Informatica

Informatica® Dynamic Data Masking
9.9.4

Administrator Guide

Informatica Dynamic Data Masking Administrator Guide
9.9.4
November 2023

Revision: 1
Publication Date: 2023-11-28

Table of Contents

Preface

Use the *Informatica Dynamic Data Masking Administrator Guide* to learn how to perform administrator tasks in Dynamic Data Masking. Understand how to configure security settings, create, and manage connections, and configure advanced Dynamic Data Masking settings.

Informatica Resources

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

Informatica Network

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

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

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

Informatica Knowledge Base

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

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

Informatica Documentation

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

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

Informatica Product Availability Matrices

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

Informatica Velocity

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

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

Informatica Marketplace

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

Informatica Global Customer Support

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

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

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

CHAPTER 1

Introduction to Dynamic Data Masking Administration

This chapter includes the following topics:

Dynamic Data Masking Administration Overview

Dynamic Data Masking is a data security product that operates between an application and a database to prevent unauthorized access to sensitive information. Dynamic Data Masking intercepts requests sent to the database and applies data masking rules to the request to mask the data before it is sent back to the application.

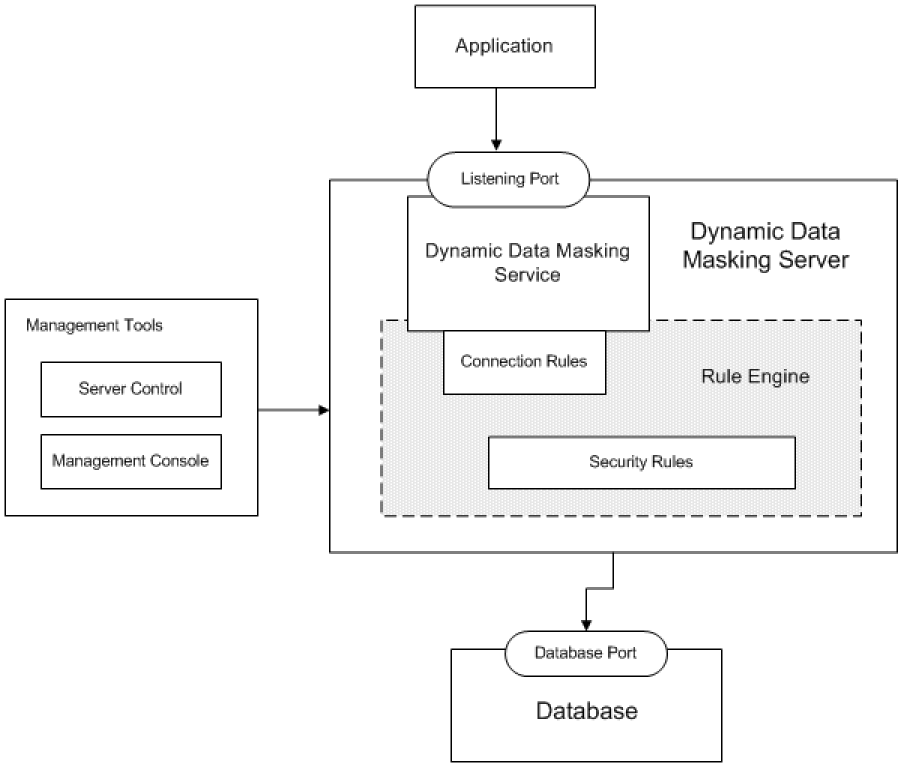
As an administrator, you use the Server Control program to manage the Dynamic Data Masking Server and start and stop the Dynamic Data Masking services. You use the Management Console to configure target databases, define listener ports, manage target databases, maintain system logs, and define rules. The administrative tasks that you perform help to ensure that Dynamic Data Masking operates effectively and efficiently.

Note: To prevent loss of data, such as connection rules, security rule sets, and database configurations, perform regular backups of the entire Dynamic Data Masking directory to a location on your network or an external storage location.

Dynamic Data Masking Architecture

Dynamic Data Masking acts as a security layer between the application and the database to protect sensitive data stored in the database. The Dynamic Data Masking Server intercepts SQL requests sent to the database and uses a set of connection and security rules to determine how to process the request.

The following figure shows the architecture of Dynamic Data Masking and how the Dynamic Data Masking Server relates to the application and the database:



The Dynamic Data Masking Server listens on the port where the application sends database requests. When the application sends a request to the database, the Dynamic Data Masking Server receives the request before it goes to the database. The Rule Engine uses the connection rules and security rules to determine the action to perform on the incoming request. The Dynamic Data Masking service sends the modified request to the database. The database processes the request and sends the results back to the application.

Dynamic Data Masking provides management tools that you can use to manage the Dynamic Data Masking Server and set up connection and security rules. With the Server Control tool, you can start, stop, and manage the Dynamic Data Masking Server. On the Management Console, you can configure and manage the Dynamic Data Masking services and create and manage connection and security rules.

Dynamic Data Masking Components

Dynamic Data Masking includes server components to intercept and process database requests and a client component to manage the server.

Dynamic Data Masking has the following components:

Dynamic Data Masking Server

The Dynamic Data Masking Server provides services and resources to intercept database requests and perform data masking tasks.

The Dynamic Data Masking Server includes the following components:

- Dynamic Data Masking services
- Rule Engine

Dynamic Data Masking Service

The Dynamic Data Masking service listens on the listener port to monitor and route incoming database requests.

You can run the following Dynamic Data Masking services:

Service	Description
DDM for Azure	Listens for and routes database requests for a Microsoft Azure SQL database. The service supports the SSL mode of communication.
DDM for DB2	Listens for and routes database requests for an IBM Db2 database. The service supports SSL and non-SSL modes of communication.
DDM for FAS	Listens for and routes database requests for Data Vault. The service supports SSL and non-SSL modes of communication.
DDM for Hive	Listens for and routes database requests for a Hive database. The service supports SSL and non-SSL modes of communication as well as Kerberos Authentication and Kerberos encrypted data.
DDM for Hive HTTP	Listens for and routes database requests for Hive databases using HTTP transport. The service supports SSL and non-SSL modes of communication, and Kerberos Authentication.
DDM for Impala	Listens for and routes database requests for an Impala database. The service supports SSL and non-SSL modes of communication as well as Kerberos Authentication.
DDM for Informix	Listens for and routes database requests in Informix native protocol to Informix databases.
DDM for Informix (DRDA)	Listens for and routes database requests in Distributed Relational Database Architecture protocol to Informix databases.
DDM for JDBC	Listens for database requests for a database that uses JDBC connectivity.
DDM for ODBC	Listens for database requests for a database that uses ODBC connectivity.
DDM for Oracle	Listens for and routes database requests for an Oracle database. The service supports SSL and non-SSL modes of communication.
DDM for PostgreSQL	Listens for and routes database requests for a PostgreSQL database.
DDM for SQL Server	Listens for and routes database requests for a Microsoft SQL Server database. The service supports SSL and non-SSL modes of communication.

Service	Description
DDM for Sybase	Listens for and routes database requests for a Sybase database.
DDM for Teradata	Listens for and routes database requests for a Teradata database.

Rule Engine

The Rule Engine evaluates incoming database requests and applies connection and security rules to determine how to route requests and mask data. The Rule Engine can modify the database request based on the rules defined in the Dynamic Data Masking Server.

The Rule Engine applies the following types of rules:

- Connection rule. Defines the conditions and actions that the Rule Engine applies to determine how to route a database connection request received from an application.
- Security rule. Contains the conditions and actions that define what to do with the database SQL request and how to apply SQL rewrites that manipulate the returned SQL result set.

Server Control

Server Control is a command line program that you use to configure and manage the Dynamic Data Masking Server. Use Server Control to start or stop the Dynamic Data Masking Server and services or to change the port number or password for the Dynamic Data Masking Server.

Management Console

The Management Console is a client application that you use to manage the Dynamic Data Masking Server. You can use the Management Console to create and manage rules and to configure and manage connections to databases.

Dynamic Data Masking Process

Dynamic Data Masking acts as a proxy server for the database.

The application sends a request to the database. As a proxy server, the Dynamic Data Masking Server intercepts the request and the Rule Engine evaluates the request before it sends the request to the database.

Dynamic Data Masking uses the following process to apply data masking to a database request:

1. The Dynamic Data Masking service listens on the listener port for requests sent to the database. When the application sends a database connection request, the Dynamic Data Masking service receives the request instead of the database.
2. The Rule Engine uses a connection rule to determine how to process the incoming connection request. The connection rule defines the criteria to identify and route the database request. If the database request matches the criteria, the Rule Engine determines the action to perform on the request. The connection rule action can include routing the connection to a specified database, host, and port, and applying a security rule set. The connection rule can block the connection request. If the database has a direct action, the connection rule can return a redirect request back to the application with the database host and port and the application can connect directly to the database.

   For example, you can define an Informatica ETL process or batch in Dynamic Data Masking that bypasses Dynamic Data Masking and reduces overhead on internal processes.

3. If the connection rule specifies the Use Rule Set processing action, Dynamic Data Masking connects the client to the database and applies the security rules to the SQL request and determines the action to perform on the request.
   The security rules can apply an action such as blocking the SQL request, modifying the SQL statement, doing nothing, or auditing the request.
4. The database processes the request and returns the results to the application. Because Dynamic Data Masking changes the SQL request, the results the database sends back to the application might be different from the results of the original database request.

Dynamic Data Masking Example

A reporting tool sends a request to the database for employee salaries. The connection matcher specifies that the Rule Engine apply a rule set called salary_rules to all incoming requests from the reporting tool.

