Informatica® MDM Registry Edition
10.1

Security Framework Guide

Informatica MDM Registry Edition Security Framework Guide
10.1
June 2018

Publication Date: 2018-07-04

Table of Contents

Preface

The Security Framework guide provides information on the security framework implemented with MDM Registry (MDM-RE).

This guide is written for use by system administrators who configures security and developers who are responsible to perform the security implementation. This guide assumes the user to have an understanding of security standards, protocols, and MDM-RE.

Learning About Informatica MDM Registry

This section provides details of documentation available with the Informatica MDM Registry product.

Introduction Guide

Introduces MDM Registry product and it's related terminology. It may be read by anyone with no prior knowledge of the product who requires a general overview of MDM Registry.

Installation Guide

This manual is intended to be the first technical material a new user reads before installing the MDM Registry software, regardless of the platform or environment.

Design Guide

This is a guide that describes the steps needed to design, define and load an MDM Registry "System".

Developer Guide

This manual describes how to develop a custom search client application using the MDM - Registry Edition API.

Operations Guide

This manual describes the operation of the run-time components of MDM - Registry Edition, such as servers, search clients and other utilties.

Populations and Controls Guide

This manual describes SSA-Name3 populations and the controls they support. The latter are added to the Controls statement used within an IDX-Definition or Search-Definition section of the SDF.

Security Framework Guide

This manual describes how to implement security in the MDM-RE product.

Release Notes

The Release Notes contain information about what's new in this version of MDM - Registry Edition. It is also summarizes any documentation updates as they are published.

What Do I Read If. . .

I am. . .

. . . a business manager

The INTRODUCTION to MDM- Registry Edition will address questions such as "Why have we got MDM - Registry Edition?", "What does MDM - Registry Edition do"?

I am. . .

. . . installing the product?

Before attempting to install MDM-RE, you should read the INSTALLATION GUIDE to learn about the prerequisites and to help you plan the installation and implementation of the MDM-Registry Edition.

I am. . .

...an Analyst or Application Programmer?

A high-level overview is provided specifically for Application Programmers in the INTRODUCTION to MDM Registry Edition.

When designing and developing the application programs, refer to the DEVELOPER GUIDE which describes a typical application process flow and API parameters. Working example programs that illustrate the calls to MDM-RE in various languages are available under the `<MDM-RE_client_installation>/samples` directory.

I am. . .

...designing and administering Systems?

The process of designing, defining and creating Systems is described in the DESIGN GUIDE. Administering the servers and utilities is described in the OPERATIONS manual.

Informatica Resources

Informatica Network

Informatica Network hosts Informatica Global Customer Support, the Informatica Knowledge Base, and other product resources. To access Informatica Network, visit https://network.informatica.com.

As a member, you can:

- Access all of your Informatica resources in one place.
- Search the Knowledge Base for product resources, including documentation, FAQs, and best practices.
- View product availability information.
- 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 search Informatica Network for product resources such as documentation, how-to articles, best practices, and PAMs.

To access the Knowledge Base, visit https://kb.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

To get the latest documentation for your product, browse the Informatica Knowledge Base at https://kb.informatica.com/_layouts/ProductDocumentation/Page/ProductDocumentSearch.aspx.

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

Informatica Product Availability Matrixes

Product Availability Matrixes (PAMs) indicate the versions of operating systems, databases, and other types of data sources and targets that a product release supports. If you are an Informatica Network member, you can access 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. Developed from the real-world experience of hundreds of data management projects, Informatica Velocity represents the collective knowledge of our consultants who have worked with organizations from around the world to plan, develop, deploy, and maintain successful data management solutions.

If you are an Informatica Network member, you can access 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 augment, extend, or enhance your Informatica implementations. By leveraging any of the hundreds of solutions from Informatica developers

and partners, you can improve your productivity and speed up time to implementation on your projects. You can access Informatica Marketplace at https://marketplace.informatica.com.

Informatica Global Customer Support

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

To find your local Informatica Global Customer Support telephone number, visit the Informatica website at the following link:
http://www.informatica.com/us/services-and-training/support-services/global-support-centers.

If you are an Informatica Network member, you can use Online Support at http://network.informatica.com.

CHAPTER 1

Introduction to MDM Registry Security Framework

This chapter includes the following topics:

Overview

MDM Registry Edition (MDM-RE) is a software to add search and matching applications to identity data stored in databases and Data Warehouses.

MDM Registry Security Framework is a Java security framework that performs authentication, authorization, cryptography, session management, and provisioning. The framework hides implementation complexities and it exposes a simple API interface that can save developer efforts in securing any application. MDM-RE security provides role-based authentication and authorization on users.

The MDM-RE Security Framework guide describes features of framework, architecture, configuration, and how to configure the framework.

Features

The MRBS framework provides the following features:

Authentication

Authentication is the process of verifying the identity of a user to ensure that the user is valid. A *user* is an *individual* who wants to access the Informatica MDM-RE resources.

Authorization

Authorization is the process of determining whether a user has sufficient privileges to access the requested resource.

Provisioning

Provisioning is a process of implementing and managing the security policies for an application. Provisioning includes the following:

1. Create/Delete Users

2. Create User roles and User groups
3. System resources that needs secure access
4. Privileges and Permissions

To create the security policies to access the system resources, Provisioning creates permissions using privileges and assigning roles to permissions and roles to users.

Cryptography

Supports crypto algorithms for securing user credentials.

Cache Manager

The MRBS framework uses cache for better performance.

