one task type must have this designation.
none	Tasks cannot be created by an IDD application user in the IDD application. This designation means that these task types can only be created as a result of a workflow.

Example: The FinalReview task type has this designation in the previous code example because this task type can only be created as part of a flow (when the Escalate action is executed on a ReviewNoApprove task).

displayType

This attribute specifies how a task should be displayed when it is opened in the data view.

The possible values are:

displayType	Description
Normal	The task is opened in the data view with the task action menu available. The data view will present the data record associated with the task.
Merge	The task is opened in the data view with the task action menu available. The Potential Matches child tab is visible and selected in the data view. The potential match associated with the task is highlighted and automatically selected in the Potential Matches child tab.
unmerge	The task is opened in the data view with the task action menu available. The Cross References dialog is open on top of the data view. The cross-reference record to unmerge is selected in the dialog.

dataUpdateType

One of the following values.

dataUpdateType	Description
ACTIVE	Any modifications made to the record shown in the task view before executing this action are saved in the ACTIVE state.
PENDING	Any modifications made to the record shown in the task view before executing this action are saved in the PENDING state. This value is used for all approval flows to save changes as pending until they are approved.
NONE	Any modifications made to the record shown in the task view before executing this action will be lost. In this case, the IDD application user sees a confirmation dialog to make sure they want to discard any changes they have made to the record. Changes can be saved with the **Save** button in the data view before executing the task action.

pendingBVT

This attribute specifies whether the data view should include pending cross reference values when constructing the BVT view in an IDD application.

When set to true, any pending cross references that are referenced by the task will be included in the BVT view, the result being that the IDD application user gets a "what-if" view of the record as it would appear if the pending cross references were made active. This is useful for approving pending changes and trying to decide whether the resulting record would be correct.

defaultApproval

This attribute must be set to true on exactly one task type.

The task type that has a value of true for this attribute is the task type that will be created when the **Send for Approval** button is clicked in the data view in the IDD.

Note: If multiple task types have this attribute set to true, then unexpected results can occur if the task type created when the **Send for Approval** button is clicked.

Description Tag

This element provides a brief description of the purpose of the task type.

Action Tag

This element is a reference to an action type described in the following section.

Target Task Tag

This tag is optional in each task action.

When set, this specifies the name of the task type that represents the next step in the workflow when the encompassing action is executed.

Example: When the Escalate action is invoked on the ReviewNoApprove task type, then a new FinalReview task is created as the next step in the workflow.

Omitting this tag implies that the action will terminate the workflow process once executed.

Example: The cancel task action, present in each task type, will end the workflow.

Task Type Customization

Task types are highly customizable.

New task types can be created as long as the previously-described rules are followed. Existing flows can be modified by altering the values in the target task tags for a given task type. The following code snippet is an example of two-step approval process and one-step approval process.

```
<taskType creationType="NONE" dataUpdateType="ACTIVE"
            defaultApproval="false" displayName="Final Review"
            displayType="NORMAL" name="FinalReview" pendingBVT="true">
            <description>Review a change and approve, reject or escalate it.</
description>
            <action name="Approve"/>
            <action name="Reject">
                <targetTask>UpdateRejectedRecord</targetTask>
            </action>
            <action name="Escalate">
                <targetTask>FinalReview</targetTask>
            </action>
            <action name="Reassign">
                <targetTask>FinalReview</targetTask>
            </action>
            <action name="CancelTask"/>
        </taskType>
```

Action Types

This section of the IDD configuration file specifies the types of actions that are available for use by each task in an IDD application.

Each Task Type defines a set of possible actions that can be performed in the task context. Because multiple task types may have the same or similar actions available, the types of actions are defined outside of the context of a task, and are referenced from within the task type definition as described previously.

When you edit a task in the IDD Configuration Manager, in the **Task Configuration** window, you can configure the action types and the next step for each task. When working with a task in the IDD application, only the selected action types will be displayed as a button to the user and the task type selected in the **Next Step** section will be execute the next step in the workflow for that particular action type.

Note: For a selected action type, if the value in the **Next Step** section is **<Empty>**, the action will terminate the workflow process once executed.

The following table provides the list of action types and its description:

Action Types	Description
SubmitForApproval	Submit a change for approval.
Augment	Reassign the task to another user for assistance.

Action Types	Description
CompleteUpdate	Commit changes made to a subject area record.
Approve	Approve and commit changes made to a subject area record.
Reject	Reject changes and reassign to the user who made the changes.
Escalate	Reassign the task to another reviewer for assistance. This could result in a new task being created.
Reassign	Reassign the task to another user/role.
Unmerge	Perform the unmerge operation defined by the task.
CancelTask	Cancel the task by deleting it.

Action Types - Sample XML

The following excerpt from an IDD configuration file pertains to task types and will be referenced later in this subsection.

```
<!-- Action Definitions - MUST come before the task types definitions. -->
        <actionType name="SubmitForApproval" displayName="Submit For Approval">
            <description>Submit a change for approval.</description>
            <class>com.siperian.dsapp.domain.task.action.SubmitForApproval</class>
        </actionType>
        <actionType name="Augment" displayName="Augment" manualReassign="true">
            <description>Reassign the task to another user for assistance.</description>
            <class>com.siperian.dsapp.domain.task.action.Reassign</class>
        </actionType>
        <actionType name="CompleteUpdate" displayName="Complete Update">
            <description>Commit changes made to a subject area record.</description>
            <class>com.siperian.dsapp.domain.task.action.CompleteUpdate</class>
        </actionType>
        <actionType name="Approve" displayName="Approve">
            <description>Approve and commit changes made to a subject area record.</
description>
            <class>com.siperian.dsapp.domain.task.action.Approve</class>
        </actionType>
        <actionType name="Reject" displayName="Reject">
            <description>Reject changes and reassign to the user
                        who made the changes.</description>
            <class>com.siperian.dsapp.domain.task.action.Reject</class>
        </actionType>
        <actionType name="Escalate" displayName="Escalate">
            <description>Reassign the task to another reviewer for assistance.
                        This could result in a new task being created.</description>
            <class>com.siperian.dsapp.domain.task.action.Reassign</class>
        </actionType>
        <actionType name="Reassign" displayName="Reassign" manualReassign="true">
            <description>Reassign the task to another user/role.</description>
            <class>com.siperian.dsapp.domain.task.action.Reassign</class>
        </actionType>
        <actionType name="Unmerge" displayName="Unmerge">
            <description>Perform the unmerge operation defined by the task.</description>
            <class>com.siperian.dsapp.domain.task.action.Unmerge</class>
        </actionType>
        <actionType name="CancelTask" displayName="Cancel Task" cancelTask="true">
```

```
            <description>Cancel the task by deleting it.</description>
            <class>com.siperian.dsapp.domain.task.action.CancelTask</class>
</actionType>
```

ActionType Attributes and Tags

name

The name attribute of an action type should never be changed.