The salary_rules rule set applies a masking function to all requests that reference employee salaries. The rule set restricts access to management salaries and masks the first three digits of the employee salary column.

The Dynamic Data Masking service intercepts the incoming SQL request, identifies that the request references the salary tables, rewrites it with the masking function, and sends the rewritten request to the database. The database receives the request and sends masked data back to the application through the Dynamic Data Masking service.

Dynamic Data Masking Implementation

Set up Dynamic Data Masking based on the type of environment where you want to implement data masking and the requirements for protecting sensitive data within that environment.

Use the following general guidelines when you set up Dynamic Data Masking:

1. Install the Dynamic Data Masking Server on the database server.
   Dynamic Data Masking adds a processing layer between applications and databases. To reduce latency in data transfer, you can install the Dynamic Data Masking Server on the same machine as the database.
2. Configure the client application and database server so that the application sends database requests to the Dynamic Data Masking listener port.
   For example, you install the Dynamic Data Masking Server on the Oracle database server. Edit the `listener.ora` file on the database server and set the database port to a different port number. Then set the Dynamic Data Masking listener port number to match the port number to which the application sends database requests.
3. Before you start using Dynamic Data Masking, examine the database and classify data as highly sensitive, moderately sensitive, or not sensitive based on the data classification policy of your organization.
   The data classification you identify determines the rules you need to create in Dynamic Data Masking.
4. Categorize users according to their access permissions.
   Create user access scenarios to determine the applications that access sensitive data and identify the data that must go through Dynamic Data Masking.
5. In Dynamic Data Masking, create connection and security rules to implement the data classification and user and application access policies of your organization.

Dynamic Data Masking Environments

You can use data masking to protect sensitive information in a production database and in non-production databases used for development, testing, and training purposes.

Production Environments

Privileged users, such as production database administrators, technical support personnel, and quality assurance teams, access personally identifiable information as a daily part of their jobs.

Non-Production Environments

Non-production environments, such as pre-production and development testing environments, typically use real data to test applications. The data can include identifiers such as national identification numbers or bank accounts numbers.

Dynamic Data Masking eliminates exposure of sensitive data without affecting the ability of users to perform their job functions.

Setting Up Dynamic Data Masking

Install and configure Dynamic Data Masking based on the type of environment where you want to implement data masking and the requirements for protecting sensitive data within that environment.

After installation, complete the following steps to set up Dynamic Data Masking:

1. Log in to the Management Console. Use the user name *admin* and the Dynamic Data Masking Server password to log in to the Management Console for the first time.
2. Optionally, change the default Internal authentication scheme. You can use LDAP or Active Directory authentication to authorize a list of Dynamic Data Masking administrators.
3. Create a Dynamic Data Masking service. Configure the listener port number to match the port number where the client sends requests to the database.
4. Define the database connection properties for the database that requires data masking.
5. Create a connection rule. Configure the rule to identify the database requests that must be masked. Assign a database and a security rule set to the connection rule set.
6. Create a security rule set. Define the rules for masking the data sent back to the application.

Dynamic Data Masking applies the security rule set to SQL requests from a client or application that initiates a connection that uses the Dynamic Data Masking listener port. When you modify rules within the security rule set, Dynamic Data Masking immediately applies the modified rules on new SQL requests.

Management Console

The Management Console is the client component of the Dynamic Data Masking Server.

You can install the Management Console on a remote machine or the local system to manage the Dynamic Data Masking service. Use the Management Console to manage and configure domains and Dynamic Data Masking services, define connection rules for Dynamic Data Masking services, define security rules, and configure target databases.

Logging In to the Management Console

You can access the Dynamic Data Masking components through the Management Console. Log in to the Management Console to manage target databases, configure listener ports, and define rules.

To log in to the Management Console, you need the server address and port number of the server that Dynamic Data Masking operates on and the administrator credentials.

Logging In to the Management Console on Windows

On Windows, open the Management Console through the Start menu.

1. On Windows 7 and earlier, select **Start** > **All Programs** > **Informatica** > **Dynamic Data Masking** > **Management Console**.

   On Windows 10 and later, select **Search the Web and Windows** > **All apps** > **Informatica Dynamic Data Masking** > **Management Console**.

   The **Login** window appears.
2. Verify that the **Server Host** and **Port** display the correct information for the Dynamic Data Masking Server.
3. Enter the Dynamic Data Masking administrator user name and password. If you use LDAP authentication, the user name must be in LDAP format. Click **Connect**.

   A tree is visible in the Management Console after you login successfully.

Logging In to the Management Console on Linux

On Linux, start the Management Console with the `mng` script.

You must have the X Window server installed on the machine that you use to log in to the Management Console.

1. Open a terminal and navigate to the Dynamic Data Masking installation directory.

   For example, you might enter the following command:

   ```
   cd /home/Informatica/DDM
   ```
2. Run the following command to start the Management Console:

   ```
   ./mng
   ```

   The **Login** window appears.
3. Verify that the **Server Host** and **Port** display the correct information for the Dynamic Data Masking Server.
4. Enter the Dynamic Data Masking administrator user name and password. If you use LDAP authentication, the user name must be in LDAP format. Click **Connect**.

   A tree is visible in the Management Console after you log in.

Database Management

A database node contains references to databases. The Dynamic Data Masking Server controls access to the databases that the database nodes reference.

A database node can reference an Oracle, Microsoft SQL Server, Db2, Informix, Sybase, Data Vault, Hive, or Teradata database. The Management Console tree can contain an unlimited number of database nodes. You can create database nodes under domain nodes. Database nodes do not have child nodes. You can set user permissions on database nodes.

Dynamic Data Masking Server Management

A server node contains a reference to the Dynamic Data Masking Server. By default, the server node is located under the root domain after a new installation of the Dynamic Data Masking Server.

The Management Console contains one server node. Each Dynamic Data Masking instance associates with one Dynamic Data Masking Server. You connect to a server when you log into the Management Console. The Dynamic Data Masking Server manages databases located under a parent domain or all sub domains of the server node in the tree.

The server node has a domain node parent. The server node can have Dynamic Data Masking service child nodes. You can edit and move the server node.

Note: You cannot add or remove the Dynamic Data Masking Server node with the Add or Remove options in the Management Console menu.

Dynamic Data Masking Service Management

The Dynamic Data Masking service routes SQL queries to Oracle, Microsoft SQL Server, Db2, Informix, Sybase, Data Vault, Hive, and Teradata databases.

The Dynamic Data Masking Server can contain single service nodes for each database. Create service nodes under the server node. Service nodes cannot have child nodes. You can add, edit, and remove service nodes.

Each Dynamic Data Masking service routes requests to a specific type of database. For example, the Dynamic Data Masking for Oracle service routes requests to Oracle databases and the Dynamic Data Masking for DB2 service routes requests to Db2 databases.

Dynamic Data Masking Listener Ports

The Dynamic Data Masking service controls connections between the client and the database through the listener port.

You must configure the database listener port to forward connections to the Dynamic Data Masking listener port. How you configure the listener ports depends on whether the Dynamic Data Masking service runs on the database server or on a standalone server. You can define the listener port that the Dynamic Data Masking service uses through the Services Editor in the Management Console.

If the Dynamic Data Masking service runs on a standalone server, you must route application connection requests to the Dynamic Data Masking listener port.

Defining a Dynamic Data Masking Listener Port

Before you can define a listener port, you must run the netstat system utility to verify port availability.

1. In the Management Console, right-click on a Dynamic Data Masking service and select **Edit**.

   The Service Editor appears.

2. Click **Add Port**.
3. Enter the listener port for the Dynamic Data Masking service.
4. Click **OK**.

Deleting a Dynamic Data Masking Listener Port

If the Dynamic Data Masking service no longer uses a listener port, delete the listener port with the Service Editor.

1. In the Management Console, right-click the Dynamic Data Masking service and select **Edit.**

   The Service Editor appears.
2. Select the port to delete.
3. Click **Remove port**.
4. Click **OK**.

Configuration Management

The Dynamic Data Masking configuration files store information about the Management Console tree nodes and user access.

You can find the Dynamic Data Masking configuration files in the following location:

```
<Dynamic Data Masking installation>/cfg
```

The `config.properties` file contains the configuration parameters of the Dynamic Data Masking Server and service nodes in the Management Console.

The `config.cfg` file contains information on domain, database, and security rule set nodes in the Management Console. The `config.cfg` file is binary and you must not edit it. The `config.cfg` file and the public key file, `config.pbk`, have a digital signature. Dynamic Data Masking updates the digital signature when a user performs an operation on the Dynamic Data Masking Server through the Management Console. If Dynamic Data Masking cannot verify the public key against the configuration file, the Dynamic Data Masking Server writes an error message to the `server.log` file and does not start.

Dynamic Data Masking Administrator Required Privileges

The Dynamic Data Masking administrator that creates the database connection must have privileges to access every object that the client user that receives masked data can access.

When the Dynamic Data Masking Server intercepts a client command, it might access client objects, such as tables, views, and stored procedures. If the administrator that created the database connection in Dynamic Data Masking cannot access the client objects, the query can return incorrect or unmasked data.

CHAPTER 2

Authentication

This chapter includes the following topics:

Authentication Overview

When you use the Management Console to connect to the Dynamic Data Masking Server, you must log in with a user name and password. Dynamic Data Masking user authentication depends on the authentication scheme that you configure.