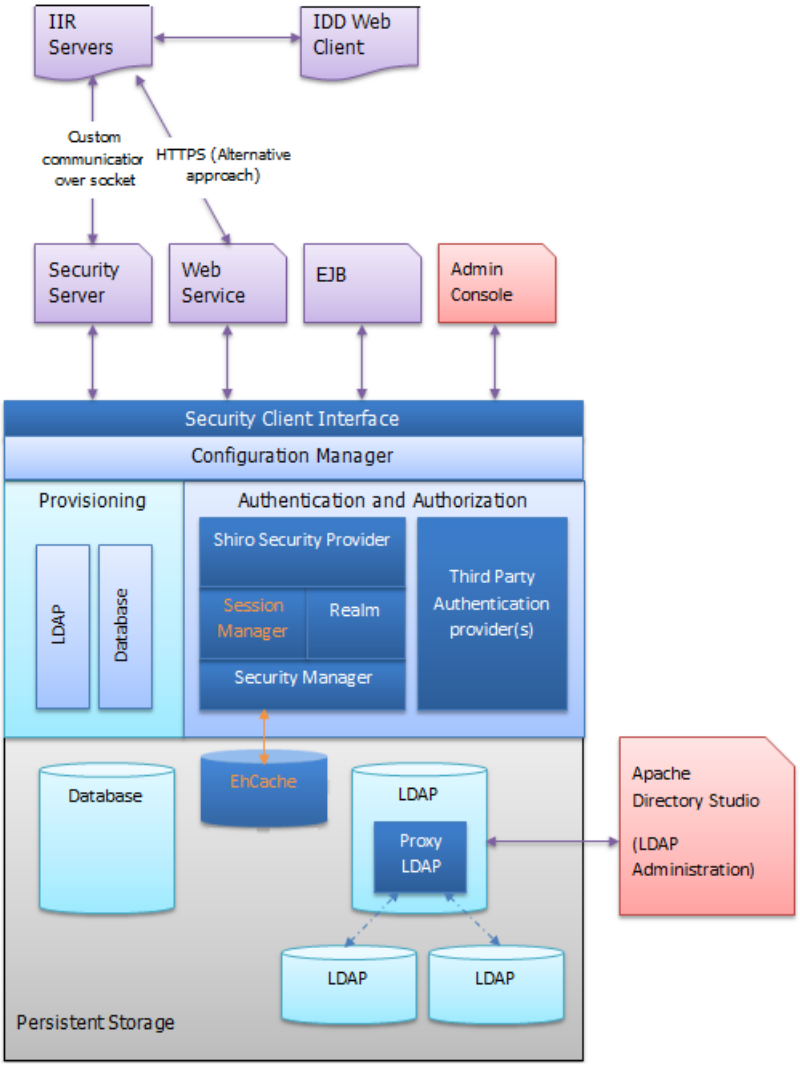
Back-end support

Database or a directory service such as LDAP (Light-Weight Directory Access Protocol) stores the application specific data such as Users, Resources, Privileges, and Permissions.

Architecture

The security framework is a generic framework that enables users to perform authentication and authorization on top of a new security provider or an existing security provider. The framework hides implementation complexities and exposes a simple API interface that can save developer's efforts in securing any application.

The security framework consists of the Security Client Interface and Configuration Manager. The following figure shows the security framework architecture:



Security Client Interface

The main client interface that exposes a set of APIs for client applications. The client application uses these APIs to build a security around its data model. The framework requires the user to configure the security provider using appropriate Configuration provider.

Configuration Manager

The security framework can work with different security providers. At present, the MDM-RE Security Framework uses Apache Shiro. If required, MDM-RE can generalize to work with other third party frameworks in the future.

The framework also supports different persistent storages such as databases and LDAP directory service.

Using the Configuration Manager, you can configure the following in LDAP or Database store:

1. Security provider: Properties of the security provider
2. A persistence storage: Properties such as server address, and root user login details of the persistent storage.

The framework provides `IConfigProviderFactory` and `IConfigProvider` interfaces to enable users to configure the Security Provider that is running in the backend (may be LDAP or a database).

Authentication and Authorization

The core module which performs the authentication of the user. After authenticating a user the user session stores the authentication information and generates a session token. The rest of the communication that is authorization from the client to framework uses this session token to validate the user.

For every user who has logged-in, the framework generates the authorization information by querying the persistence storage. The framework caches this authorization information into memory for boosting the performance.

Provisioning

Provisioning manages the security policies for the following:

1. Creation or Deletion of users
2. User roles
3. Groups
4. System resources that needs secure access
5. Privileges and Permissions

The database or the directory services stores provisioning information.

Persistent Storage

Persistent storage stores all the metadata of security framework. At present, the framework supports database and LDAP but is easily extensible for another storage if required.

Related Topics:

- "Security Client Interface APIs" on page 28

Informatica MDM-RE Security Concepts

Security is the ability to protect information privacy, confidentiality, and data integrity by guarding against unauthorized access. Before setting up security for Informatica MDM-RE implementation, it is important for you to understand some key concepts.

Authentication

Authentication is the process of verifying the identity of a user to ensure that the user is a valid user.

A user is an individual who wants to access Informatica MDM-RE. MDM-RE authenticates based on the supplied credentials, user name / password.

Authorization

Authorization is the process of determining whether a user has sufficient privileges to access a requested Informatica MDM-RE resource.

CHAPTER 2

Configuration Tasks

This chapter includes the following topics:

Overview

The security framework requires certain configuration of the database and LDAP. After configuring the framework, MDM Registry Edition uses it for authentication and authorization of users.

Before you install MDM Registry Edition, perform the following configuration tasks for the security framework:

1. Configure database.
2. Configure and setup LDAP.
3. Create users.

For information about installing MDM Registry Edition, see the *MDM Registry Edition Installation Guide*.

Configuring Database

MDM Registry Edition supports Oracle, Microsoft SQL Server, and IBM DB2 UDB databases. Configure the security framework to use the security information from the database or use the following default metadata, which is required to configure the framework appropriately.

You can find the scripts to create the required tables for different databases in the `$SSATOP/security/scripts` directory.

Users

Column	Type	Description
REC_NO	INTEGER	The record index and is the primary key.
ID	VARCHAR(32)	User identifier. It must be unique.
FIRST_NAME	VARCHAR(64)	First name of the user.
LAST_NAME	VARCHAR(64)	Last name of the user.
EMAIL	VARCHAR(255)	Email address of the user.
PASSWORD	VARCHAR(512)	User's password.

Roles

Column	Type	Description
REC_NO	INTEGER	The record index and is the primary key.
ID	VARCHAR(32)	Role identifier. It must be unique.
DESCRIPTION	VARCHAR(255)	Description about the role.

Resources

Column	Type	Description
REC_NO	INTEGER	The record index and is the primary key.
ID	VARCHAR(32)	Resource identifier. It must be unique.
DESCRIPTION	VARCHAR(255)	Description about the resources.

Privileges

Column	Type	Description
REC_NO	INTEGER	The record index and is the primary key.
ID	VARCHAR(32)	Privilege identifier. It must be unique.
DESCRIPTION	VARCHAR(255)	Description about the privilege.

Groups

Column	Type	Description
REC_NO	INTEGER	The record index and is the primary key.
ID	VARCHAR(32)	Group identifier. It must be unique.
DESCRIPTION	VARCHAR(255)	Description about the group.

Permissions

Column	Type	Description
REC_NO	INTEGER	The record index and is the primary key.
ID	VARCHAR(32)	Permission identifier. It must be unique.
RESOURCE_ID	INTEGER	Reference to resource record.
DESCRIPTION	VARCHAR(255)	Description about the permission.

Privileges of permission

Column	Type	Description
PERMISSION_ID	INTEGER	Reference to permission record.
PRIVILEGE_ID	INTEGER	Reference to privilege record.

User to role

Column	Type	Description
USER_ID	INTEGER	Reference to user record.
ROLE_ID	INTEGER	Reference to role record.

User to group

Column	Type	Description
USER_ID	INTEGER	Reference to user record.
GROUP_ID	INTEGER	Reference to group record.

Roles to permissions

Column	Type	Description
ROLE_ID	INTEGER	Reference to role record.
PERMISSION_ID	INTEGER	Reference to permission record.