This is for internal use by an IDD application and Informatica MDM Hub, so there is no need to change these settings. If a new action type is introduced, any name can be specified, as it will not be significant.

displayName

This is the name of the action as it is displayed in an IDD application.

However, the actual name shown in an IDD application is taken from a resource bundle, so changes to this value may not result in corresponding visible changes in the IDD application.

Description Tag

This element provides a brief description of the purpose of the action type.

manualReassign

When this attribute is set to true, the IDD application user is prompted to select a specific user for assignment of the task before the action is performed.

This is used, for example, when manually reassigning a task to another user. If set to false, then task assignment for this action type is automatic.

closeTaskView

When this attribute is set to true, the tab in which the IDD application user was working when performing this action will be closed, and the user will be returned to the Start workspace page.

The following code snippet is an example of an action type.

```
<actionType cancelTask="true" closeTaskView="true"
            displayName="Cancel Task" manualReassign="false" name="CancelTask">
            <description>Cancel the task by deleting it.</description>
            <class>com.siperian.dsapp.domain.task.action.CancelTask</class>
        </actionType>
```

Note: You can configure this attribute for each action type using the IDD configuration file (IDDConfig.xml). The default value of this attribute is true.

cancelTask

When this attribute is set to true, the task is cancelled when this action is performed.

Consequently, the task is completely deleted and not recoverable, and any pending changes associated with the task are permanently deleted.

Class Tag

This attribute must NOT be modified in this release because it specifies the Java class that is used to execute the action.

There is no way to add custom action handlers in this release, but this functionality is planned in a future release.

Task Security Configuration

When an IDD application instance is deployed, or when the application server is restarted, the IDD application synchronizes a set of custom resources with Informatica MDM Hub.

This set of custom resources includes a custom resource for each subject area, and each task type per subject area (as configured in the IDD configuration file).

Use the Roles tool in the Hub Console to configure security for tasks by specifying privileges on the task type custom resources.

The following privileges for task types are applied in an IDD application:

Privilege	Description
Read	Unused.
Create	This privilege is required for an IDD application user to create new tasks. When the user chooses the Create Task command from the data view, the IDD application displays a dialog that contains a list of possible task types to create. This list contains only those task types for which the user has the create privilege. In addition, the tasks displayed in this list must also be configured properly in the IDD configuration file by setting the creationType attribute to "create".
Update	Unused.
Delete	Unused.
Merge	Unused.
Execute	This privilege is required for an IDD application user to view details about a task, and to make modifications to the task details (which includes adding comments, modifying the due date, and even reassigning the task). IDD application users who have execute privileges on a task type are allowed to execute all actions for that task type. This is true regardless of what the action does when it executes. For example, if there is an action that creates a new task, the user will be able to execute the action even if they do not have create privileges on the task type that is created by the action.

Important: The privileges for tasks, subject areas, and base objects all work together in SAM. An incorrect SAM configuration can lead to unexpected behavior in an IDD application. Task assignment (described below

and managed in the IDD Configuration Manager) is done by role or user. IDD does not verify that the role or user has the security configuration to allow operations on that task. It is up to the IDD application implementer to configure this correctly. Also, for an IDD application user to be able to cancel a task, the user must have the DELETE privilege on the XREF for each base object in a subject area.

Task Assignment

Task Assignment Configuration

Each Subject Area of the IDD application can be configured to use a specific set of task types.

Each task type can in turn be linked with one or more security roles or with a single user name. This means that task of a specific Task Type can be assigned only to users with the specified security roles, or to the user that is specified in the Task Assignment definition.

In the XML configuration file, task assignment can be defined using the `taskAssignmentConfig` tag.

Example:

```
<taskAssignmentConfig task="UpdateWithApproval">
                    <securityRole roleUid="DataSteward"/>
                </taskAssignmentConfig>
                <taskAssignmentConfig task="UpdateWithOptionalApproval" >
                    <securityRole roleUid="DataSteward"/>
                    <securityRole roleUid="Customer-NY"/>
                </taskAssignmentConfig>
                <taskAssignmentConfig task="UpdateRejectedRecord" user=”user1”/>
                <taskAssignmentConfig task="ReviewNoApprove">
                    <securityRole roleUid="Manager"/>
                </taskAssignmentConfig>
                <taskAssignmentConfig task="FinalReview" >
                    <securityRole roleUid="SrManager"/>
                </taskAssignmentConfig>
                <taskAssignmentConfig task="Merge">
                    <securityRole roleUid="DataSteward"/>
                </taskAssignmentConfig>
                <taskAssignmentConfig task="Unmerge">
                    <securityRole  roleUid="DataSteward"/>
                </taskAssignmentConfig>
```

In the preceding example, the `UpdateWithOptionalApproval` tasks can be assigned to users with either a Data Steward role or a Customer-NY role. Tasks of type UpdateRejectedRecord can be assigned only to one user (user1).

The task assignment element must contain the required `attribute` task, with the name of one of the task types defined in the IDD configuration. Additionally, it must contain either one or more child element's security Role, or user attribute with the name of a user, to whom the task of the particular type can be assigned.

Task Assignment Configuration UI

You can specify task assignment using the Task Assignment tab in the Subject Area dialog of the IDD.

When you click the **Task Assignment** tab, task types that can be used with the Subject Area are displayed. You can select a task type and click **Add** to add it to the Subject Area.

If all task types defined in the IDD application instance are already added to the Subject Area, then the Add button is disabled.

You can modify a selected Task type by using the **Edit** button. The **Delete** button lets you remove a task type from the Subject Area.

You must select the **Assign To Role** option, to modify or add roles. The security roles defined in the MDM Hub (using the MDM Hub) can be moved to the Selected Roles list and linked with Task Type for a Subject Area.

Automatic Task Assignment

Automatic task assignment is controlled through a server daemon that runs as part of the Informatica MDM Hub.

The frequency with which this runs is controlled by the value of the sip.task.assignment.interval property in cmxserver.properties. By default, this is set to 0, meaning that the daemon is disabled. It should only be enabled if you are running IDD applications and require task assignment. To enable the daemon, set sip.task.assignment.interval to a value in minutes. A value of 1 will cause the daemon to run once a minute. This daemon has two jobs:

It will assign any tasks that have no owner (null rowid_user) as configured in the task assignment configuration of the IDD application.

It will examine all match table entries associated with a configured subject area's primary table and create tasks to assign to available IDD application users.

An *available user* (a) has fewer than the configured maximum number of tasks currently assigned to them, and (b) is assigned the role specified in the task assignment configuration. You can configure the maximum number of tasks that can be automatically assigned to an IDD application user by specifying the sip.task.maximum.assignment property in the cmxserver.properties file. The default maximum number of tasks to assign per user is 25.