You can configure the following authentication schemes for Dynamic Data Masking:

- LDAP authentication. Authentication scheme that uses the LDAP software protocol to locate organizations, groups, individuals, and other resources in a network.
- Active Directory authentication. Authentication scheme that uses the Active Directory service to enforce security for users and resources in a Windows network.
- Internal authentication. Authentication scheme that uses the Dynamic Data Masking Server password to authenticate users who log in to the Management Console.

Use the Management Console to set the authentication scheme for Dynamic Data Masking. By default, the Management Console uses internal authentication. The first time you log in to the Management Console, you enter the user name and the server password that you created when you installed the Dynamic Data Masking Server.

LDAP Authentication

You can use LDAP authentication to authenticate users who log in to the Management Console.

If you use LDAP authentication, you must provide the user name in LDAP format when you log in to the Management Console.

The following table describes the LDAP properties you configure if you use the LDAP authentication scheme:

Property	Description
Domain Server	Host name or IP address of the LDAP server.
Domain Server Port	Port number for the LDAP server.
Security Protocol	LDAP transport security protocol. Select SSL or None.
Administration Group	Distinguished name (DN) of the LDAP group to use for authentication.
Root DN	Base distinguished name (DN) of the LDAP domain in which to begin user lookups.
User Object Class	Object class type used for the LDAP user object.
User MemberOf Attribute	Locate users by attribute. Name of the user attribute to use to identify the groups associated with a user.
Group Object Class	Object class type used for the LDAP group object.
Group MemberOf Attribute	Locate users by attribute. Name of the group attribute to use to identify the groups associated with a user.
Group Members Attribute	Locate users by group. Name of the attribute to use to identify the members of a group.

Active Directory Authentication

You can use Active Directory authentication to authenticate users who log in to the Management Console.

The following table describes the properties you configure if you use the Active Directory authentication scheme:

Property	Description
Domain Server	Host name or IP address of the Active Directory server.
Domain Server Port	Port number for the Active Directory server.
Domain	Name of the domain name that contains the Active Directory server
Administration Group	Name of the Active Directory group to use for authentication.

Internal Authentication

Dynamic Data Masking provides a simple authentication scheme that authenticates a user based on the Dynamic Data Masking Server password.

When you install the Dynamic Data Masking Server, you must create a password for the server. Dynamic Data Masking uses the server password for authentication when you initially log in to the Management Console.

The Management Console login requires a user name and password. The Internal authentication scheme uses only the password for authentication. It uses the user name to identify the user and to track user activity in the Management Console.

You can use the Server Control command SetInternalPassword to change the Dynamic Data Masking Server password at any time.

Admin User Name

The user name `admin` is a key user name for Dynamic Data Masking authentication. If you log in to the Management Console with the `admin` user name, Dynamic Data Masking uses the server password for authentication.

The `admin` user name supercedes the type of authentication configured for the Dynamic Data Masking. You can log in to the Management Console with the `admin` user name at any time. If you log in to the Management Console with the `admin` user name, Dynamic Data Masking authenticates based on the server password even if the Dynamic Data Masking is configured to use LDAP or Active Directory authentication.

When you log in as `admin`, Dynamic Data Masking uses internal authentication only for the session. It does not permanently change the authentication scheme configured for Dynamic Data Masking. When you log out, you can log back in with an LDAP or Active Directory user account.

If Dynamic Data Masking uses LDAP authentication and the configuration of the LDAP server or network changes, you might not be able to log in to the Management Console. You must update the LDAP server attributes in the Management Console for LDAP authentication to work properly. Log in to the Management Console with the `admin` user name and server password and update the attributes of the LDAP server. The next time you log in, you can use LDAP authentication.

Setting Up Authentication

Use the Management Console to set up the authentication system for Dynamic Data Making.

To set the authentication for the first time, you must log in with a user name and the server password. You set the initial server password when you install the Dynamic Data Masking Server.

1. Log in to the Management Console.
2. In the Management Console tree, select the Dynamic Data Masking Server.

   The properties of the Dynamic Data Masking Server displays on the right panel, showing the current authentication scheme.
3. Click **Tree** > **Edit** to edit the properties of the Dynamic Data Masking Server.
4. In the **Edit** window, select the authentication scheme you want to use.

   The **Edit** window displays the properties required to connect to the authentication server.
5. Enter the authentication server domain and user properties.

- For the LDAP authentication scheme, enter the LDAP domain, group, and user properties. Consult the LDAP administrator to get the correct information.
- For the Active Directory authentication scheme, enter the Active Directory domain properties and the administration group. Consult the Active Directory administrator to get the correct information.
- For the Internal authentication scheme, you do not need to enter any information. Use the Server Control command SetInternalPassword to manage the Dynamic Data Masking Server password.

6. Click OK.

Log out and log in again to the Management Console to verify that the selected authentication scheme is in effect.

CHAPTER 3

Security

This chapter includes the following topics:

Target Database Credentials Management

When you configure a target database, the database credentials that you enter are stored in a secure keystore, not in the Dynamic Data Masking Management Console tree. A security provider allows access to the keystore.

The keystore and security provider protect database credentials and prevent unauthorized access to the target database. When you configure a database, you can choose to use either the default keystore or a custom keystore and security provider. If you choose the default keystore, Dynamic Data Masking also uses the default security provider. Both the default keystore and default security provider are preconfigured for use in Dynamic Data Masking. If you choose a custom keystore and security provider, you must configure Dynamic Data Masking for use with the custom keystore and security provider.

Default Keystore

The default keystore and security provider are preconfigured for use with any database supported by Dynamic Data Masking.

The default keystore is a JCEKS-type keystore that permits both read and write operations. If the keystore does not already exist, it is created in the following location when the Dynamic Data Masking Server starts: `<DDM>/cfg/ddm.jceks`

When you configure the target database, you can select the default keystore option and then enter the database user name and password. When you save the database object, an alias is automatically generated and saved in the keystore along with the database credentials. The Dynamic Data Masking Server reads the database credentials from the keystore to create an internal connection to the database. The alias is not visible in the database form, and the Dynamic Data Masking Server never sends the credentials to the client or outside of the Dynamic Data Masking Server.

Dynamic Data Masking upgrades each database object in the following process:

1. Sets the default keystore in the database object.
2. Sets the automatically-generated alias in the database object.
3. Saves the alias, user name, and password of the database object in the default keystore.
4. Removes the user name and password from the database object.
5. Saves the resulting database object in the Management Console tree. The database object contains the alias and default keystore, but not the user name or password.

This upgrade is performed only when the database objects were created in versions of Dynamic Data Masking prior to 9.8.3.

Custom Keystore

You can use a custom keystore and security provider to store and access the target database credentials. To use a custom keystore and security provider, you must create an XML configuration file called `ddm.security`. If you want to use CyberArk as a security provider, you must also create a CyberArk properties file. Then you can create the target database connection.

ddm.security File

The `ddm.security` file contains the information used to define the custom keystore and security provider. To configure custom keystores and security providers, create the file in the following location: `<DDM>/cfg/ddm.security`

Use the following parameters to configure the `ddm.security` file for the custom security provider:

Name	Description
<fqcn>	Mandatory. Fully-qualified class name of the security provider. For example: com.security.provider.MyProvider
<file>	Optional. Provider-specific initialization parameter. For example, the path to a configuration file.

Use the following parameters to configure the `ddm.security` file for the custom keystore:

Name	Default	Description
storeName	-	Mandatory. Unique name of the keystore. Once you have defined the keystore name, do not modify it.
storeType	-	Mandatory. Type of keystore. For CyberArk, enter the storeType as CyberArk.
storeFile	null	Optional. Path to the keystore file.
storePassword	null	Optional. Keystore password.
provider	-	Optional. Name of the custom security provider that Provider.getName() returns. Note that this is not the name of the class. If the security provider is CyberArk, this parameter is mandatory. Provide the name of the security provider. This name should match the provider.name property in the CyberArk properties file.
encrypted	false	Optional. Specify a clear password for the keystore in the `ddm.security` file. Dynamic Data Masking encrypts the password at run-time and sets *encrypted=true* in the file.

After you configure the `ddm.security` file, start the Dynamic Data Masking Server. When you configure the database object, enter the keystore name defined in the `ddm.security` file and the alias associated with the database user name and password in the custom keystore. For CyberArk accounts, the alias name was defined during creation of the CyberArk account.

Custom security providers can allow read-only or read and write access to the keystore. For a read-only keystore, enter the existing alias.

Sample ddm.security Files

The following file is an example of a `ddm.security` file that contains two custom security providers, one with a configuration file as the provider-specific initialization parameter, and another provider without an initialization parameter. The file contains three custom keystores that have unique names, with two keystores including the name of the designated security provider.

```
<?xml version="1.0"?>
<XML>
    <keyStores type="ArrayList">
        <entry type="StoreDescriptor">
            <storeName>store1</storeName>
            <storePassword>admin</storePassword>
            <storeType>JCEKS</storeType>
            <encrypted>false</encrypted>
            <storeFile>cfg/store.jceks</storeFile>
        </entry>
        <entry type="StoreDescriptor">
            <storeName>store3</storeName>
            <storePassword>admin</storePassword>
            <storeType>PKCS12</storeType>
            <encrypted>false</encrypted>
            <provider>PKCS12-Provider-5</provider>
            <storeFile>cfg/store.pkcs12</storeFile>
        </entry>
        <entry type="StoreDescriptor">
            <storeName>store2</storeName>
            <storeType>PKCS11</storeType>
            <encrypted>false</encrypted>
            <provider>MyProvider-HSM</provider>
```

```
                <storeFile>cfg/store.pkcs11</storeFile>
            </entry>
        </keystores>
        <providers type="ArrayList">
            <entry type="ProviderDescriptor">
                <file>cfg/xyz.conf</file>
                <fqcn>com.security.provider.XYZProvider</fqcn>
            </entry>
            <entry type="ProviderDescriptor">
                <fqcn>com.security.provider.MyProvider</fqcn>
            </entry>
        </providers>
    </XML>
```

The following file is an example of a `ddm.security` file configured for use with a CyberArk keystore:

```
<?xml version="1.0"?>
<XML>
  <keyStores type="ArrayList">
 <entry type="StoreDescriptor">
    <storeType>CyberArk</storeType>
    <storeName>DDMQASafe</storeName>
    <provider>QASafeProvider</provider>
  </entry>
 </keyStores>

 <providers type="ArrayList">
  <entry type="ProviderDescriptor">
    <file>cfg\CyberArk_DDMQASafe.props</file>
    <fqcn>com.informatica.security.jce.cyberark.CyberarkProvider</fqcn>
  </entry>
 </providers>
</XML>
```

You can configure multiple CyberArk entries in the `ddm.security` file with the same method.