To create schema, use the following commands:

For Oracle:

```
sqlplus <user>/<password>@service @secora.sql
```

For Microsoft SQL Server:

```
sqlcmd -U <User Id> -P <Password> -S <Server> -i secmsq.sql
```

For IBM DB2 UDB:

- Edit the `secudb.sql` file and update the line

```
CONNECT TO <service>USER <user> USING "<password>";
```

```
db2 -tf secudb.sql
```

Download and Install JDBC Library

The Security Framework requires JDBC libraries to connect to the database. You must download and install the appropriate JDBC libraries from your database vendor.

1. Based on the database, download the required libraries.

   The following table lists the required libraries for each supported database:

Database	Library Names
DB2	db2jcc.jar
Oracle	ojdbc5.jar Orai18n.jar dms.jar
Microsoft SQL	sqljdbc4 .jar

2. Copy the library files to the following directory:
   - On Windows. `<MDM Registry Edition Installation Directory>\security\lib`
   - On UNIX. `<MDM Registry Edition Installation Directory>/security/lib`
3. If you use Oracle, copy the `dms.jar` file to the following directory:
   - On Windows. `<MDM Registry Edition Installation Directory>\tomcat\lib`
   - On UNIX. `<MDM Registry Edition Installation Directory>/tomcat/lib`

Configuring LDAP

See the Configuring LDAP section.

Related Topics:

- “Configuring LDAP” on page 19

CHAPTER 3

Setting Up LDAP

This chapter includes the following topics:

Overview

Use LDAP (Lightweight Directory Access Protocol), Active Directory, or a database to store the application-specific data such as users, resources, privileges, and permissions. Active Directory is a directory service that Microsoft developed for Windows domain network. Active Directory uses LDAP as the core directory protocol.

Setting Up LDAP

To set up a LDAP server on Linux machine:

1. You need to install Berkeley Database Version 11 gR2 (11.2.5.1.29) as one of the pre-requisite.

   To download this software, go to http://download.oracle.com/berkeley-db/db-5.1.29.tar.gz .

   See the installation instructions from http://docs.oracle.com/cd/E17076_02/html/installation/build_unix.html

   Note: The latest version of Berkeley Database does not work with LDAP.

2. Before installing LDAP, set up the following environment variables:

```
$ LDFLAGS="-L/usr/local/lib -L/usr/local/BerkeleyDB.5.1/lib  -R/usr/local/BerkeleyDB.
5.1/lib"
$ export LDFLAGS

$ CPPFLAGS="-I/usr/local/BerkeleyDB.5.1/include/"
```

```
$ export CPPFLAGS

$ LIBS="-ldb -lgcc_s"
$ export LIBS

$ LD_LIBRARY_PATH=/usr/local/BerkeleyDB.5.1/lib
$ export LIBRARY_PATH
```

a. Ensure that the path for Berkeley DB.5.1 lib and include folder is correct.

```
$ configure --enable-ldap
$ make depend
$ su root -c 'make install'
```

3. To install LDAP Version (2.4.23 (20100719)), download it from ftp://ftp.openldap.org/pub/OpenLDAP/openldap-stable/openldap-stable-20100719.tgz

Configuring LDAP

To configure LDAP:

1. Edit the default `slapd.conf` file installed as `/usr/local/etc/openldap/slapd.conf` on your computer.

   To edit this file, see the latest copy of the `slapd.conf` file from the project repository, `$SSATOP/security/scripts/slapd.conf`

2. If the following schema files are not available, you need to add these files:

```
include         /usr/local/etc/openldap/schema/core.schema
include         /usr/local/etc/openldap/schema/cosine.schema
include         /usr/local/etc/openldap/schema/inetorgperson.schema
```

3. Enter the DBD database definitions as follows:

```
database        bdb
suffix          "dc=informatica,dc=com"

## Give Admins immediate write access:
access to dn.subtree="dc=informatica,dc=com"
       by group/organizationalRole/roleOccupant=
       "cn=Administrators,ou=Groups,dc=informatica,dc=com" write
       by * none break

## This rule is needed by authz-regexp
## (Note: Since uid is used in DN, user cannot change its own uid.)
access to attrs=uid
       by anonymous read
       by users read

## Grant access to passwords for auth, but allow users to change
## their own.
access to attrs=userPassword
       by anonymous auth
       by self write

## The default rule: Allow DNs to modify their own records. Give
## read access to everyone else.
access to *
       by self write
       by users read

rootdn          "cn=Manager,dc=informatica,dc=com"
```

```
rootpw          secret
directory       /usr/local/var/informatica-data

# Indices to maintain
index   objectClass     eq
Save and close slapd.conf file.
```

Configuring LDAP Proxy server

If there is any other remote LDAP sever which hosts user information, you need to define it as a proxy server.

1. To define LDAP as a proxy server, edit the `slapd.conf` file as follows:

```
database ldap
uri "ldap://10.72.40.173:389"
suffix "dc=xyz,dc=com"
#idassert-authzFrom dn.subtree="ou=users,dc=xyz,dc=com"
```

2. Create the directories for database files as:

```
/usr/local/var/openldap-data
/usr/local/var/informatica-data
```

Starting and Stopping LDAP

You can start and stop the LDAP server using commands.

1. To start the LDAP server, use the command:

```
$ su root -c /usr/local/libexec/slapd
```

2. To stop the LDAP server, use the command:

```
$ kill `pgrep slapd`
```

Loading Schema

1. To load schema, use the command:

```
ldapadd -x -D "cn=Manager, dc=informatica,dc=com " -W -f$SSATOP/security/scripts/
ldapstore.ldif
```

This script will create the default schema required for the security system and will create default administrator user with user id as “admin” and default password as “password”.

2. From the LDAP browser utility, add new users using the following command:

```
secuser -t<DB|LDAP> -x<config file> -a(add)|-d(delete) -u<Admin User Id> -p<Admin
password> -i<User ID> -f<First Name> -l<Last Name> -w<Password>
```

```
secuser -tLDAP -x$SSATOP/security/SecConfig.xml -a -uadmin -ppassword -ijsmith -fJohn -
lSmith -wmypassword
```