When tasks are automatically assigned, IDD application users from the configured role are selected for task assignment in a round-robin fashion until there are no more users who have fewer than the maximum allowed tasks assigned. Each time the assignment daemon runs, it will assign any unassigned tasks that it can. If there are not enough users to receive all unassigned tasks, there might be unassigned tasks remaining after the daemon runs (which will be assigned once space becomes available in a target IDD application user's task queue). When automatic task assignment is occurring, the IDD application user who receives a specific task cannot be predicted with certainty. If a task needs to be assigned to a specific user, then manual assignment should be used.

Customizing Automatic Task Assignment

Automatic task assignment can be customized through the AssignTasks user exit.

The AssignTasks user exit works with the Siperian BPM workflow adapter.

Manual Task Assignment

Manual task assignment is controlled by the IDD application user in the IDD application.

When creating tasks, users have the option to select a target user for the task. If specified, the selected user becomes the owner of the newly-created task. If left blank, the automatic assignment daemon assigns the task to the next available user.

Customizing Task Assignment

Manual task assignment can be customized through the GetAssignableUsersForTasks user exit.

The GetAssignableUsersForTasks user exit works with the Siperian BPM workflow adapter.

Changing Assigned Tasks

IDD applications can administer task assignments in the task administration tab.

If a task is assigned to a user who is out of the office, for example, an administrator can use the IDD application to assign their tasks to another user.

If a user is going to be unavailable for a period of time, you can prevent automatic task assignment to this user by removing that user from the role.

Task Notification

Task notification is straightforward.

At a configured interval, a digest email can be sent to users who own tasks. The daemon runs as part of the Informatica MDM Hub. The interval at which notifications are sent can be configured as a specified number of hours in the cmxserver.properties file with the sip.task.digest.interval property. The default notification interval is 0 hours, meaning that digests are disabled. To enable digests, change this to a value in hours.

The sample email digest is as given below:

```
From: siperian_task_notification@siperian.com
To: null
Subject: Data Steward Task Digest for admin
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit

Tasks completed since last notification: 0

Total Assigned Tasks: 17

This message was sent by the Siperian Hub Server Task Notification Daemon.
```

Note: You cannot customize the body of the email digest.

Configuring the Task Notification Email

To configure the task notification email, edit the properties in the file cmxserver.properties. You must configure an outgoing SMTP server location.

The following list describes the task notification email properties that you can configure in the file cmxserver.properties:

mail.smtp.sender

The email address of the sender. The default is siperian_task_notification@siperian.com.

mail.smtp.host

The mail server host name.

mail.smtp.port

The mail server port number.

mail.smtp.auth

Determines whether the specified mail server requires authentication for outgoing messages. You must set mail.smtp.auth to true if you are using the Informatica MDM Hub mail server.

mail.smtp.user

The user name for the outgoing mail server. You must specify a value for mail.smtp.user if mail.smtp.auth is true.

mail.smtp.password

The password for the specified mail.smtp.user. You must specify a value for mail.smtp.password if mail.smtp.auth is true.

User Manager Configuration in the Hub Console

In addition, in order for IDD application users to receive email, the incoming email account must be set in Informatica MDM Hub.

In the Users tool of the Hub Console, specify the email address to which notifications for the IDD user should be sent. An email will be sent only if there are tasks assigned to an IDD application user.

Reports and Task Management Metrics

Task management metrics show the distribution of Data Director tasks.

Administrators can use task management metrics to show the distribution of Informatica Data Director tasks. You can generate reports based on the following task management metrics:

Task subject area by create date

Users can view create date trends for a subject area. Use `task_sa_by_create_date` as the report name to configure and populate the data mart for this report.

Task subject area by due date

Users can view reports based on the number of records with due dates ranges that the administrator specifies when the report is configured. Use `task_sa_by_due_date` as the report name to configure and populate the data mart for this report.

Task subject area by priority

Users can view reports based on the number of records with priorities such as "Low," "Medium," and "High." Use `task_sa_by_priority` as the report name to configure and populate the data mart for this report.

Task subject area by status

Users can view reports based on the number of records with due date statuses such as "On Time" and "Overdue." Use `task_sa_by_status` as the report name to configure and populate the data mart for this report.

Task subject area by task type

Users can view reports based on the number of records with task types such as "Update with Approval," "Merge", and "Unmerge." Use `task_sa_by_task_type` as the report name to configure and populate the data mart for this report.

Task by subject area

Users can view reports based on the number of records with subject areas such as "Person," "Organization," and "Product." Use `task_by_subject_area` as the report name to configure and populate the data mart for this report.

Task assignee by priority

Users can view reports based on the number of records for an assignee that have priorities such as "Low," "Medium," and "High." Use `task_assignee_by_priority` as the report name to configure and populate the data mart for this report.

Task assignee by status

Users can view reports based on the number of records for an assignee that have due date statuses such as "On Time" and "Overdue." Use `task_assignee_by_status` as the report name to configure and populate the data mart for this report.

Data Security in Task Data

IDD allows authorized users to participate in workflows, which are computer models of real work involving a series of operations or activities. IDD data security impacts the following task areas:

- Check for view permissions - whether a task can be opened by a user or not. If the user is not allowed to open a task, a warning message is displayed to the user.
- Filter out children data - which child records are seen by user on Data View.

Note:

- Data security filters do not get applied on XREF data. For example, if a user has access to the primary object data and to its children data, then according to data security the user is allowed to see all the contributing XREFs.
- The logic of applying data security filters depends on the type of task.

Review Task

The main difference between opening the regular primary objects and reviewing the tasks on the data view is that, the security filters are applied not to an active state of primary object and its children. Data security is applied after the preview BVT function has been executed to the overall pending records associated with the task.

Open Review Tasks with a Single Role

A User with a single role can open a task only if the following conditions are satisfied:

- All pending records associated with the task should satisfy the data security filters.
- If there are multiple filters on a single column for a single role, then user will have access to a union of all data that meet each filter.

- If there are filters on multiple columns for a single role, then user will have access to intersection of all data that meet each filter.
- If there are security filters configured over the children or grandchildren records, then one of the following conditions must be true:
  - Primary object has at least one record passing security restrictions in each child tab with enabled data security.
  - There is a pending record associated with the task, which belongs to child tab, with data security enabled and meets the data security settings according to the preceding condition.

For example, consider a data security model in which the user has the role, SalesManager- NY and has the following security filters configured:

- Filter 1: State code is NY.
- Filter 2: Phone type is Business and Home.
- Filter 3: Person salutation code is MR.

Using the data security model mentioned above, consider a scenario where the database has a primary object record, Mr. Florian Amadeu, who has the Billing address in NY state and facsimile as phone type. User with no data security restrictions adds a new Business phone and creates **send for approval** task. User with Sales Manager- NY role will be able to open the Mr. Florian Amadeu record on the data view as it satisfies all three conditions above, PO itself satisfies Data Security (Filter3), and has at least one record in every child where Data Security is enabled – NY address (active record) and BUSINESS phone (pending record).