CyberArk Properties File

If you use CyberArk as a security provider, you must create a CyberArk properties file in addition to the `ddm.security` file. The CyberArk properties file is a text properties file that contains parameters specific to CyberArk.

Create the CyberArk properties file within the `<DDM>/cfg/` folder. For example: `<DDM>/cfg/CyberArk.props`. Provide the location of the file in the <file> parameter of the security provider section of the `ddm.security` file.

The CyberArk properties file includes the following parameters:

Name	Description
provider.name	Name of the security provider. Name must match the <provider> tag in the keystore section of the `ddm.security` file.
provider.client.appid	Application ID. The application ID was created during the CyberArk installation.
provider.safe	Mandatory. Name of the specific safe within CyberArk.

Name	Description
provider.uniqueattr.name	Unique attribute name of the account. By default, the name of the account is a unique attribute that is internally mapped with the string "Object." In this case, Object is the value for the uniqueattr.name parameter. However, if you have selected any other attribute other than name as an unique identifier, give that attribute name as the value of the property. For example, if you selected the attribute host as the unique identifier, give the value of provider.uniqueattr.name as host.
provider.folder.path	Path from the root to the folder containing the given account. If you leave this parameter blank, Dynamic Data Masking assumes the account is under the root.

Note: If you plan to use CyberArk as a security provider, you must put the CyberArk `JavaPasswordSDK.jar` file into the `<Dynamic Data Masking installation>/lib/ext` directory to complete the integration. The `JavaPasswordSDK.jar` file is located in the `ApplicationPasswordSdk` directory of the CyberArk AIM installation (on Microsoft Windows) or the `/opt/CARKaim/sdk/` directory (on Linux).

Sample CyberArk Properties File

The following file is an example of a CyberArk properties file:

```
provider.name=QASafeProvider
provider.client.appid=DDMJavaTest
provider.safe=DDMQASafe
provider.uniqueattr.name=Object
provider.folder.path=root\\subfolder
```

Managing Keystores and Aliases

You can use a Server Control command to change the default keystore to a custom keystore, or a custom keystore to the default keystore. To change the keystore from default to custom, you provide the custom keystore name and alias in the command. To change the keystore from custom to default, you provide the database user name and password within the command. In both scenarios you must provide the database path.

1. Launch Server Control from the **Start** menu on Microsoft Windows, or use the `server` shell script on Linux or UNIX.
2. Enter the following syntax: `server config setKeyStore -path <path> [-storeName <storeName> -alias <alias>] | [-user <user> -password <password>]`

   Path is a required parameter. If you want to change the keystore from default to custom, both keystore name and alias are required parameters. If you want to change the keystore from custom to default, both user name and password are required parameters. Do not use both sets of parameters in the same command invocation.
   You can also set another alias in a database object. The alias must already exist in the designated keystore. You can set the alias, the keystore name, or both.

SSL Communication in Dynamic Data Masking

Dynamic Data Masking supports SSL communication between the Dynamic Data Masking Server and multiple database instances, types, and clients, such as the Management Console and Server Control. The Dynamic Data Masking Server can load multiple existing keystores and truststores, which in most cases you can copy from the database or database client to the Dynamic Data Masking installation without any modification.

When you enable SSL communication, you configure the `cfg/ddm.security` file for keystores and truststores used by the Dynamic Data Masking Server. You also configure the `cfg/client.security` file for truststores used by clients such as the Management Console and Server Control. Configuration parameters for the `cfg/client.security` and `cfg/ddm.security` files are the same.

You also use the `cfg/ddm.security` file to configure key strategies and trust strategies. Key strategies are required when Dynamic Data Masking uses multiple signed certificates to perform the handshake with database clients. Trust strategies tell Dynamic Data Masking how to handle a certificate that does not exist in the Dynamic Data Masking truststore and is therefore rejected by the trust manager.

Dynamic Data Masking supports various security protocol and cipher suites. You can define global settings for security protocols and cipher suites, or you can configure protocols and ciphers that map to a specific Dynamic Data Masking host and port.

You can enable SSL communication for Oracle, IBM Db2, and Microsoft SQL Server target databases.

Enable SSL Communication in Dynamic Data Masking

After you install or upgrade Dynamic Data Masking, you can enable SSL communication. The upgrade and installation processes do not automatically configure Dynamic Data Masking for SSL communication. However, an installation or upgrade does prepare Dynamic Data Masking for SSL communication.

After upgrade or installation, the Dynamic Data Masking Server generates a self-signed certificate in the file `cfg/ddm.jceks`. By default, the Dynamic Data Masking Server is not configured with keystores and key strategies. The Dynamic Data Masking Server uses the automatically generated self-signed certificate to perform the SSL handshake.

The Dynamic Data Masking administration tools, for example the Management Console and command line tools, are not preconfigured with truststores and trust strategies. By default, when SSL is enabled, the administration tools accept any server certificates without SSL authentication of the certificate.

Enabling SSL Communication in Dynamic Data Masking

To enable SSL communication, stop the Dynamic Data Masking Server and run a server network command to switch the administration port from clear to SSL. You can also run a command to view the current network settings.

From the Management Console, verify that you have selected the **SSL** check box on the Add/Edit Database form for the specific database that you want to enable SSL communication for. In the service configuration form for the corresponding Dynamic Data Masking service, input "SSL" on the appropriate row of the **Security** column.

1. To switch the Dynamic Data Masking Server administration port between SSL and clear, run the following command:

```
server network edit [<host>:]<port>[;SSL]
```

- For example, to switch the administration port from clear to SSL:

  ```
  server network edit 8195;SSL
  ```

  or

  ```
  server network edit 127.0.0.1:8195;SSL
  ```

- To switch the administration port from SSL to clear:

  ```
  server network edit 8195
  ```

  or

  ```
  server network edit 127.0.0.1:8195
  ```

On Linux operating systems, you must enclose the `server network edit` parameter value in double quotes. For example: `server network edit "8195;SSL"`

2. To view the current network settings, run the following server network command:

   ```
   server network
   ```

Keystore Configuration

To configure SSL communication in Dynamic Data Masking, copy keystores from the target database to the Dynamic Data Masking Server. Usually you can copy the database keystores without modifying them. After you copy the keystores to the Dynamic Data Masking Server installation, configure the `cfg/ddm.security` file to set the keystore parameters.

If a database certificate contains information that limits the area where you can use the certificate, you might have to change the installation. For example, if a certificate is valid only on the database host, you can install Dynamic Data Masking on the same host. Then you can copy the database keystore as-is to the Dynamic Data Masking Server.

Dynamic Data Masking can use one key password for each keystore to read key entries. Dynamic Data Masking tries to read as many key entries as possible using the given key password, and skips entries with another key password. If you do not specify the parameter keyPassword in the `cfg/ddm.security` file, Dynamic Data Masking tries to read all of the key entries using an empty password and then the keystore password. If all tries fail, Dynamic Data Masking logs a message that it cannot use that keystore.

The storePassword and keyPassword parameters can be the same. For example, the following image shows the keystore configuration section of the `cfg/ddm.security` file:

```
<keyStores type="ArrayList">
    <entry type="StoreDescriptor">
        <storePassword>OILOPHAFGKACGPLKONFJNMGFIJGMADKF</storePassword>
        <storeType>JCEKS</storeType>
        <encrypted>true</encrypted>
        <keyPassword>OILOPHAFGKACGPLKONFJNMGFIJGMADKF</keyPassword>
        <storeFile>cfg/keyStore.jceks</storeFile>
    </entry>
    <entry type="StoreDescriptor">
        <storePassword>OILOPHAFGKACGPLKACCNBJIDCAOGCEJO</storePassword>
        <storeType>JCEKS</storeType>
        <encrypted>true</encrypted>
        <keyPassword>OILOPHAFGKACGPLKACCNBJIDCAOGCEJO</keyPassword>
        <storeFile>cfg/ddmOracleKeyStore.jceks</storeFile>
        <preferred>true</preferred>
    </entry>
</keyStores>
```

The following table describes the keystore properties that you configure in the keystore section of the `cfg/ddm.security` file:

Parameter Name	Description	Required	Default Value
storeType	Type of keystore, for example JKS, JCEKS, or PKCS12.	Yes	
storeFile	Path to the keystore file.	No	
storePassword	Password for the keystore.	No	
keyPassword	Key password to read key entries in the keystore.	No	
provider	Name of the specific security provider that works with the keystore.	No	
encrypted	If set to "true," Dynamic Data Masking encrypts unencrypted passwords at run-time.	No	false
preferred	If set to "true," Dynamic Data Masking loads the preferred keystore first, before any other keystores.	No	false

At run time, Dynamic Data Masking loads keystores and searches aliases in the following order:

1. The preferred keystore, specified in the `cfg/ddm.security` file.
2. Keystores specified using the Java Virtual Machine system property command-line options, for example in the file `jvm.params`.
   - javax.net.ssl.keyStore
   - javax.net.ssl.keyStoreType
   - javax.net.ssl.keyStorePassword
   - javax.net.ssl.keyStoreProvider
   - javax.net.ssl.keyPassword

Note: the key password javax.net.ssl.keyPassword is not a standard Java Virtual Machine option.