Setting Up LDAP Browser

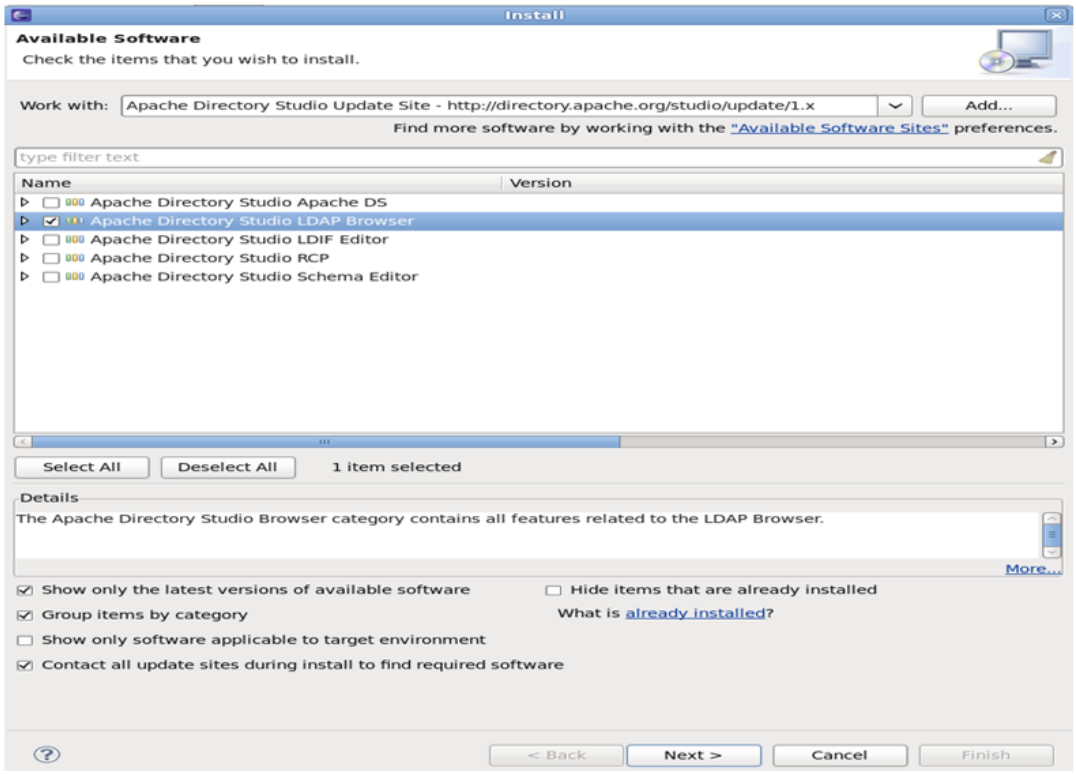
LDAP browser is an Eclipse plug-in of Apache Directory Studio. This tool helps to read and display the tree of LDAP Server. Use this tool to create, edit, and delete entries in the tree.

Installing LDAP Browser

To install LDAP browser:

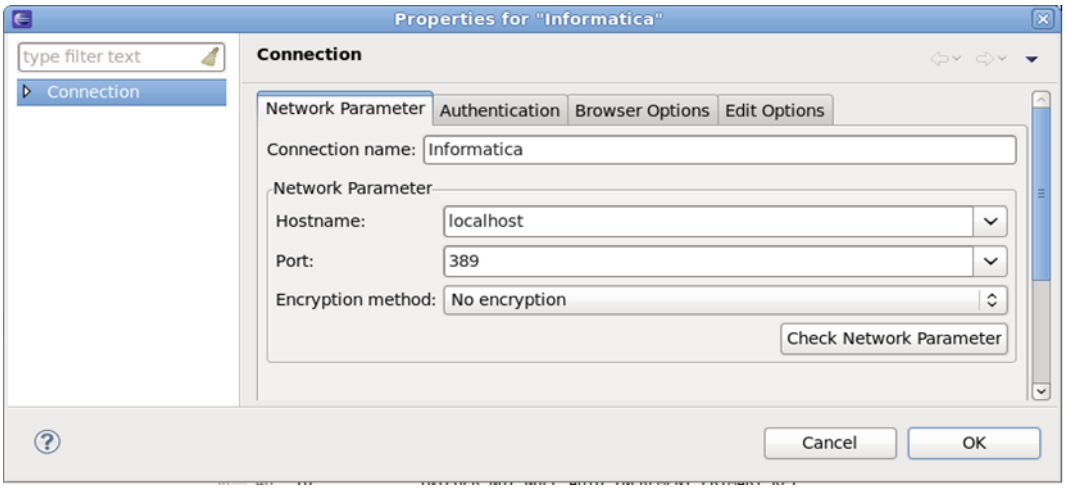
1. In Eclipse, go to **Help** > **Install new software**
2. Enter following URL in **Work with:** as shown:

   http://directory.apache.org/studio/update/1.x

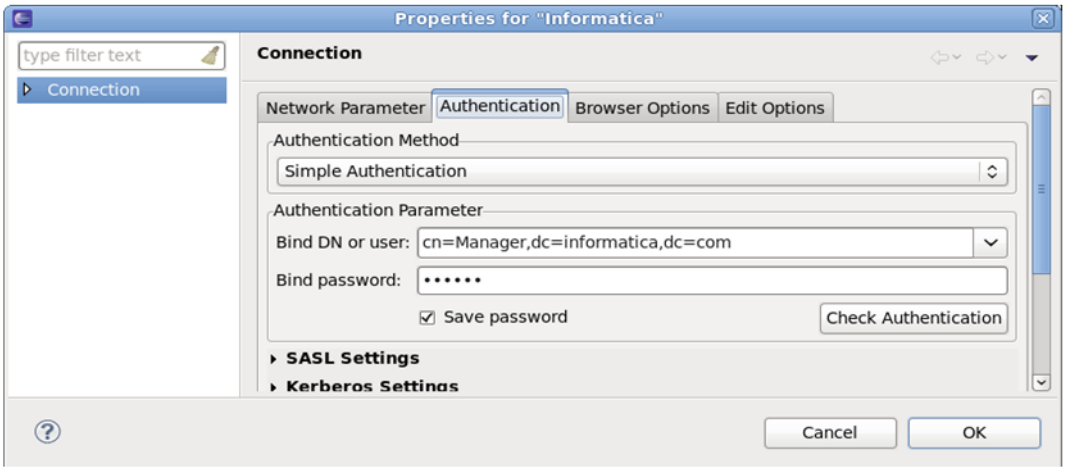


3. Select the LDAP browser and install the plug-in.
4. Open LDAP Perspective: In Eclipse go to **window** > **Open Perspective**and select **other**.
5. Create a connection for **informatica.com**.