Using the same data security model mentioned above, consider a scenario where the database has a primary object record, Mr. Dominic Wilkins, who has the Billing Address in NY State and no phone type. User with no data security restrictions adds a new Business phone and creates **send for approval** task. User with Sales Manager- NY role will be unable to open the task as the user has no phone that does meet Filter2.

Open Review Tasks with Multiple Roles

User with multiple roles can open the task only if the following conditions are satisfied.

- All the pending records associated with task should satisfy data security filters for at least one user role.
- A user with multiple roles can have combinations of filters applied. The result is that the user has access to all the data available in each assigned role - a union of the filter assignments.
- If security filters are configured over children or grandchildren, then one of the following conditions must be true.
  - Primary object has at least one record passing security restrictions in each child tab with enabled data security.
  - The pending record associated with the task has a child tab with data security enabled and meets the data security setting as mentioned in the preceding condition.

For example, consider a data security model in which the user has the role, Sales Manager- NY and has the data security filters as mentioned in the section "Open Review Tasks with a Single Role" on page 146 and also role, CarSalesManager-NJ, which has the following security filters configured.

- Filter 1: State code is NJ.
- Filter 2: Car Year is 2009.

Also, user has another role CarSalesManager-CA, which has the following security filter configured.

- Filter 1: Address State code is CA.
- Filter 2: Car year is 2008.

Using the data security model mentioned above, consider a scenario where the database has a primary object record, Mr. Derrick Rose, who has the Billing address in CA State and home as phone type. User with no data security restrictions adds a new billing address in NY State and creates **Send for Approval** task. User with Sales Manager- NY role will be able to open the Mr. Derrick Rose record on the data view as it satisfies security filters of SalesManager-NY role.

Using the same data security model mentioned above, consider a scenario where the database has a primary object record, Mr. Tyros Thomas, who has the Billing Address in CA State and a car produced in 2008. User with no data security restrictions changed the billing address to NJ and creates **Send for Approval** task. User with both the roles both the roles CarSalesManager-CA and CarSalesManager-NJ is not allowed to open the task as Mr. Tyros Thomas does not satisfy filters for CarSalesManager-CA and CarSalesManager-NJ with pending record for new address.

Filter Child Record in the Task View

IDD applies security filters when retrieving data for children tabs on the data view. For example, consider a data security model in which the user has the role, SalesManager-CA and has the security filter billing address state code – CA.

Using the data security model mentioned above, consider a scenario where the database has a primary object record, Mr. Blake Griffin, who has two Billing address in New York and Bloomfield Hills cities, both in NY State. User with no data security restrictions, changes the state value for Bloomfield Hills to CA and creates additional billing Address in LA (CA State) and then creates **Send for Approval** task. User with Sales Manager- NY role will be able to open the Mr. Blake Griffin record on the data view and can see two CA addresses in **Bill Address** tab. One of them is former NY address that is changed, the second one is the new address that is added. NY address that is not changed is filtered out when security filters are applying on preview BVT.

Open Merge/Unmerge Tasks

IDD applies the following rules to determine whether the merge or unmerge task can be opened by the user.

- Merge task can be opened, only if all the Primary Object that are to be merged meet the data security settings.
- Unmerge task can be opened if Primary Object can be opened according to Data Security settings.

For example, consider a data security model in which the user has the role, SalesManager-CA and has the security filter billing address state code – CA.

Using the data security model mentioned above, consider a scenario where there are two persons in database with same name Kevin Durant. One of the individuals has one billing address in LA (CA State), and the other one has a Bill Address in New York (NY State). User with no data security restrictions creates **Merge** task for two person records. Users with SalesManager- CA role will not be able to open the task as they do not have the required permission to open a person record with billing Address in NY State and thus cannot perform a whole **Merge** task.

Data Aware Task Assignment

When assigning a task in the **Assign Task** dialog box, IDD filters out task reviewers who do not have the privilege to open the task. Also, for automatic task assignment, daemon assigns the task only to users who have the privilege to open the task.

APPENDIX H

Locale Codes

This appendix includes the following topics:

Language Codes

ISO Code	Language
aa	Afar
ab	Abkhazian
af	Afrikaans
am	Amharic
ar	Arabic
as	Assamese
ay	Aymara
az	Azerbaijani
ba	Bashkir
be	Byelorussian
bg	Bulgarian
bh	Bihari
bi	Bislama
bn	Bengali, Bangla
bo	Tibetan

ISO Code	Language
br	Breton
ca	Catalan
co	Corsican
cs	Czech
cy	Welsh
da	Danish
de	German
dz	Bhutani
el	Greek
en	English
eo	Esperanto
Es	Spanish
et	Estonian
eu	Basque
fa	Persian
fi	Finnish
fj	Fiji
fo	Faroese
fr	French
fy	Frisian
ga	Irish
gd	Scots Gaelic
gl	Galician
gn	Guarani
gu	Gujarati
ha	Hausa
he	Hebrew (formerly iw)

ISO Code	Language
hi	Hindi
hr	Croatian
hu	Hungarian
hy	Armenian
ia	Interlingua
id	Indonesian (formerly in)
ie	Interlingue
ik	Inupiak
is	Icelandic
it	Italian
iu	Inuktitut
ja	Japanese
jw	Javanese
ka	Georgian
kk	Kazakh
kl	Greenlandic
km	Cambodian
kn	Kannada
ko	Korean
ks	Kashmiri
ku	Kurdish
ky	Kirghiz
la	Latin
ln	Lingala
lo	Laothian
lt	Lithuanian
lv	Latvian, Lettish

ISO Code	Language
mg	Malagasy
mi	Maori
mk	Macedonian
ml	Malayalam
mn	Mongolian
mo	Moldavian
mr	Marathi
ms	Malay
mt	Maltese
my	Burmese
na	Nauru
ne	Nepali
nl	Dutch
no	Norwegian
oc	Occitan
om	(Afan) Oromo
or	Oriya
pa	Punjabi
pl	Polish
ps	Pashto, Pushto
pt	Portuguese
qu	Quechua
rm	Rhaeto-Romance
rn	Kirundi
ro	Romanian
ru	Russian
rw	Kinyarwanda

ISO Code	Language
sa	Sanskrit
sd	Sindhi
sg	Sangho
sh	Serbo-Croatian
si	Sinhalese
sk	Slovak
sl	Slovenian
sm	Samoan
sn	Shona
so	Somali
sq	Albanian
sr	Serbian
ss	Siswati
st	Sesotho
su	Sudanese
sv	Swedish
sw	Swahili
ta	Tamil
te	Telugu
tg	Tajik
th	Thai
ti	Tigrinya
tk	Turkmen
tl	Tagalog
tn	Setswana
to	Tonga
tr	Turkish