3. A keystore of the Java Virtual Machine, if any.
4. Other keystores specified in the `cfg/ddm.security` file.

You can configure Dynamic Data Masking to read database credentials and signed certificates from the same custom keystore.

If you configure a custom security provider, for example CyberArk, to work with the keystore, the security provider must support reading key entries from the keystore. Otherwise, Dynamic Data Masking is not able to get signed certificates from the keystore to perform the SSL handshake with the clients.

If you use one key password to create all key entries in the keystore, Dynamic Data Masking can load all signed certificates at run time. Otherwise, if you use different key passwords to create key entries, Dynamic Data Masking will reads as many key entries as possible using one specified key password and skips key entries with another key password.

If you do not want to mistakenly store database credentials and signed certificates in the same keystore, do not specify the <storeName> parameter in the keystore configuration section of the `cfg/ddm.security` file. If you do not specify the <storeName> parameter, that store is unavailable for selection when you configure a database in the **Add Database** form.

Dynamic Data Masking internally manages the default keystore file, `cfg/ddm.jceks`. Do not change the default keystore, for example by importing certificates to the default keystore. Do not specify the default keystore in the `cfg/ddm.security` file.

Truststore Configuration

To configure SSL communication in Dynamic Data Masking, copy truststores from the database client to the Dynamic Data Masking Server. Usually you can copy the client truststores without changing them. After you copy the truststores to the Dynamic Data Masking Server installation, configure the `cfg/ddm.security` file to set the truststore parameters.

If necessary, you can configure the trust strategy to accept new database certificates at run time, and permanently store them in the preferred truststore.

The following table describes the truststore properties that you configure in the truststore section of the `cfg/ddm.security` file:

Parameter Name	Description	Required	Default Value
storeType	Type of truststore, for example JKS, JCEKS, or PKCS12.	Yes	
storeFile	Path to the truststore file.	No	
storePassword	Password for the truststore.	No	
provider	Name of the specific security provider that works with the truststore.	No	

Parameter Name	Description	Required	Default Value
encrypted	If set to "true," Dynamic Data Masking encrypts unencrypted passwords at run-time.	No	false
preferred	If set to "true," Dynamic Data Masking loads the preferred truststore first, before any other truststores. Dynamic Data masking can add new certificates to the preferred truststore at run time. If you do not set a preferred truststore, Dynamic Data Masking might use new accepted certificates for the duration of current session. For more information, see "Trust Strategies" on page 36. You can set one preferred truststore in the `cfg/ddm.security` file.	No	false

At run time, Dynamic Data Masking loads truststores and checks trusted certificates in the following order:

1. The preferred truststore, specified in the `cfg/ddm.security` file.
2. Truststores specified using the Java Virtual Machine system property command-line options, for example in the file `jvm.params`.
   - javax.net.ssl.trustStore
   - javax.net.ssl.trustStoreType
   - javax.net.ssl.trustStorePassword
   - javax.net.ssl.trustStoreProvider
3. A truststore of the Java Virtual Machine, if any.
4. Other truststores specified in the `cfg/ddm.security` file.

If you do not configure the Dynamic Data Masking administrative tools with truststores in the file `cfg/client.security`, the administrative tools accept any signed certificate that the Dynamic Data Masking Server provides.

Note that JDBC database drivers can use only one truststore with all public certificates which is set in the Java Virtual Machine system properties.

The Dynamic Data Masking Server automatically generates a composite temporary truststore file named `cfg/ddm.temp.jceks`.

This truststore is used to discover metadata in SSL-enabled databases and perform impersonation in Dynamic Data Masking. It is also used to test the connection to SSL-enabled databases through the **Add Database** form.

The composite truststore file contains entries added from all configured truststores, including the truststore defined in the Java Virtual Machine system properties. That composite truststore is set, and can override an existing truststore, in the Java Virtual Machine system properties. The Dynamic Data Masking Server always creates a new temporary composite truststore at startup for the current Dynamic Data Masking Server session. The Dynamic Data Masking Server also deletes the old temporary composite truststore, if it remains after a previous session.

The temporary composite truststore is used to support the functionality of JDBC drivers in the Java Virtual Machine. If you have not configured the Dynamic Data Masking Server to work with SSL-enabled databases, the Dynamic Data Masking Sever does not generate the composite truststore.

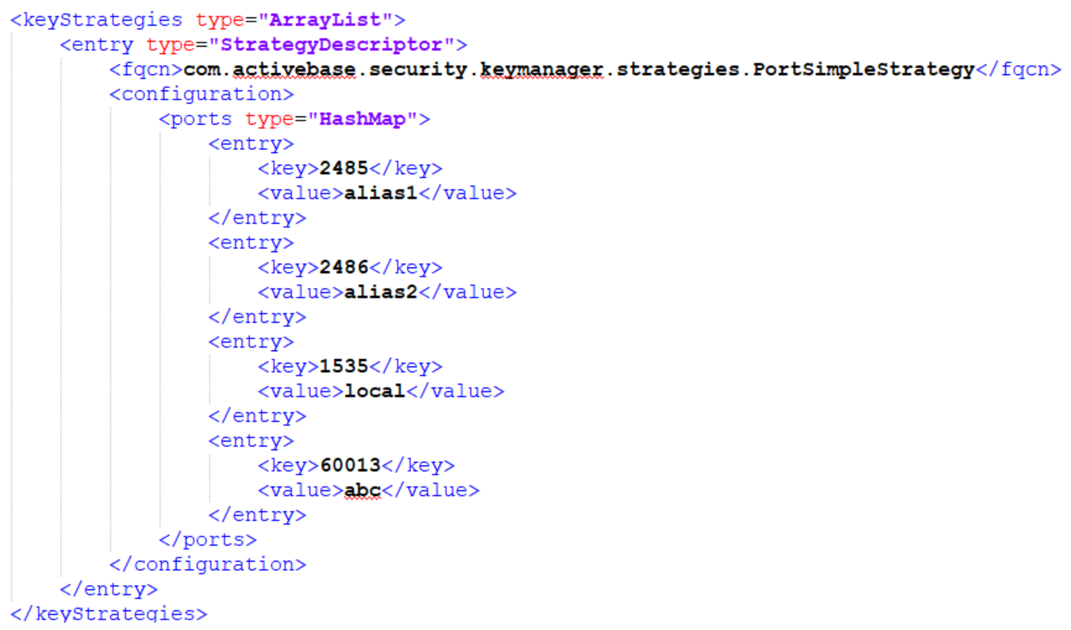
Key Strategies

If Dynamic Data Masking uses multiple signed certificates to perform the handshake with database clients, you must configure a simple port strategy or an advanced port strategy in the `cfg/ddm.security` file.

By default, the standard Java implementation chooses the first alias it finds in the keystore for which there is a private key and a key of the right type for the chosen cipher suite. The selected alias does not necessarily correspond to the requested domain, which can cause certificate errors.

To overcome this limitation, you can configure a simple port strategy or an advanced port strategy that map each SSL host and port combination in Dynamic Data Masking to a specific alias. Each alias corresponds to a key entry with a private key and signed certificate in the keystore that Dynamic Data Masking uses for the SSL handshake with the client.

The following image shows the keystrategies configuration section of the `cfg/ddm.security` file:



```
<keyStrategies type="ArrayList">
    <entry type="StrategyDescriptor">
        <fqcn>com.activebase.security.keymanager.strategies.PortSimpleStrategy</fqcn>
        <configuration>
            <ports type="HashMap">
                <entry>
                    <key>2485</key>
                    <value>alias1</value>
                </entry>
                <entry>
                    <key>2486</key>
                    <value>alias2</value>
                </entry>
                <entry>
                    <key>1535</key>
                    <value>local</value>
                </entry>
                <entry>
                    <key>60013</key>
                    <value>abc</value>
                </entry>
            </ports>
        </configuration>
    </entry>
</keyStrategies>
```

Alias names must be unique for all keystores. If you do not give unique alias names, Dynamic Data Masking uses the key entry with the first found matching alias to handshake with the client.

You do not need to configure the keystrategy section when you have not configured the Dynamic Data Masking Server with a keystore, or when a keystore configured in the Dynamic Data Masking Server contains only one signed certificate.

If you have not configured the Dynamic Data Masking Server with a keystore, Dynamic Data Masking uses the self-signed certificate generated in the default Dynamic Data Masking keystore file `cfg/ddm.jceks`. This certificate is associated with the reversed alias name *ddm_self_signed*. Do not use this alias name in any other keystores.

In this scenario, the Dynamic Data Masking Server provides the self-signed certificate to the client.

If you have configured the Dynamic Data Masking Server with one keystore, defined in the file `jvm.params` or `cfg/ddm.security`, that contains only one signed certificate, Dynamic Data Masking uses that certificate.