6. Select the **Network Parameter** tab and enter the values for the fields as follows:

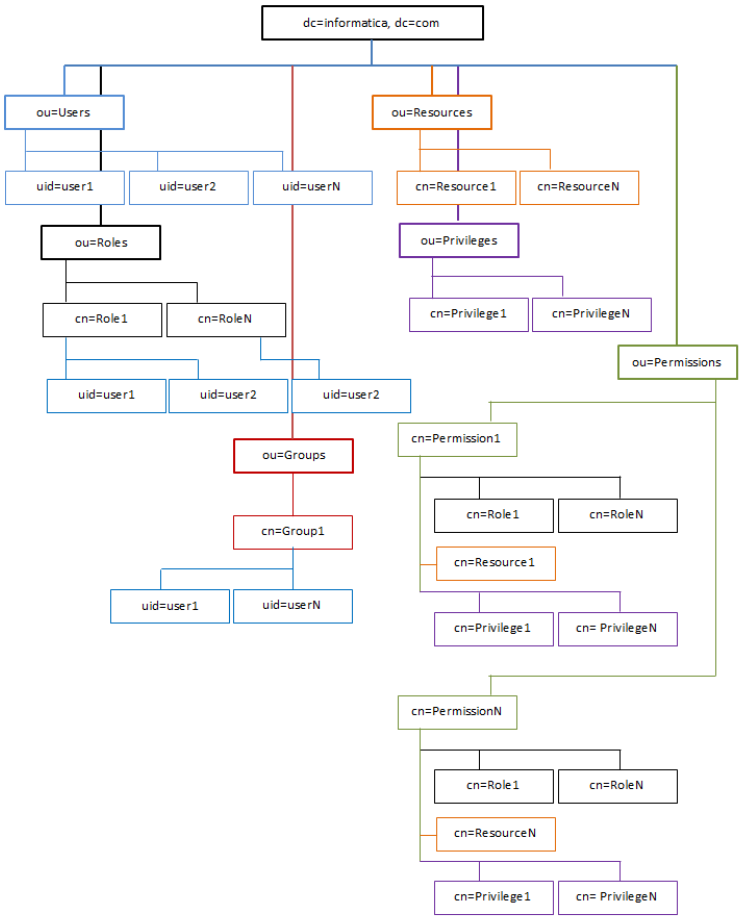


7. Select the **Authentication** tab and enter the values for the fields as follows:



Directory Tree Structure for LDAP

Directory structure depicts the hierarchy of the organization. The following provides the directory tree structure for LDAP:



The following object classes for entities are provided:

Root DN Record for organization

DN: dc=informatica,dc=com	
Attribute	**Value**
objectClass	dcObject
dc	organization (structural)
o	informatica
description	Informatica.com

DN for Users

DN: ou=Users,dc=informatica,dc=com	
Attribute	**Value**
objectClass	organizationalUnit

ou	Users
description	Users of informatica.com

DN for User

DN: uid=xxxxx,ou=Users,dc=informatica,dc=com	
Attribute	**Value**
objectClass	inetOrgPerson
objectClass	organizationalPerson
objectClass	Person
cn	Common name
uid	Unique ID of user (xxxx)
userPassword	User password

DN for Roles

DN: ou=Users,dc=informatica,dc=com	
Attribute	**Value**
objectClass	organizationalUnit
ou	Roles
description	Roles for the informatica.com

DN for Role

DN: ou=Roles,dc=informatica,dc=com	
Attribute	**Value**
objectClass	organizationalRole
objectClass	top
cn	Name of a role
roleOccupant	uid=xxxx,ou=Users,dc=informatica,dc=com
roleOccupant	uid=yyyy,ou=Users,dc=informatica,dc=com

DN for Resources

DN: ou=Resources,dc=informatica,dc=com	
Attribute	**Value**
objectClass	organizationalUnit
ou	Resources
description	Resources of informatica.com

DN for Resource

DN: cn=RESOURCE_NAME,ou=Resources,dc=informatica,dc=com	
Attribute	**Value**
objectClass	organizationalRole
objectClass	top
cn	Name of a resource
Description	Description about the resource

DN for Privileges

DN: ou= Privileges,dc=informatica,dc=com	
Attribute	**Value**
objectClass	organizationalUnit
ou	Privileges
Description	Access privileges for the system resources

DN for Privilege

DN: cn=PRIVILEGE_NAME,ou=Privileges,dc=informatica,dc=com	
Attribute	**Value**
objectClass	organizationalRole
objectClass	top
cn	Name of a privilege
Description	Description about the privilege

DN for Groups

DN: ou= Groups,dc=informatica,dc=com	
Attribute	**Value**
objectClass	organizationalUnit
ou	Groups
Description	Groups for the informatica.com

DN for Group

DN: cn=GROUP_NAME,ou=Groups,dc=informatica,dc=com	
Attribute	**Value**
objectClass	organizationalRole
objectClass	top
cn	Name of a group
Description	Description about the group
roleOccupant	Users of the group

The ACL (access control list) uses the group entities to add restrictions on users. To provide admin users a write access to all, define the `slapd.conf` file as:

```
## Give Admins immediate write access:
access to dn.subtree="dc=informatica,dc=com"
by group/organizationalRole/roleOccupant=
"cn=Administrators,ou=Groups,dc=informatica,dc=com" write
       by * none break
```

DN for Permissions

DN: ou= Permissions,dc=informatica,dc=com	
Attribute	**Value**
objectClass	organizationalUnit
ou	Permissions
Description	Permissions for the informatica.com

DN for Permission

DN: cn=PERMISSION_NAME,ou=Permissions,dc=informatica,dc=com	
Attribute	**Value**
objectClass	organizationalRole
objectClass	top
cn	Name of a permission
Description	Description about the permission
roleOccupant	Role entity cn=Admin,ou=Roles,dc=informatica,dc=com
roleOccupant	Resource entity cn=IDD_APPLICATION,ou=Resources,dc=informatica,dc=com
roleOccupant	Privilege entity cn=CREATE,ou=Privileges,dc=informatica,dc=com

Import Configuration into LDAP

MDM-RE provides `ldif` files that can be imported into an existing LDAP store. The sample `ldif` scripts is found at the location: `$SSATOP/security/scripts` directory.

To import the sample `ldif` into an LDAP store, run the following command:

```
    ldapadd -x -D "cn=Manager, dc=informatica,dc=com " -W -f$SSATOP/security/scripts/
ldapstore.ldif
```

CHAPTER 4

Security Implementation

This chapter includes the following topics:

Overview

The security framework requires configuration parameters to enable security in MDM Registry Edition. The Security Client Interface exposes a list of security APIs for a developer to implement security.

To configure the security framework for MDM Registry Edition, specify the configuration parameters in a security configuration file, create users, and configure the MDM Registry Edition Security Server.

You can create a security configuration file by using the security APIs or the sample configuration files that MDM Registry Edition provides. A security configuration file contains information about the security provider, which can be LDAP or a database. MDM Registry Edition Security Server uses the security configuration file when the server starts.

Security Client Interface APIs

The security framework exposes a set of APIs for a client application through the Security Client Interface, which is the main client interface to build security around the client application's data model. Use the APIs to perform the authentication, authorization, and provisioning tasks.

Note: MDM Registry Edition does not have a provisioning interface or an administration interface.