ISO Code	Language
ts	Tsonga
tt	Tatar
tw	Twi
ug	Uighur
uk	Ukrainian
ur	Urdu
uz	Uzbek
vi	Vietnamese
vo	Volapuk
wo	Wolof
xh	Xhosa
yi	Yiddish (formerly ji)
yo	Yoruba
za	Zhuang
zh	Chinese
zu	Zulu

Related Topics:

- "Lookup Tables With Subtype Column" on page 60

Country Codes

Country	Two-letter Code	ISO #
AALAND ISLANDS	AX	248
AFGHANISTAN	AF	4
ALBANIA	AL	8
ALGERIA	DZ	12

Country	Two-letter Code	ISO #
AMERICAN SAMOA	AS	16
ANDORRA	AD	20
ANGOLA	AO	24
ANGUILLA	AI	660
ANTARCTICA	AQ	10
ANTIGUA AND BARBUDA	AG	28
ARGENTINA	AR	32
ARMENIA	AM	51
ARUBA	AW	533
AUSTRALIA	AU	36
AUSTRIA	AT	40
AZERBAIJAN	AZ	31
BAHAMAS	BS	44
BAHRAIN	BH	48
BANGLADESH	BD	50
BARBADOS	BB	52
BELARUS	BY	112
BELGIUM	BE	56
BELIZE	BZ	84
BENIN	BJ	204
BERMUDA	BM	60
BHUTAN	BT	64
BOLIVIA	BO	68
BOSNIA AND HERZEGOWINA	BA	70
BOTSWANA	BW	72
BOUVET ISLAND	BV	74
BRAZIL	BR	76

Country	Two-letter Code	ISO #
BRITISH INDIAN OCEAN TERRITORY	IO	86
BRUNEI DARUSSALAM	BN	96
BULGARIA	BG	100
BURKINA FASO	BF	854
BURUNDI	BI	108
CAMBODIA	KH	116
CAMEROON	CM	120
CANADA	CA	124
CAPE VERDE	CV	132
CAYMAN ISLANDS	KY	136
CENTRAL AFRICAN REPUBLIC	CF	140
CHAD	TD	148
CHILE	CL	152
CHINA	CN	156
CHRISTMAS ISLAND	CX	162
COCOS (KEELING) ISLANDS	CC	166
COLOMBIA	CO	170
COMOROS	KM	174
CONGO, Democratic Republic of (was Zaire)	CD	180
CONGO, Republic of	CG	178
COOK ISLANDS	CK	184
COSTA RICA	CR	188
COTE D'IVOIRE	CI	384
CROATIA (local name: Hrvatska)	HR	191
CUBA	CU	192
CYPRUS	CY	196
CZECH REPUBLIC	CZ	203

Country	Two-letter Code	ISO #
DENMARK	DK	208
DJIBOUTI	DJ	262
DOMINICA	DM	212
DOMINICAN REPUBLIC	DO	214
ECUADOR	EC	218
EGYPT	EG	818
EL SALVADOR	SV	222
EQUATORIAL GUINEA	GQ	226
ERITREA	ER	232
ESTONIA	EE	233
ETHIOPIA	ET	231
FALKLAND ISLANDS (MALVINAS)	FK	238
FAROE ISLANDS	FO	234
FIJI	FJ	242
FINLAND	FI	246
FRANCE	FR	250
FRENCH GUIANA	GF	254
FRENCH POLYNESIA	PF	258
FRENCH SOUTHERN TERRITORIES	TF	260
GABON	GA	266
GAMBIA	GM	270
GEORGIA	GE	268
GERMANY	DE	276
GHANA	GH	288
GIBRALTAR	GI	292
GREECE	GR	300
GREENLAND	GL	304

Country	Two-letter Code	ISO #
GRENADA	GD	308
GUADELOUPE	GP	312
GUAM	GU	316
GUATEMALA	GT	320
GUINEA	GN	324
GUINEA-BISSAU	GW	624
GUYANA	GY	328
HAITI	HT	332
HEARD AND MC DONALD ISLANDS	HM	334
HONDURAS	HN	340
HONG KONG	HK	344
HUNGARY	HU	348
ICELAND	IS	352
INDIA	IN	356
INDONESIA	ID	360
IRAN (ISLAMIC REPUBLIC OF)	IR	364
IRAQ	IQ	368
IRELAND	IE	372
ISRAEL	IL	376
ITALY	IT	380
JAMAICA	JM	388
JAPAN	JP	392
JORDAN	JO	400
KAZAKHSTAN	KZ	398
KENYA	KE	404
KIRIBATI	KI	296
KOREA, DEMOCRATIC PEOPLE'S REPUBLIC OF	KP	408

Country	Two-letter Code	ISO #
KOREA, REPUBLIC OF	KR	410
KUWAIT	KW	414
KYRGYZSTAN	KG	417
LAO PEOPLE'S DEMOCRATIC REPUBLIC	LA	418
LATVIA	LV	428
LEBANON	LB	422
LESOTHO	LS	426
LIBERIA	LR	430
LIBYAN ARAB JAMAHIRIYA	LY	434
LIECHTENSTEIN	LI	438
LITHUANIA	LT	440
LUXEMBOURG	LU	442
MACAU	MO	446
MACEDONIA, THE FORMER YUGOSLAV REPUBLIC OF	MK	807
MADAGASCAR	MG	450
MALAWI	MW	454
MALAYSIA	MY	458
MALDIVES	MV	462
MALI	ML	466
MALTA	MT	470
MARSHALL ISLANDS	MH	584
MARTINIQUE	MQ	474
MAURITANIA	MR	478
MAURITIUS	MU	480
MAYOTTE	YT	175
MEXICO	MX	484
MICRONESIA, FEDERATED STATES OF	FM	583

Country	Two-letter Code	ISO #
MOLDOVA, REPUBLIC OF	MD	498
MONACO	MC	492
MONGOLIA	MN	496
MONTSERRAT	MS	500
MOROCCO	MA	504
MOZAMBIQUE	MZ	508
MYANMAR	MM	104
NAMIBIA	NA	516
NAURU	NR	520
NEPAL	NP	524
NETHERLANDS	NL	528
NETHERLANDS ANTILLES	AN	530
NEW CALEDONIA	NC	540
NEW ZEALAND	NZ	554
NICARAGUA	NI	558
NIGER	NE	562
NIGERIA	NG	566
NIUE	NU	570
NORFOLK ISLAND	NF	574
NORTHERN MARIANA ISLANDS	MP	580
NORWAY	NO	578
OMAN	OM	512
PAKISTAN	PK	586
PALAU	PW	585
PALESTINIAN TERRITORY, Occupied	PS	275
PANAMA	PA	591
PAPUA NEW GUINEA	PG	598