The following table describes the keystore strategy requirements for both the Dynamic Data Masking Server and the administrative tools:

Dynamic Data Masking Server		**Administrative Tools**	
Keystore or Keystores	**Dynanic Data Masking Server SSL Functionality**	**Truststore or Truststores**	**Dynamic Data Masking Administrative Tools SSL Functionality**
not set	Provides the self-signed certificate to the SSL client. The key strategy is not relevant.	not set	Accepts any certificate. Can connect.
set with certificates	Provides to SSL client: - If you have not configured a port strategy, provides the first found signed certificate. - If you have configured a port strategy, provides a signed certificate associated with an alias mapped to a specific SSL port.	not set	Accepts any certificate. Can connect.
not set	Provides the self-signed certificate to SSL client. Key strategy is not relevant.	set with certificates	Cannot connect, because the Dynamic Data Masking Server has no matching private key.
set with certificates	Must have a configured port strategy. Provides to SSL client a signed certificate associated with an alias mapped to a specific SSL port.	set with certificates	Can connect, but the Dynamic Data Masking Server must have matching private key.

Trust Strategies

A trust strategy tells Dynamic Data Masking how to handle a certificate that does not exist in the Dynamic Data Masking truststore and is therefore rejected by the trust manager.

When Dynamic Data Masking fails to validate a specific certificate, it consults the configured trust strategy, if the strategy trusts the certificate chain. If a trust strategy determines that it trusts the certificate, it returns an "accept" value to the trust manager. If the trust strategy accepts the certificate and returns the "accept_permanently" or "accept_temporarily" value to the trust manager, Dynamic Data Masking adds the certificate to the preferred or temporary truststore. Dynamic Data Masking then validates the certificate chain again with the likely positive result.

If the trust strategy determines that it cannot trust a certificate, it returns a "reject" value to the trust manager. If all configured trust strategies reject the certificate and return a "reject" value to the trust manager, Dynamic Data Masking denies the client connection.

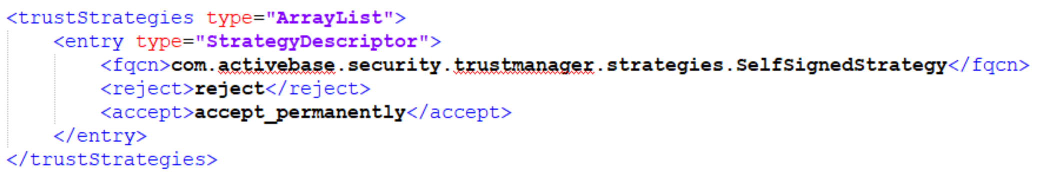
Trust strategies allow Dynamic Data Masking to add accepted certificates to the Dynamic Data Masking truststore at run time, similar to web browsers. For example, if a database uses a self-signed certificate to perform the SSL handshake with the client, you can configure the self-signed strategy in Dynamic Data Masking to accept all self-signed certificates. Dynamic Data Masking adds those certificates to the truststore. Similarly, you can use different strategies to accept and store new certificates in the truststore without any manual action.

Dynamic Data Masking is installed with two trust strategies that you can configure in the `cfg/ddm.security` file:

- A strategy to accept self-signed certificates: com.activebase.security.trustmanager.strategies.SelfSignedStrategy

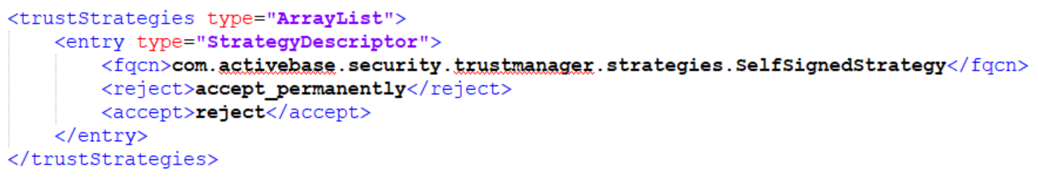
  When you configure the strategy, you provide a value for both the <accept> and <reject> parameters. The <accept> parameter tells Dynamic Data Masking what action to take on self-signed certificates. The <reject> value tells Dynamic Data Masking what action to take on certificates that are not self-signed.

  For example, the following image shows a configuration to permanently accept all self-signed certificates and reject all other certificates:

```
<trustStrategies type="ArrayList">
    <entry type="StrategyDescriptor">
        <fqcn>com.activebase.security.trustmanager.strategies.SelfSignedStrategy</fqcn>
        <reject>reject</reject>
        <accept>accept_permanently</accept>
    </entry>
</trustStrategies>
```

  The following image shows a configuration to reject all self-signed certificates and permanently accept all other certificates:

```
<trustStrategies type="ArrayList">
    <entry type="StrategyDescriptor">
        <fqcn>com.activebase.security.trustmanager.strategies.SelfSignedStrategy</fqcn>
        <reject>accept_permanently</reject>
        <accept>reject</accept>
    </entry>
</trustStrategies>
```

- A strategy to accept certificates with a matching fingerprint: com.activebase.security.trustmanager.strategies.FingerprintStrategy

  This strategy generates a fingerprint from the certificate that Dynamic Data Masking checks using the specified crytographic <algorithm> parameter, "SHA-1" for example. The strategy compares the generated fingerprint with the configured <fingerprint> parameter. You also configure the <reject> and <accept> parameters to tell Dynamic Data Masking what to do with certificates that are accepted or rejected based on the strategy logic.

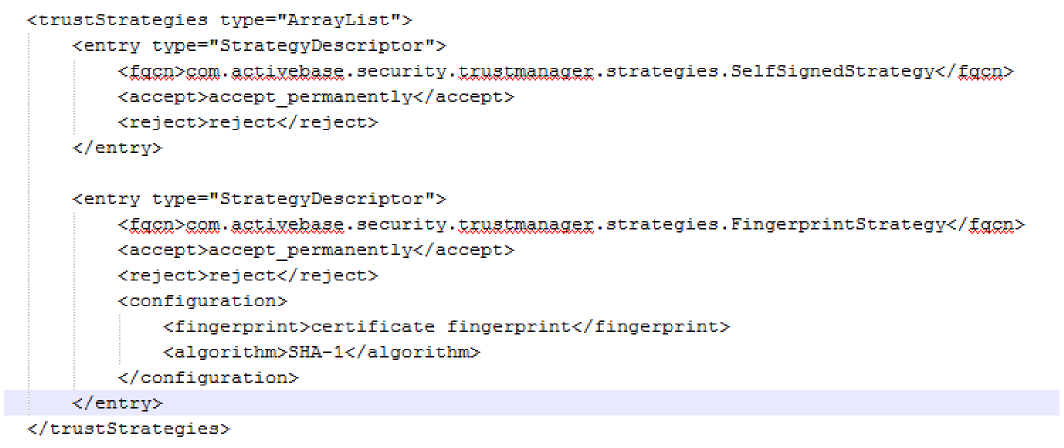
The final result of trust strategy processing is a combination of the following steps:

1. Acceptance or rejection of the certificate, based on the logic of the strategy.
2. Assignment of the appropriate value to the acceptance or rejection.

The following table describes when Dynamic Data Masking stores an accepted certificate in either the preferred or temporary truststore, based on the accept value of the trust strategy and the preferred value of the truststore:

Preferred Store	Trust Strategy Result	Client Connection	Action
set	Accept permanently	Allow	Dynamic Data Masking permanently stores the new certificate and uses it for the duration of the current and all future sessions.
set	Accept temporarily	Allow	Dynamic Data Masking temporarily accepts the new certificate and uses it for the duration of the current session.
not set	Accept permanently	Allow	Dynamic Data Masking temporarily accepts the new certificate and uses it for the duration of the current session.
not set	Accept temporarily	Allow	Dynamic Data Masking temporarily accepts the new certificate and uses it for the duration of the current session.
set	Reject	Refuse	No action taken.
not set	Reject	Refuse	No action taken.
not set	-	Refuse	No action taken.
set	-	Refuse	No action taken.

The following image shows a configuration of the two trust strategies in the `cfg/ddm.security` file:



```
<trustStrategies type="ArrayList">
    <entry type="StrategyDescriptor">
        <fqcn>com.activebase.security.trustmanager.strategies.SelfSignedStrategy</fqcn>
        <accept>accept_permanently</accept>
        <reject>reject</reject>
    </entry>

    <entry type="StrategyDescriptor">
        <fqcn>com.activebase.security.trustmanager.strategies.FingerprintStrategy</fqcn>
        <accept>accept_permanently</accept>
        <reject>reject</reject>
        <configuration>
            <fingerprint>certificate fingerprint</fingerprint>
            <algorithm>SHA-1</algorithm>
        </configuration>
    </entry>
</trustStrategies>
```

The certificate fingerprint is a long string that represents a real fingerprint.

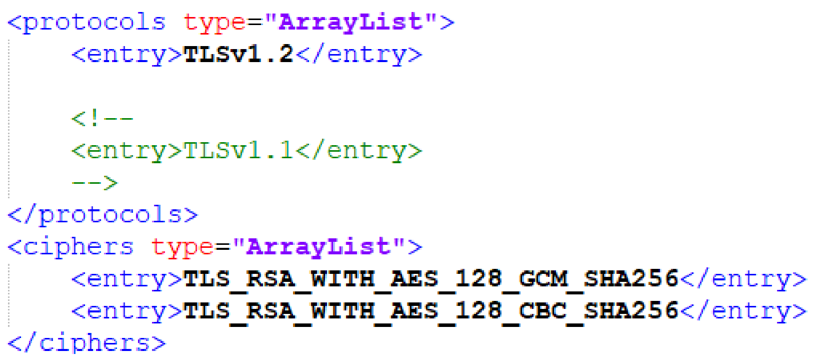
Protocols and Ciphers Suites

To improve network security in Dynamic Data Masking, you can set stronger security protocols and cipher suites in the `cfg/ddm.security` file. Stronger security protocols might help prevent multiple types of malicious attacks, such as a man-in-the-middle attack. You can define global settings for protocols and cipher suites, or you can configure protocols and ciphers that map to a specific Dynamic Data Masking host and port.