Authentication and Authorization APIs

The following table describes the authentication and authorization APIs that the Security Client Interface exposes:

API	Description	Return Type
login	`login (String userId, String password) throws SecurityClientException` Performs a login attempt for a user.	byte[]
logout	`logout (byte[] sessionId) throws SecurityClientException` Logs out a user and invalidates and removes the associated session.	void
isPermitted	`isPermitted (byte[] sessionToken, String resourceId, String privilegeId) throws SecurityClientException` Returns true if the user is permitted to perform the action specified by the provided privilege against the provided resource.	boolean
isPermittedAll	`isPermittedAll (byte[] sessionToken, String resourceId, String...privilegeId) throws SecurityClientException` Returns true if the user is permitted to perform the actions specified by the provided privileges against the provided resource.	boolean
isPermittedAll	`isPermittedAll (byte[] sessionToken, String resourceId, Set< String > privilegeIds) throws SecurityClientException` Returns true if the user is permitted to perform the actions specified by the provided privileges against the provided resource.	boolean
hasRole	`hasRole (byte[] sessionToken, String roleId) throws SecurityClientException` Returns true if the user has the provided role.	boolean

Provisioning APIs

The following table describes the provisioning APIs that the Security Client Interface exposes:

API	Description	Return Type
addOrUpdateResource	`addOrUpdateResource (byte[] sessionToken, String resourceId, Properties resourceProperties) throws SecurityClientException` Add a new resource to the resource collection or update an existing resource.	void
remove Resource	`remove Resource (byte[] sessionToken, String resourceId) throws SecurityClientException` Removes resources from the resource collection.	void
getAllResources	`getAllResources (byte[] sessionToken) throws SecurityClientException` Get all the resources IDs accessible from this session.	Set< String >
getResourceProperties	`getResourceProperties (byte[] sessionToken, String resourceId) throws SecurityClientException` Return the resource properties.	Properties
addOrUpdatePrivilege	`addOrUpdatePrivilege (byte[] sessionToken, String privilegeId, Properties privilegeProperties) throws SecurityClientException` Add a new privilege or update an existing one.	void
removePrivilege	`removePrivilege (byte[] sessionToken, String privilegeId) throws SecurityClientException` Remove the privilege from the privileges collection.	void
getAllPrivileges	`getAllPrivileges (byte[] sessionToken) throws SecurityClientException` Return a collection of privileges granted to the user on the specified resource.	Set< String >
getPrivilegeProperties	`getPrivilegeProperties (byte[] sessionToken, String privilegeId) throws SecurityClientException` Return the privilege properties.	Properties
addOrUpdateUser	`addOrUpdateUser (byte[] sessionToken, String userId, Properties userProperties) throws SecurityClientException` Add a new user or update an existing one.	void
removeUser	`removeUser (byte[] sessionToken, String userId) throws SecurityClientException` Remove the user from the users collection.	void

API	Description	Return Type
getAllUsers	`getAllUsers (byte[] sessionToken) throws SecurityClientException` Get all the user IDs accessible from this session.	Set< String >
getUserProperties	`getUserProperties (byte[] sessionToken, String userId) throws SecurityClientException` Return the user properties.	Properties
addOrUpdateRole	`addOrUpdateRole (byte[] sessionToken, String roleId, Properties roleProperties) throws SecurityClientException` Add a new role or update an existing one.	void
removeRole	`removeRole (byte[] sessionToken, String roleId) throws SecurityClientException` Remove the role from the roles collection.	void
getAllRoles	`getAllRoles (byte[] sessionToken) throws SecurityClientException` Get all the role IDs accessible from this session.	Set< String >
getRoleProperties	`getRoleProperties (byte[] sessionToken, String roleId) throws SecurityClientException` Return the role properties.	Properties
addOrUpdatePermission	`addOrUpdatePermission (byte[] sessionToken, String permissionId, String resourceId, Set< String > privilegeId, Properties permissionProperties) throws SecurityClientException` Add a new permission or update an existing one.	void
removePermission	`removePermission (byte[] sessionToken, String permissionId) throws SecurityClientException` Remove the permission from the permissions collection.	void
getAllPermission	`getAllPermissions (byte[] sessionToken) throws SecurityClientException` Get all the permission IDs accessible from this session.	Set< String >
getPermissionProperties	`getPermissionProperties (byte[] sessionToken, String permissionId) throws SecurityClientException` Return the permission properties.	Properties
getPermissionResourceId	`getPermissionResourceId (byte[] sessionToken, String permissionId) throws SecurityClientException` Return the permission resource ID.	String

API	Description	Return Type
getPermissionPrivilegesIds	`getPermissionPrivilegesIds (byte[] sessionToken, String permissionId) throws SecurityClientException` Return the permission privileges IDs.	Set< String >
getRolePermissionsIds	`getRolePermissionsIds (byte[] sessionToken, String roleId) throws SecurityClientException` Return the role permissions IDs.	Set< String >
getPermissionRoleIds	`getPermissionRoleIds (byte[] sessionToken, String permissionId) throws SecurityClientException` Return the role IDs of permission.	Set< String >
assignPermissionToRole	`assignPermissionToRole (byte[] sessionToken, String permissionId, String roleId) throws SecurityClientException` Assign permission to role.	void
revokePermissionFromRole	`revokePermissionFromRole (byte[] sessionToken, String permissionId, String roleId) throws SecurityClientException` Revoke permission from role.	void
assignRoleToUser	`assignRoleToUser (byte[] sessionToken, String roleId, String userId) throws SecurityClientException` Assign a role to a user.	void
revokeRoleFromUser	`revokeRoleFromUser (byte[] sessionToken, String roleId, String userId) throws SecurityClientException` Revoke a role from an user.	void
getAllUserRoles	`getAllUserRoles (byte[] sessionToken, String userId) throws SecurityClientException` Return a collection of roles IDs assign to user.	Set< String >
getAllUsersWithRole	`getAllUsersWithRole (byte[] sessionToken, String roleId) throws SecurityClientException` Return a collection of users IDs within role.	Set< String >
addOrUpdateUsersGroup	`addOrUpdateUsersGroup (byte[] sessionToken, String usersGroupId, Properties usersGroupProperties) throws SecurityClientException` Add a new users group or update an existing one.	void

API	Description	Return Type
removeUsersGroup	`removeUsersGroup (byte[] sessionToken, String usersGroupId) throws SecurityClientException` Remove the users group from the users group collection.	void
getAllUsersGroups	`getAllUsersGroups (byte[] sessionToken) throws SecurityClientException` Get all the users group IDs accessible from this session.	Set< String >
getUsersGroupProperties	`getUsersGroupProperties (byte[] sessionToken, String usersGroupId) throws SecurityClientException` Return the users group properties.	Properties
addUserToUserGroup	`addUserToUserGroup (byte[] sessionToken, String usersGroupId, String userId) throws SecurityClientException` Add a user to a users group.	void
removeUserFromUserGroup	`removeUserFromUserGroup (byte[] sessionToken, String usersGroupId, String userId) throws SecurityClientException` Remove a user from a users group.	void
getUsersFromUsersGroup	`getUsersFromUsersGroup (byte[] sessionToken, String usersGroupId) throws SecurityClientException` Get users IDs associated with the users group.	Set< String >
getUserGroupsForUser	`getUserGroupsForUser (byte[] sessionToken, String usersId) throws SecurityClientException` Get users groups IDs for the user.	Set< String >

Configuring the Security Configuration File

The security framework uses LDAP, Active Directory, or a database security provider. Before you use the security framework, you must specify the configuration parameters in a security configuration file. The security framework requires the security configuration file to implement security in MDM Registry Edition.

Configuring Security Framework by Using the Security APIs

Use the `IConfigProvider` and `IConfigProviderFactory` interfaces to programmatically configure the security framework.