Country	Two-letter Code	ISO #
PARAGUAY	PY	600
PERU	PE	604
PHILIPPINES	PH	608
PITCAIRN	PN	612
POLAND	PL	616
PORTUGAL	PT	620
PUERTO RICO	PR	630
QATAR	QA	634
REUNION	RE	638
ROMANIA	RO	642
RUSSIAN FEDERATION	RU	643
RWANDA	RW	646
SAINT HELENA	SH	654
SAINT KITTS AND NEVIS	KN	659
SAINT LUCIA	LC	662
SAINT PIERRE AND MIQUELON	PM	666
SAINT VINCENT AND THE GRENADINES	VC	670
SAMOA	WS	882
SAN MARINO	SM	674
SAO TOME AND PRINCIPE	ST	678
SAUDI ARABIA	SA	682
SENEGAL	SN	686
SERBIA AND MONTENEGRO	CS	891
SEYCHELLES	SC	690
SIERRA LEONE	SL	694
SINGAPORE	SG	702
SLOVAKIA	SK	703

Country	Two-letter Code	ISO #
SLOVENIA	SI	705
SOLOMON ISLANDS	SB	90
SOMALIA	SO	706
SOUTH AFRICA	ZA	710
SOUTH GEORGIA AND THE SOUTH SANDWICH ISLANDS	GS	239
SPAIN	ES	724
SRI LANKA	LK	144
SUDAN	SD	736
SURINAME	SR	740
SVALBARD AND JAN MAYEN ISLANDS	SJ	744
SWAZILAND	SZ	748
SWEDEN	SE	752
SWITZERLAND	CH	756
SYRIAN ARAB REPUBLIC	SY	760
TAIWAN	TW	158
TAJIKISTAN	TJ	762
TANZANIA, UNITED REPUBLIC OF	TZ	834
THAILAND	TH	764
TIMOR-LESTE	TL	626
TOGO	TG	768
TOKELAU	TK	772
TONGA	TO	776
TRINIDAD AND TOBAGO	TT	780
TUNISIA	TN	788
TURKEY	TR	792
TURKMENISTAN	TM	795
TURKS AND CAICOS ISLANDS	TC	796

Country	Two-letter Code	ISO #
TUVALU	TV	798
UGANDA	UG	800
UKRAINE	UA	804
UNITED ARAB EMIRATES	AE	784
UNITED KINGDOM	GB	826
UNITED STATES	US	840
UNITED STATES MINOR OUTLYING ISLANDS	UM	581
URUGUAY	UY	858
UZBEKISTAN	UZ	860
VANUATU	VU	548
VATICAN CITY STATE (HOLY SEE)	VA	336
VENEZUELA	VE	862
VIET NAM	VN	704
VIRGIN ISLANDS (BRITISH)	VG	92
VIRGIN ISLANDS (U.S.)	VI	850
WALLIS AND FUTUNA ISLANDS	WF	876
WESTERN SAHARA	EH	732
YEMEN	YE	887
ZAMBIA	ZM	894
ZIMBABWE	ZW	716

APPENDIX I

Troubleshooting

This appendix includes the following topics:

Troubleshooting Overview

This appendix describes some tips about what to check for when encountering unexpected results in your IDD application configuration.

Check Your SAM Configuration

Verify that SAM has the correct permissions assigned at all levels necessary in accordance with the documentation.

Areas to check for CRUD include:

- If cross-references and changes histories are required, with buttons enabled in the IDD application, then the appropriate metadata content (XREF and HIST objects) are SECURE resources and are also configured accordingly.
- Queries/Packages - Ensure that resources are made SECURE. Otherwise, an IDD application might not allow access to the entire subject area.

Check Your Cleanse Function Configuration

If cleanse functions are configured, ensure that:

- Each cleanse function is a SECURE resource.
- Any roles that need access to the cleanse function has Execute permission.

Informatica Data Director Metadata Has Not Updated

Informatica Data Director maintains a cache of MDM Hub metadata that describes base objects, columns, relationships, and other details. To clear the cache for the selected IDD application, and force IDD to reload the metadata, click **Clear Cache** in the IDD Configuration Manager.

You can also restart the application server to clear the cache.

Informatica Data Director Stops Responding When You Switch Entities

Informatica Data Director stops responding when you switch entities for hierarchy manager relationships for source systems that are not enabled for state management override.

The behavior occurs for JBoss environments running on Java 1.7. To resolve this issue, you must configure the `standalone-full.xml` file.

1. Open the `standalone-full.xml` file for editing. The file is in the following directory:
   - On UNIX. `<JBoss installation directory>/jboss-eap-6.1/standalone/configuration`
   - On Windows. `<JBoss installation directory>\jboss-eap-6.1\standalone\configuration`
2. Add the following XML code to the `standalone-full.xml` file to configure asynchronous handling for the logger:

```
<async-handler name="ASYNC">
   <level name="INFO"/>
   <queue-length value="1024"/>
   <overflow-action value="BLOCK"/>
      <subhandlers>
         <handler name="FILE"/>
         <handler name="CONSOLE"/>
      </subhandlers>
</async-handler>
```

3. Under <subsystem xmlns="urn:jboss:domain:logging:1.2"> in the `standalone-full.xml` file, add the following XML code to configure asynchronous handling for the root logger:

```
<root-logger>
   <level name="INFO"/>
   <handlers>
      <handler name="ASYNC"/>
   </handlers>
</root-logger>
```

4. Restart the application server.

Informatica Data Director Configuration Is Not Valid

If you receive an error that the Informatica Data Director configuration is not valid, validate the `IDDConfig.xml` file against the `siperian-bdd-config-6.xsd` schema.

The `siperian-bdd-config-6.xsd` schema is in the resource kit in the following directory:

- On UNIX. `<infamdm installation directory>/hub/resourcekit/sdk/bddXsdDoc`
- On Windows. `<infamdm installation directory>\hub\resourcekit\sdk\bddXsdDoc`

Match Performance is Very Slow

Users of the IDD application report that match performance is very slow.

Enable the `needLoadChildOnOpen` property and restart the application server.