Global Configuration

You can define protocols and ciphers as global settings that apply to all SSL ports in Dynamic Data Masking. The following image shows an example configuration in the `cfg/ddm.security` file for:

- TLSv1.2 communication between clients and Dynamic Data Masking, and communication between Dynamic Data Masking and databases
- Specification of two cipher suites for message encryption



```
<protocols type="ArrayList">
    <entry>TLSv1.2</entry>

    <!--
    <entry>TLSv1.1</entry>
    -->
</protocols>
<ciphers type="ArrayList">
    <entry>TLS_RSA_WITH_AES_128_GCM_SHA256</entry>
    <entry>TLS_RSA_WITH_AES_128_CBC_SHA256</entry>
</ciphers>
```

In this example you have manually disabled the TLSv1.1 protocol, because TLSv1.1 does not support the specified ciphers suites. Otherwise, Dynamic Data Masking disables protocols that do not support any of the listed ciphers.

When you start Dynamic Data Masking, it logs a warning about unsupported protocols and automatically disabled protocols. To print a list of global protocols and cipher suites that Dynamic Data Masking uses at run-time, run the following server network commands:

- `server network protocols`
- `server network ciphers`

To print a list of all available protocols and cipher suites enabled in the Java Virtual Machine on which Dynamic Data Masking runs, run the following server network commands:

- `server network protocols available`
- `server network ciphers available`

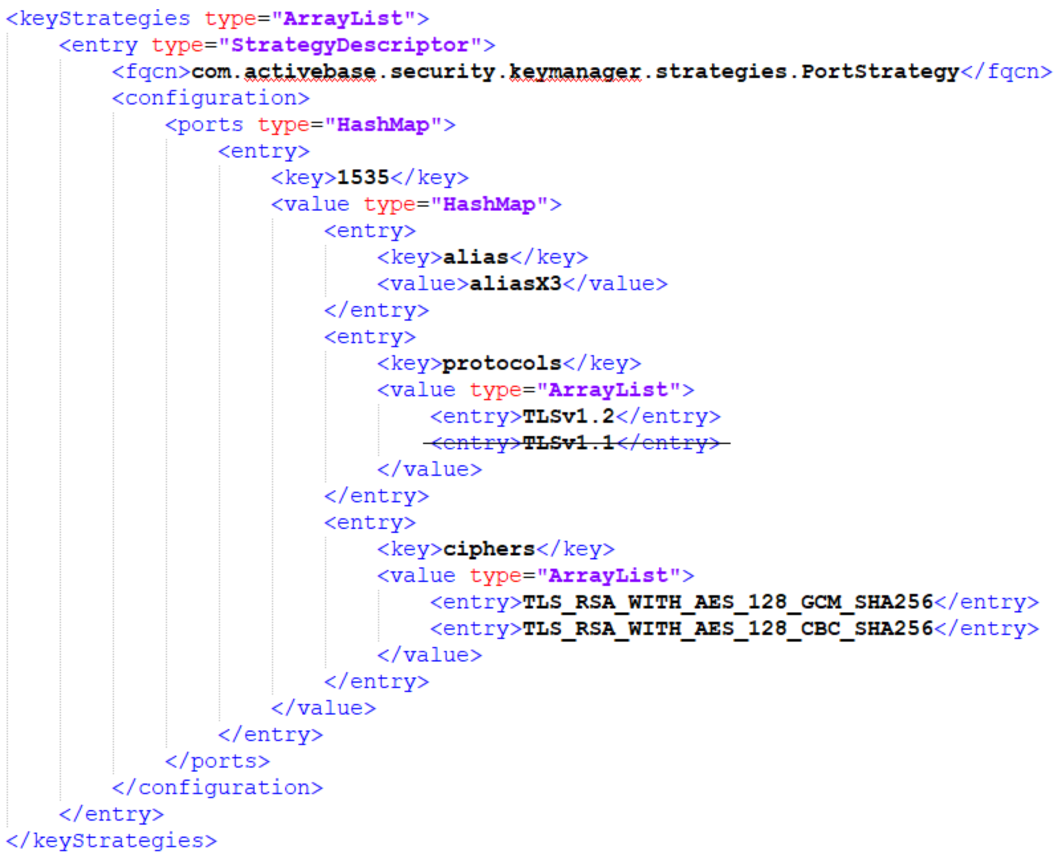
The protocols and ciphers that you configure must be a subset of the available protocols and ciphers.

If a database client does not support protocols and ciphers that you configure in the `cfg/ddm.security` file, the client cannot connect to Dynamic Data Masking or any databases through Dynamic Data Masking. In this case, you must configure the client software with the appropriate protocols and ciphers.

Port Configuration

You can configure an advanced port strategy that maps SSL-enabled host and port values to an alias in Dynamic Data Masking, in addition to optional security protocol and cipher suites. Configuring a port strategy gives you the ability to fine-tune network security with the clients that request a connection to specific SSL-enabled ports in Dynamic Data Masking. The port strategy might disable protocols that do not support any of the specified cipher suites.

The following image shows an example of advanced port configuration in the `cfg/ddm.security` file:



```
<keyStrategies type="ArrayList">
    <entry type="StrategyDescriptor">
        <fqcn>com.activebase.security.keymanager.strategies.PortStrategy</fqcn>
        <configuration>
            <ports type="HashMap">
                <entry>
                    <key>1535</key>
                    <value type="HashMap">
                        <entry>
                            <key>alias</key>
                            <value>aliasX3</value>
                        </entry>
                        <entry>
                            <key>protocols</key>
                            <value type="ArrayList">
                                <entry>TLSv1.2</entry>
                                ~~<entry>TLSv1.1</entry>~~
                            </value>
                        </entry>
                        <entry>
                            <key>ciphers</key>
                            <value type="ArrayList">
                                <entry>TLS_RSA_WITH_AES_128_GCM_SHA256</entry>
                                <entry>TLS_RSA_WITH_AES_128_CBC_SHA256</entry>
                            </value>
                        </entry>
                    </value>
                </entry>
            </ports>
        </configuration>
    </entry>
</keyStrategies>
```

In this example, TLSv1.2 communication is allowed only on the port 1535, and two cipher suites are defined.

You can configure as many strategies as required in the `cfg/ddm.security` file. However, do not specify each host and port more than once. Dynamic Data Masking uses the first found entry with a matching [host:]port entry to configure the SSL port and find the matching signed certificate to perform the SSL handshake with the client.

Protocols and cipher suites defined in an advanced port strategy might override any global settings. The following table shows example configuration settings and the corresponding run-time behavior:

Scenario	**Configured Protocols**		**Runtime Protocols**	
	Global	Strategy on Port 1535	Port to Communicate With	Port to Communicate With
			Client (1535)	Database
1	TLSv1.2 TLSv1.1	TLSv1.2	TLSv1.2	TLSv1.2 TLSv1.1

Scenario	Configured Protocols		Runtime Protocols	
2	TLSv1.2	TLSv1.2 TLSv1.1	TLSv1.2 TLSv1.1	TLSv1.2
3	TLSv1.2 TLSv1.1	Not set	TLSv1.2 TLSv1.1	TLSv1.2 TLSv1.1
4	Not set	TLSv1.2 TLSv1.1	TLSv1.2 TLSv1.1	all protocols
5	Not set	Not set	all protocols	all protocols

To print a list of configured protocols and cipher suites for a specific SSL host and port, run the following server network commands:

- `server network protocols [host:]port`
- `server network ciphers [host:]port`

For example:

```
server network protocols 10.40.1.172:1535

server network ciphers 1535
```

Java Security Protocol Configuration

By default, TLSv1, TLSv1.1, and TLSv1.2 algorithms are disabled in the JRE version included with Dynamic Data Masking. You can choose to enable the algorithms if required.

To enable TLSv1, TLSv1.1, and TLSv1.2 perform the following tasks:

1. Open the following file: `<Dynamic Data Masking installation directory>/jre/lib/security/java.security`
2. Search the file content for the `jdk.tls.disabledAlgorithms` property.
3. Delete the TLSv1 and TLSv1.1 values from the list of values.
4. Save the changes to the file.

Note: Enabling older TLS versions reduces the overall security of the product.

For more information about JRE algorithms, see the Azul JDK support website.

Reloading the Security Configuration

The Dynamic Data Masking Server loads the security configuration once during startup.

If you make changes to the `cfg/ddm.security` file, run the following command to re-load the security configuration without restarting the Dynamic Data Masking Server:

```
server reload security
```

Example SSL Configuration Process

This example demonstrates the process of enabling and configuring SSL communication in Dynamic Data Masking using a custom keystore and truststore.