If you use LDAP, use the following `IConfigProvider` and `IConfigProviderFactory` interfaces:

```
IConfigProvider<?> configProvider;
IConfigProviderFactory<?> configs = null;
Class<LdapConfigProvider> ldapConfigProviderClass =
```

```
com.informatica.rbs.config.LdapConfigProvider.class;
   ConfigProviderFactory.registerProduct(SecurityUser.class.toString(), "SHIRO",
ldapConfigProviderClass);
   configs = new ConfigProviderFactory<LDAP>(<CustomImplementation>.class.toString());
   configProvider = configs.getInstance();
   configProvider.initProvider();
```

If you use a database security provider, use the following `IConfigProvider` and `IConfigProviderFactory` interfaces:

```
   IConfigProvider<?> configProvider;
   IConfigProviderFactory<?> configs = null;
   Class<JdbcConfigProvider> jdbcConfigProviderClass =
com.informatica.rbs.config.JdbcConfigProvider.class;
   ConfigProviderFactory.registerProduct(SecurityUser.class.toString(), "SHIRO",
jdbcConfigProviderClass);
   configs = new ConfigProviderFactory<JDBC>(<CustomImplementation>.class.toString());
   configProvider = configs.getInstance();
   configProvider.initProvider();
```

Use the following APIs provided by the `IConfigProvider` interface:

- initProvider()
- verifyProvider()
- readConfig("<File Name>"). Reads an existing configuration XML file.
- readConfigEncrypted("<File Name>"). Reads an existing encrypted configuration file.
- writeConfig("<File Name>"). Writes a configuration file with the configured properties.

Note: To add or modify the security properties, you can use the other internal APIs that the security framework exposes.

Use the following APIs from the `IConfigProvider` interface to read the configuration parameters from a security configuration file or an encrypted dict file.

- `configProvider.readConfig("SecConfig.xml");  //Security configuration file`
- `configProvider.readConfigEncrypted("SecConfig.dic"); //Encrypted configuration file`

Configuring the Security Configuration File by Using a Sample File

A security configuration file contains the configuration parameters that the security framework requires to implement security in MDM Registry Edition. The configuration parameters vary based on the security provider, which can be a database, LDAP, or Active Directory.

You can use the sample configuration files that you can find in the following directory to create a configuration file:

- On Windows: `<MDM Registry Edition Installation Directory>\security\config`
- On UNIX: `<MDM Registry Edition Installation Directory>/security/config`

To configure the security configuration file, perform the following tasks:

1. Perform one of the following tasks:
   - If you use the database security provider, open the `SecConfigJdbc<DBType>.xml` file, where `DBType` can be one of the following values:
     - `Msq` for Microsoft SQL Server
     - `Ora` for Oracle

- `Udb` for IBM DB2 UDB

- If you use LDAP or Active Directory, open the `SecConfigLdap.xml` file.

2. Update the values of the configuration parameters, and save the file.

   Note: If you use Active Directory, ensure that you change the value of the `Subtype` parameter to PROVIDER_SUBTYPE_ADS.

3. Rename the file to `SecConfig.xml`.
4. Move the `SecConfig.xml` file to the following directory:
   - On Windows: `<MDM Registry Edition Installation Directory>\security`
   - On UNIX: `<MDM Registry Edition Installation Directory>/security`

After you configure the security configuration file, configure the MDM Registry Edition Security Server. For more information about configuring the MDM Registry Edition Security Server, see the *MDM Registry Edition Installation and Upgrade Guide*.

Configuration Parameters

Before you use the security framework, you must configure the configuration parameters for the security provider. The configuration parameters vary based on the security provider, which can be a database, LDAP, or Active Directory.

Configuration Parameters for LDAP

The following table describes all the configuration parameters that LDAP requires:

Parameter	Description
URL	The LDAP server address for authentication and authorization.
AUTH_MECHANISM	The authentication mechanism for LDAP server.
SYSTEM_ACCOUNT	System account for the LDAP server.
SYSTEM_PASSWORD	Password for system account.
HASH_ALGORITHM_TYPE	Hash algorithm used for storing the user passwords.
USER_DN	Distinguished name for user entity.
USER_ATTRIBUTE_ID	Unique attribute ID for the user.
USER_QUERY	LDAP query for retrieving a particular users.
USER_OBJECT_CLASSES	Object classes for the User entity.
ROLE_DN	Distinguished name for role entity.
ROLE_ATTRIBUTE_ID	Unique attribute ID for the role.
RESOURCE_DN	Distinguished name for resources entity.
RESOURCE_ATTRIBUTE_ID	Unique attribute ID for the resources.

Parameter	Description
PRIVILEGE_DN	Distinguished name for Resources entity.
PRIVILEGE_ATTRIBUTE_ID	Unique attribute ID for the privileges.
PERMISSION_DN	Distinguished name for permissions entity.
PERMISSION_ATTRIBUTE_ID	Unique attribute ID for the permissions.
USERS_GROUP_DN	Distinguished name for users group entity.
USERS_GROUP_ATTRIBUTE_ID	Unique attribute ID for the permissions.
CACHE_MANAGER_CONF_FILE	Path to the cache manager file. Cache is for performance and session storage.
ACTIVE_SESSION_CACHE_NAME	Name of the cache.
ENABLE_AUTORIZATION_CACHE	Enable or disable cache for authorization information.
SESSION_TIME_OUT	Timeout value for the session, after this time user session expires.
REMEMBER_ME	Remember user credentials.

Configuration Parameters for Active Directory

Before you configure the parameters for Active Directory, create two security groups named `IDD_ADMIN` and `IDD_APP_USER` in Active Directory. The `IDD_ADMIN` role has administrator privileges with which you can deploy an Informatica Data Director application. The `IDD_APP_USER` role has user privileges with which you can access a deployed Informatica Data Director application. You must assign the `IDD_ADMIN` and `IDD_APP_USER` roles to the appropriate users based on your requirement.

The following table describes all the configuration parameters that Active Directory requires:

Parameter	Description
URL	The Active Directory server address for authentication and authorization.
AUTH_MECHANISM	The authentication mechanism for the Active Directory server.
SYSTEM_ACCOUNT	System account for the Active Directory server.
SYSTEM_PASSWORD	Password for system account.
HASH_ALGORITHM_TYPE	Hash algorithm used for storing the user passwords.
USER_DN	Distinguished name for user entity.
USER_ATTRIBUTE_ID	Unique attribute ID for the user.
USER_QUERY	Active Directory query for retrieving a particular users.