To enable the property, run the following SQL statements on the ORS database:

```
insert into C_REPOS_DS_PREF_DETAIL (ROWID_DS_PREF_DETAIL, ROWID_DS_PREF, NAME, VALUE) select
'BDDGP.30', rowid_ds_pref, 'needLoadChildOnOpen', 'true' from C_REPOS_DS_PREF where name =
'___SYSTEM_PREFERENCES_ROOT___';

commit;
```

APPENDIX J

Glossary

administrator

IDD application user who has the primary responsibility for configuring the IDD application.

authentication

Process of verifying the identity of a user to ensure that they are who they claim to be. In IDD application, users are authenticated based on their supplied credentials—user name / password, security payload, or a combination of both. IDD application provides an internal authentication mechanism and also supports user authentication using third-party authentication providers.

auxiliary files

Auxiliary files are temporary files and that are created in different circumstances, when editing or exporting a project.

base object

A table that contains information about an entity that is relevant to your business, such as customer or account.

best version of the truth (BVT)

A record that has been consolidated with the best cells of data from the source records. Sometimes abbreviated as BVT.

business entity

A nested structure of base objects. Use the Entity 360 framework in Informatica Data Director to view all the information that relates to the root base object of a business entity. Perform a search in Informatica Data Director to find data within a business entity.

business entity service

A business entity service is a set of operations that run MDM Hub code to create, update, delete, and search for base object records in a business entity.

business process

A business process is a workflow that achieves an organizational goal and implements a business function. A business process contains the activities that are required to achieve the goal and defines paths of execution through the activities. Multidomain MDM ships with predefined Informatica ActiveVOS business processes that are managed by the ActiveVOS Server. The organizational goal of these processes is to ensure that authorized personnel, such as business managers or data stewards, review all updates to master data.

business process management (BPM)

Business process management focuses on adapting an organization's processes. Informatica MDM ships with an embedded business process management engine. Using this engine, you can automate the review-and-approval processes for master data.

cleanse function

IDD allows you to use cleanse functions already defined in MDM to cleanse, standardize and validate input data. You can use this function for address standardization and validation and also for data augmentation from other sources.

Cleanse Match Server

The Cleanse Match Server run-time component is a servlet that handles cleanse requests. This servlet is deployed in an application server environment. The servlet contains two server components:

- a cleanse server that handles data cleansing operations
- a match server that handles match operations

The Cleanse Match Server is multi-threaded so that each instance can process multiple requests concurrently. It can be deployed on a variety of application servers.

The Cleanse Match Server interfaces with any of the supported cleanse engines, such as the Trillium Director cleanse engine. The Cleanse Match Server and the cleanse engine work to standardize the data. This standardization works closely with the Informatica Consolidation Engine (formerly referred to as the Merge Engine) to optimize the data for consolidation.

content metadata

Data that describes the business data that has been processed by Informatica MDM Hub. Content metadata is stored in support tables for a base object, including cross-reference tables, history tables, and others. Content metadata is used to help determine where the data in the base object came from, and how the data changed over time.

Custom Login Provider

Is a pluggable module that authenticates users, when IDD application is started.

database

Organized collection of data in the Hub Store. Informatica MDM Hub supports two types of databases: a Master Database and an Operational Reference Store (ORS).

data cleansing

The process of standardizing data content and layout, decomposing and parsing text values into identifiable elements, verifying identifiable values (such as zip codes) against data libraries, and replacing incorrect values with correct values from data libraries.

Data governance

Data governance is the practice of managing data as a corporate asset across the enterprise. It involves processes, policies, standards, technologies, and people across the organization to ensure the availability of accurate, consistent and timely data for better decision making and improved business processes.

Data Masking

Is a mechanism for hiding information based on security roles.

Data model

Data model is an abstract model that describes how data is structured and organized.

Data security

Data security is restricting users from seeing certain records based on the contents of those records.

datasource

In the application server environment, a datasource is a JDBC resource that identifies information about a database, such as the location of the database server, the database name, the database user ID and password, and so on. Informatica MDM Hub needs this information to communicate with an ORS.

data steward

IDD application user who has the primary responsibility for data quality.

data type

Defines the characteristics of permitted values in a table column—characters, numbers, dates, binary data, and so on.

deduplicate

Is a technique to eliminate redundant data.

design object

Parts of the metadata used to define the schema and other configuration settings for an implementation. Design objects include instances of the following types of Informatica MDM Hub objects: base objects and columns, landing and staging tables, columns, indexes, relationships, mappings, cleanse functions, queries and packages, trust settings, validation and match rules, Security Access Manager definitions, Hierarchy Manager definitions, and other settings.

duplicate

One or more records in which the data in certain columns (such as name, address, or organization data) is identical or nearly identical. Match rules executed during the match process determine whether two records are sufficiently similar to be considered duplicates for consolidation purposes.

entity

An entity is any object, person, place, or thing that has meaning and can be acted upon in your database.

External Login Provider

A plug-in that is used with IDD to authenticate users against external identity providers.

foreign key

In a relational database, a column (or set of columns) whose value corresponds to a primary key value in another table (or, in rare cases, the same table). The foreign key acts as a pointer to the other table. For example, the Department_Number column in the Employee table would be a foreign key that points to the primary key of the Department table.

fuzzy match

A match / search strategy that uses probabilistic matching, which takes into account spelling variations, possible misspellings, and other differences that can make matching records non-identical.

fuzzy match key

Special column in the base object that the Schema Manager adds if a match column uses the fuzzy match / search strategy. This column is the primary field used during searching and matching to generate match candidates for this base object. All fuzzy base objects have one and only one Fuzzy Match Key.

hierarchy

In Hierarchy Manager, a set of relationship types. These relationship types are not ranked based on the place of the entities of the hierarchy, nor are they necessarily related to each other. They are merely relationship types that are grouped together for ease of classification and identification.

Hierarchy Manager

The Hierarchy Manager allows users to manage hierarchy data that is associated with the records managed in the MDM Hub. For more information, see the *Multidomain MDM Configuration Guide* and the *Multidomain MDM Data Steward Guide*.

history table

A type of table in an ORS that contains historical information about changes to an associated table. History tables provide detailed change-tracking options, including merge and unmerge history, history of the pre-cleansed data, history of the base object, and history of the cross-reference.

Hub Console

Informatica MDM Hub user interface that comprises a set of tools for administrators and data stewards. Each tool allows users to perform a specific action, or a set of related actions, such as building the data model, running batch jobs, configuring the data flow, running batch jobs, configuring external application access to Informatica MDM Hub resources, and other system configuration and operation tasks.

hub object

A generic term for various types of objects defined in the Hub that contain information about your business entities. Some examples include: base objects, cross reference tables, and any object in the hub that you can associate with reporting metrics.

Hub Server

A run-time component in the middle tier (application server) used for core and common services, including access, security, and session management.

Hub Store

In a Informatica MDM Hub implementation, the database that contains the Master Database and one or more Operational Reference Store (ORS) database.

IDD Application

The main configuration and deployment unit for IDD implementation. An IDD application is what business users see when they launch and log in to IDD.

IDD Configuration Manager

A web-based utility used to add, modify and manage IDD applications.

In-flight data

In-flight data is business data that can go through different states (ACTIVE, PENDING, or DELETED) while progressing through a workflow.

lineage

Which systems, and which records from those systems, contributed to consolidated records in the Hub Store.

mapping

Defines a set of transformations that are applied to source data. Mappings are used during the stage process (or using the SiperianClient CleansePut API request) to transfer data from a landing table to a staging table. A mapping identifies the source column in the landing table and the target column to populate in the staging table, along with any intermediate cleanse functions used to clean the data.

Master data

A collection of common, core entities—along with their attributes and their values—that are considered critical to a company's business, and that are required for use in two or more systems or business processes. Examples of master data include customer, product, employee, supplier, and location data.

Master Database

Database that contains the Informatica MDM Hub environment configuration settings—user accounts, security configuration, ORS registry, message queue settings, and so on. A given Informatica MDM Hub environment can have only one Master Database. The default name of the Master Database is CMX_SYSTEM.

match

The process of determining whether two records should be automatically merged or should be candidates for manual merge because the two records have identical or similar values in the specified columns.

match column

A column that is used in a match rule for comparison purposes. Each match column is based on one or more columns from the base object.

match key

Encoded strings that represent the data in the fuzzy match key column of the base object. Match keys consist of fixed-length, compressed, and encoded values built from a combination of the words and numbers in a name or address such that relevant variations have the same match key value. Match keys are one part of the match tokens that are generated during the tokenize process, stored in the match key table, and then used during the match process to identify candidates for matching.

Match Path

Allows you to traverse the hierarchy between records—whether that hierarchy exists between base objects (intertable paths) or within a single base object (intra-table paths). Match paths are used for configuring match column rules involving related records in either separate tables or in the same table.

match rule

Defines the criteria by which Informatica MDM Hub determines whether records might be duplicates. Match columns are combined into match rules to determine the conditions under which two records are regarded as being similar enough to merge. Each match rule tells Informatica MDM Hub the combination of match columns it needs to examine for points of similarity.

match rule set

A logical collection of match rules that allow users to execute different sets of rules at different stages in the match process. Match rule sets include a search level that dictates the search strategy, any number of automatic and manual match rules, and optionally, a filter that allows you to selectively include or exclude records during the match process Match rules sets are used to execute to match column rules but not primary key match rules.

match table

Type of system table, associated with a base object, that supports the match process. During the execution of a Match job for a base object, Informatica MDM Hub populates its associated match table with the ROWID_OBJECT values for each pair of matched records, as well as the identifier for the match rule that resulted in the match, and an automerge indicator.

match type

Each match column has a match type that determines how the match column will be tokenized in preparation for the match comparison.

metadata

Data that is used to describe other data. In Informatica MDM Hub, metadata is used to describe the schema (data model) that is used in your Informatica MDM Hub implementation, along with related configuration settings.

null value

The absence of a value in a column of a record. Null is not the same as blank or zero.

Operational Reference Store (ORS)

Is a database schema that stores the rules for processing the master data, the rules for managing the set of master data objects, along with the processing rules and auxiliary logic used by the Informatica MDM Hub in defining the best version of the truth (BVT).

parentReference

A parentReference can be defined in the XML for the column that is the foreign key to the child record This defines a label to be displayed in the grandchild record that contains data from the child, to help users understand the relationship of the grandchildren to the child

process

See *business process* on page 167.

relationship base object

A relationship base object is a base object used to store information about Hierarchy Manager relationships.

resource group

A collection of secure resources that simplify privilege assignment, allowing you to assign privileges to multiple resources at once, such as easily assigning resource groups to a role.

Resource Kit

The Informatica MDM Hub Resource Kit is a set of utilities, examples, and libraries that provide examples of Informatica MDM Hub functionality that can be expanded on and implemented.

schema

The data model that is used in a customer's Informatica MDM Hub implementation. Informatica MDM Hub does not impose or require any particular schema. The schema is independent of the source systems.

Schema Manager

The Schema Manager is a design-time component in the Hub Console used to define the schema, as well as define the staging and landing tables. The Schema Manager is also used to define rules for match and merge, validation, and message queues.

Security Access Manager (SAM)

Security Access Manager (SAM) is Informatica MDM Hub's comprehensive security framework for protecting Informatica MDM Hub resources from unauthorized access. At run-time, SAM enforces your organization's security policy decisions for your Informatica MDM Hub implementation, handling user authentication and access authorization according to your security configuration.

Security filter

Security filter specifies a condition that IDD applies to restrict and secure subject area data that individual users can access. Filters can be defined on primary object column, child column and grandchild column. You can set up any number of filters for a subject area.

Services Integration Framework (SIF)

The part of Informatica MDM Hub that interfaces with client programs. Logically, it serves as a middle tier in the client/server model. It enables you to implement the request/response interactions using any of the following architectural variations:

- Loosely coupled Web services using the SOAP protocol.
- Tightly coupled Java remote procedure calls based on Enterprise JavaBeans (EJBs) or XML.
- Asynchronous Java Message Service (JMS)-based messages.
- XML documents going back and forth via Hypertext Transfer Protocol (HTTP).

sibling reference

A sibling reference is a relationship from one record in a subject area to a child record within that subject area. For example, a customer could include both address and phone number child records, with the phone number having a foreign key to associate it with a specific address.

state management

The process for managing the system state of base object and cross-reference records to affect the processing logic throughout the data flow. You can assign a system state to base object and cross-reference records at various stages of the data flow using the Hub tools that work with records. In addition, you can

use the various Hub tools for managing your schema to enable state management for a base object, or to set user permissions for controlling who can change the state of a record.

State management is limited to the following states: ACTIVE, PENDING, and DELETED.

stored procedure

A named set of Structured Query Language (SQL) statements that are compiled and stored on the database server. Informatica MDM Hub batch jobs are encoded in stored procedures so that they can be run using job execution scripts in job scheduling software (such as Tivoli or CA Unicenter).

Subject area

The core organizing concept for an IDD application. A subject area represents a collection of data that should be treated as a unit from a business perspective.

Subject area group

A set of one or more subject areas that have the same base object at their root (also called the primary object).

Subject Area Relationships

Subject Area Relationships defines how subject areas are related to each other. Subject area can have child subject areas, grandchild areas, and sibling references.

system state

Describes how base object records are supported by Informatica MDM Hub. The following states are supported: ACTIVE, PENDING, and DELETED.

Trust

Trust is a mechanism for measuring the confidence factor associated with each cell based on its source system, change history and other business rules. Trust takes into account the age of data, how much its reliability has decayed over time, and the validity of the data.

unmerge

Process of unmerging previously-merged records. For merge-style base objects only.

user exit

User exits provide a means to add custom business logic to standard IDD operations.

user group

A logical collection of user accounts.

validation process

Process of verifying the completeness and integrity of the metadata that describes a repository. The validation process compares the logical model of a repository with its physical schema. If any issues arise, the Repository Manager generates a list of issues requiring attention.

workflow

In Informatica Multidomain MDM, a workflow represents a business process within an organization. See *business process* on page 167.

INDEX

I

L

M

O

P

R

S

T

U

V