Parameter	Description
USER_OBJECT_CLASSES	Object classes for the User entity.
ROLE_DN	Distinguished name for role entity.
ROLE_ATTRIBUTE_ID	Reserved for future use and do not remove the parameter.
RESOURCE_DN	Reserved for future use and do not remove the parameter.
RESOURCE_ATTRIBUTE_ID	Reserved for future use and do not remove the parameter.
PRIVILEGE_DN	Reserved for future use and do not remove the parameter.
PRIVILEGE_ATTRIBUTE_ID	Reserved for future use and do not remove the parameter.
PERMISSION_DN	Reserved for future use and do not remove the parameter.
PERMISSION_ATTRIBUTE_ID	Reserved for future use and do not remove the parameter.
USERS_GROUP_DN	Reserved for future use and do not remove the parameter.
USERS_GROUP_ATTRIBUTE_ID	Reserved for future use and do not remove the parameter.
CACHE_MANAGER_CONF_FILE	Path to the cache manager file. Cache is for performance and session storage.
ACTIVE_SESSION_CACHE_NAME	Name of the cache.
ENABLE_AUTORIZATION_CACHE	Enable or disable cache for authorization information.
SESSION_TIME_OUT	Timeout value for the session, after this time user session expires.
REMEMBER_ME	Remember user credentials.

ADS_GROUPS_TO_ROLES_MAP	Name of the map that defines the domain names for the `IDD_ADMIN` and `IDD_APP_USER` roles.
	You can use client tools, such as `ldp.exe` or an eclipse plug-in, to browse the Active Directory and view the domain names of the roles.
	For example:
	`<ProviderMapProperties>`
	`<Map name="ADS_GROUPS_TO_ROLES_MAP">`
	`<Property name="CN=IDD_ADMIN,CN=Builtin,DC=infatest,DC=local">IDD_ADMIN</Property>`
	`<Property name="CN=IDD_APP_USER,CN=Builtin,DC=infatest,DC=local">IDD_APP_USER</Property>`
	`</Map>`
	`</ProviderMapProperties>`

Configuration Parameters for a Database Security Provider

The following table describes all the configuration parameters that a database security provider requires:

Option	Description
DB_TYPE	Type of a database.
DB_SERVER_NAME	Address of database server.
DB_SERVER_PORT	Port address for the clients to connect the DB server.
DB_NETWORK_PROTOCOL	Network protocol for the DB connection, required for Oracle.
DB_DRIVER_TYPE	Driver type for the JDBC connection. For Oracle, the value is 'thin'.
DB_SERVICE_NAME	Service name, required when database is Oracle.
DB_USER	Database user which has admin rights of the database.
DB_PASSWORD	Password of the Database user.
DB_DATABASE_NAME	Name of the database.
HASH_ALGORITHM_TYPE	Hash algorithm used for storing the user passwords.
ENABLE_PERMISSION_LOOKUP	Enable or disable permission lookup.
AUTHENTICATION_QUERY	SQL query to perform authentication.
USER_ROLE_QUERY	SQL query to retrieve role for the particular user.
PERMISSION_QUERY	SQL query to retrieve permission details.
CACHE_MANAGER_CONF_FILE	Path to the Cache manager file. Cache is for performance and session storage.
ACTIVE_SESSION_CACHE_NAME	Name of the cache.
ENABLE_AUTORIZATION_CACHE	Enable or disable cache for authorization information.
SESSION_TIME_OUT	Timeout value for the session, after this time user session expires.
REMEMBER_ME	Remember user credentials.

Disabling Authorization

You can disable the authorization process based on your requirement. If you disable the authorization process, the security framework does not verify whether a user has sufficient privileges to access the requested resource. By default, the authorization process is enabled.

To disable the authorization process, configure the SSANOSECAUTHORIZATION environment variable to Yes and restart the MDM Registry Edition Server.

For example, `C:\InformaticaIR\bin>set SSANOSECAUTHORIZATION=Yes`

Note: If you use LDAP or a database security provider, you must disable the authorization process.

Encrypting the Security Configuration File

Use the secutil command to encrypt or decrypt a security configuration file for security purposes.

The secutil utility uses the following syntax:

```
secutil -i<Input File> -o<Output File> [-d]
```

The secutil command uses the following parameters:

-i<Input File>

Absolute path for the security configuration file if you want to encrypt it or the encrypted dict file if you want to decrypt it. You can find the security configuration file, `SecConfig.xml`, in the following directory:

- On Windows: `<MDM Registry Edition Installation Directory>\security`
- On UNIX: `<MDM Registry Edition Installation Directory>/security`

-o<Output File>

Absolute path for the dict file if you want to encrypt a configuration file or the configuration file if you want to decrypt a dict file.

-d

Instructs the secutil utility to decrypt the specified dict file.

MDM Registry Edition Security Server uses the encrypted dict file when it starts. After you create the encrypted dict file, add the `SSA_SEC_DICT` environment variable to the following file:

- On Windows: `<MDM Registry Edition Installation Directory>\env\mdmres.bat`
- On UNIX: `<MDM Registry Edition Installation Directory>/env/mdmres`

For example:

- On Windows, to encrypt the security configuration file, run the following commands:

```
set SSA_SEC_CONFIG=%SSATOP%\security\SecConfig.xml
set SSA_SEC_DICT=%SSATOP%\security\SecConfig.dic
secutil -i%SSA_SEC_CONIG% -o%SSA_SEC_DICT%
```

- On UNIX, to decrypt an encrypted dict file, run the following commands:

```
SSA_SEC_CONFIG="$SSATOP/security/SecConfig.xml"
SSA_SEC_DICT="$SSATOP/security/SecConfig.dic"
export SSA_SEC_DICT
$SSABIN/secutil -i$SSA_SEC_DICT -o$SSA_SEC_CONFIG -d
```

Provisioning Users

Use the secuser utility to add or delete users. The utility requires a configuration file for the security provider.

The secuser utility uses the following syntax:

```
secuser -t<LDAP|DB> -x<Configuration File Name> -a|-d -u<Administrator User Name>
-p<Administrator Password> -i<User Name> -f<First Name> -l<Last Name> -w<Password>
```

The secuser command uses the following parameters:

-t<LDAP|DB>

Specifies whether you use LDAP or database security provider. Use `-tLDAP` for LDAP and `-tDB` for database.

-x<Configuration File Name>

Absolute path of the security configuration file. You can find the security configuration file, `SecConfig.xml`, in the following directory:

- On Windows: `<MDM Registry Edition Installation Directory>\security`
- On UNIX: `<MDM Registry Edition Installation Directory>/security`

-a|-d

Specifies whether you want to add or delete a user. Use `-a` to add a user and `-d` to delete a user.

-u<Administrator User Name>

Name of the administrator user.

-p<Administrator Password>

Password for the administrator user.

-i<User Name>

Name of the user that you want to add or delete.

-f<First Name>

First name of the user.

-l<Last Name>

Last name of the user.

-w<Password>

Password for the user that you add.

For example:

- On Windows, to add a user, run the following command:

```
secuser -tLDAP -x%SSA_SEC_CONFIG% -a -uadmin -ppassword -iuserid -ffirstname
-llastname -wpassword
```

- On UNIX, to delete a user, run the following command:

```
secuser -tLDAP -x$SSA_SEC_CONFIG -d -uadmin -ppassword -iuserid -ffirstname
-llastname -wpassword
```

INDEX