1. Stop the Dynamic Data Masking Server.
2. To convert the Dynamic Data Masking Server port from clear to SSL, run the following command: `server network edit 8195;SSL`
3. Use the keytool command to add a private key and signed certificate to a new keystore. For example, create a self-signed certificate with a 2048-bit RSA key pair, that is valid for 2555 days, under the specified alias in the keystore:

   ```
   keytool -genkey -trustcacerts -alias myHost -validity 2555 -keyalg RSA -keysize 2048
   -keystore keyStore.jceks -storetype JCEKS -storepass keystorePass -dname cn= myHost
   ```
4. Export the signed certificate from the keystore to a file. For example, export a binary DER-encoded certificate from the keystore to the file `myHost.der`:

   ```
   keytool -exportcert -noprompt -alias myHost -file myHost.der -keystore
   keyStore.jceks -storetype JCEKS -storepass keystorePass
   ```
5. Import the certificate file to the truststore. For example, import the certificate from file `myHost.der` under the specified alias myHost:

   ```
   keytool -importcert -noprompt -trustcacerts -alias myHost -file myHost.der -keystore
   trustStore.jceks -storetype JCEKS -storepass truststorePass
   ```
6. Copy the keystore to the Dynamic Data Masking Server. For example, run the following command:

   ```
   copy keyStore.jceks C:\DDM\cfg\
   ```
7. Copy the truststore to the Management Console. For example, run the following command:

   ```
   copy trustStore.jceks C:\DDM\cfg\
   ```
8. Use one of the following files to configure the keystore in the Dynamic Data Masking Server:
   - `cfg/ddm.security`
   - `jvm.params`
9. In the `cfg/client.security` file, configure the truststore you copied to the Management Console.
10. Start the Dynamic Data Masking Server.
11. Login to the Dynamic Data Masking Server from the Management Console.
    - Select **SSL** in the **Secure Layer** menu.
    - To connect in Dynamic Data Masking versions before 9.8.4, select **None** in the **Secure Layer** menu.

    If the handshake with the Dynamic Data Masking Server fails, the command line tool exits with "error code 1" and prints an error message. Check the security configuration files and try again.
12. To run commands in the local or remote Dynamic Data Masking Server, specify the SSL argument in the connection parameter. For Dynamic Data Masking versions before 9.8.4, do not specify the SSL argument in the connection parameter. For example, in version 9.8.4, run the following command to check the server status:

    -
      ```
      server status
      server service add "DDM for ORACLE" -targets admin/admin@127.0.0.1:8195;SSL
      ```
    - Run the following command to check the server status in Dynamic Data Masking versions before 9.8.4:

      ```
      server status server service add "DDM for ORACLE" -targets admin/
      admin@127.0.0.1:8195
      ```

Kerberos Authentication for Hive or Impala Databases

You can enable Kerberos authentication for Hive, Hive (HTTP), and Impala databases. Copy the `krb5.conf` file and the keytab file for the Dynamic Data Masking service principal to the Dynamic Data Masking Server machine. Then configure the `ddm.security` file.

For more information on Kerberos authentication for Hive and Impala, see the H2L "*Enabling Kerberos for Hive and Impala Databases in Dynamic Data Masking*."

1. Stop the Dynamic Data Masking Server.
2. Copy the `krb5.conf` file and the keytab file for the Dynamic Data Masking service principal to the Dynamic Data Masking Server machine.
3. If you have not already created an XML `ddm.security` configuration file, create the file in the following location: `<DDM>/cfg/ddm.security`
4. Configure the `ddm.security` file as shown in the example below:

```
<XML>
      <kdc>/etc/krb5.conf</kdc>

    <jaasConfig type="ArrayList">
        <entry type="JaasDescriptor">
            <fqcn>com.activebase.security.jaas.JaasProcessorImpl</fqcn>
            <configuration>
                <jaasEntries type="HashMap">
                    <entry>
                        <key>default</key>
                        <value type="ArrayList">
                            <entry type="HashMap">
                                <entry>
                                    <key>moduleClass</key>
                                    <value>com.sun.security.auth.module.Krb5LoginModule</value>
                                </entry>
                                <entry>
                                    <key>moduleFlag</key>
                                    <value>required</value>
                                </entry>
                                <entry>
                                    <key>options</key>
                                    <value type="HashMap">
                                        <entry>
                                            <key>principal</key>
                                            <value>ddmserver/ddmhost@realm.com</value>
                                        </entry>
                                        <entry>
                                            <key>keyTab</key>
                                            <value>cfg/ddmService.keytab</value>
                                        </entry>
                                        </value>
                                    </entry>
                                </entry>
                                </value>
                            </entry>
                        </jaasEntries>
                    </configuration>
                        </entry>
                    </jaasConfig>

                    </XML>
```

5. Configure the following values in the `ddm.security` file:
   - For the value of the principal key, specify the Dynamic Data Masking service principle name (SPN).
   - For the value of the keyTab key, specify the keytab file name with path.

If SSL authentication is enabled on the Hive or Impala server, follow the steps in "Keystore Configuration" on page 31 and "Truststore Configuration" on page 33 to enable SSL for Hive or Impala services.

Configuring Kerberos Encryption for Hive Databases

You can configure Kerberos encryption for Hive databases. Copy the `krb5.conf` file and the keytab file for the Dynamic Data Masking service principal to the Dynamic Data Masking Server machine. Then configure the `ddm.security` file.

For information about enabling Kerberos for Hive, see the H2L "*Enabling Kerberos for Hive and Impala Databases in Dynamic Data Masking*."

1. Stop the Dynamic Data Masking Server.
2. On Windows, remove the Dynamic Data Masking Service.
3. Copy the `krb5.conf` file and the keytab file for the Dynamic Data Masking service principal to the Dynamic Data Masking Server machine.
4. If you have not already created an XML `ddm.security` configuration file, create the file in the following location: `<DDM>/cfg/ddm.security`
5. Configure the `ddm.security` configuration file. For more information, see "Configuring the ddm.security file" on page 44.
6. Start the Dynamic Data Masking Server.

   Note: Add the port that you configured in the `ddm.security` file to the service. In the example, the `ddm.security` file is configured for the Dynamic Data Masking for Hive service running on port 10081. In this case, you would add port 10081 to the Dynamic Data Masking for Hive service.
7. Configure the JDBC or ODBC client. For more information, see "Client Configuration" on page 45.

Configuring the ddm.security file

To configure Kerberos encryption for Hive databases, you must specify the following values in the `ddm.security` file.

1. For the value of the principal key, specify the Dynamic Data Masking SPN.
   For example: `ddmserver/ddmhost@realm.com`
2. For the value of the keyTab key, specify the keytab file name with path.
   For example: `cfg/ddmService.keytab`
3. If Hive is configured with auth-conf, define the port strategy for the service:
   - Key is the port number of Dynamic Data Masking for the Hive service.
   - Value is a map with the key as `sasl.qop` and the value as `auth-conf`.

   For example, the following `ddm.security` file is configured for the Dynamic Data Masking for Hive service running on port 10001:

```
<XML>
 <kdc>/etc/krb5.conf</kdc>
 <jaasConfig type="ArrayList">
    <entry type="JaasDescriptor">
        <fqcn>com.activebase.security.jaas.JaasProcessorImpl</fqcn>
        <configuration>
            <jaasEntries type="HashMap">
                <entry>
                    <value type="ArrayList">
                        <entry type="HashMap">
                            <entry>
                                <value>com.sun.security.auth.module.Krb5LoginModule</value>
                                <key>moduleClass</key>
                            </entry>
                            <entry>
                                <value>required</value>
                                <key>moduleFlag</key>
                            </entry>
                            <entry>
```

```
                                <value type="HashMap">
                                    <entry>
                                        <value>ddmserver/ddmhost@realm.com</value>
                                        <key>principal</key>
                                    </entry>
                                    <entry>
                                        <value>cfg/ddmService.keytab</value>
                                        <key>keyTab</key>
                                    </entry>
                                </value>
                                <key>options</key>
                            </entry>
                        </entry>
                    </value>
                    <key>default</key>
                </entry>
            </jaasEntries>
        </configuration>
    </entry>
</jaasConfig>
<serviceStrategies type="ArrayList">
    <entry type="StrategyDescriptor">
        <fqcn>com.activebase.security.service.strategies.PortStrategy</fqcn>
        <configuration>
            <ports type="HashMap">
                <entry>
                    <key>10001</key>
                    <value type="HashMap">
                        <entry>
                            <key>sasl.qop</key>
                            <value>auth-conf</value>
                        </entry>
                    </value>
                </entry>
            </ports>
        </configuration>
    </entry>
</serviceStrategies>
</XML>
```

Client Configuration

After you configure the `ddm.security` file, configure the client.

JDBC Client Configuration

The Dynamic Data Masking host and port in the client JDBC URL must point to the Dynamic Data Masking Hive service host and port. The principal in the client JDBC URL must be the Dynamic Data Masking Server principal name.

You can connect to the client using a direct connection or you can connect to the client through Dynamic Data Masking. In the client connection URL, add the property `SaslQOP` using one of the following formats:

JDBC URL (Direct Connection)

If you are connecting to the client directly, use the following format:

```
jdbc:hive2://<HiveServer2 Host>:<HiveServer2 Port>/;principal=<Server Principal of Hive
>;SaslQOP=auth-conf;
```

JDBC URL (Connection through Dynamic Data Masking)

If you are connecting to the client through Dynamic Data Masking, use the following format:

```
jdbc:hive2://<DDM Server Host>:<DDM Service Port configured in
ddm.secuirty>/;principal=<Server Principal of DDM>;SaslQOP=auth-conf;
```

ODBC Client Configuration

Configure the ODBC client through the Windows ODBC Data Source Administrator and create the ODBC data source. When you create the ODBC Data Source, select **Thrift Transport** > **SASL**.

JAAS Configuration Options in the ddm.security File

When you configure the `cfg/ddm.security` for Kerberos authentication, SSL authentication, or for use with a custom security provider, you use Java Authentication and Authorization Service configuration options.

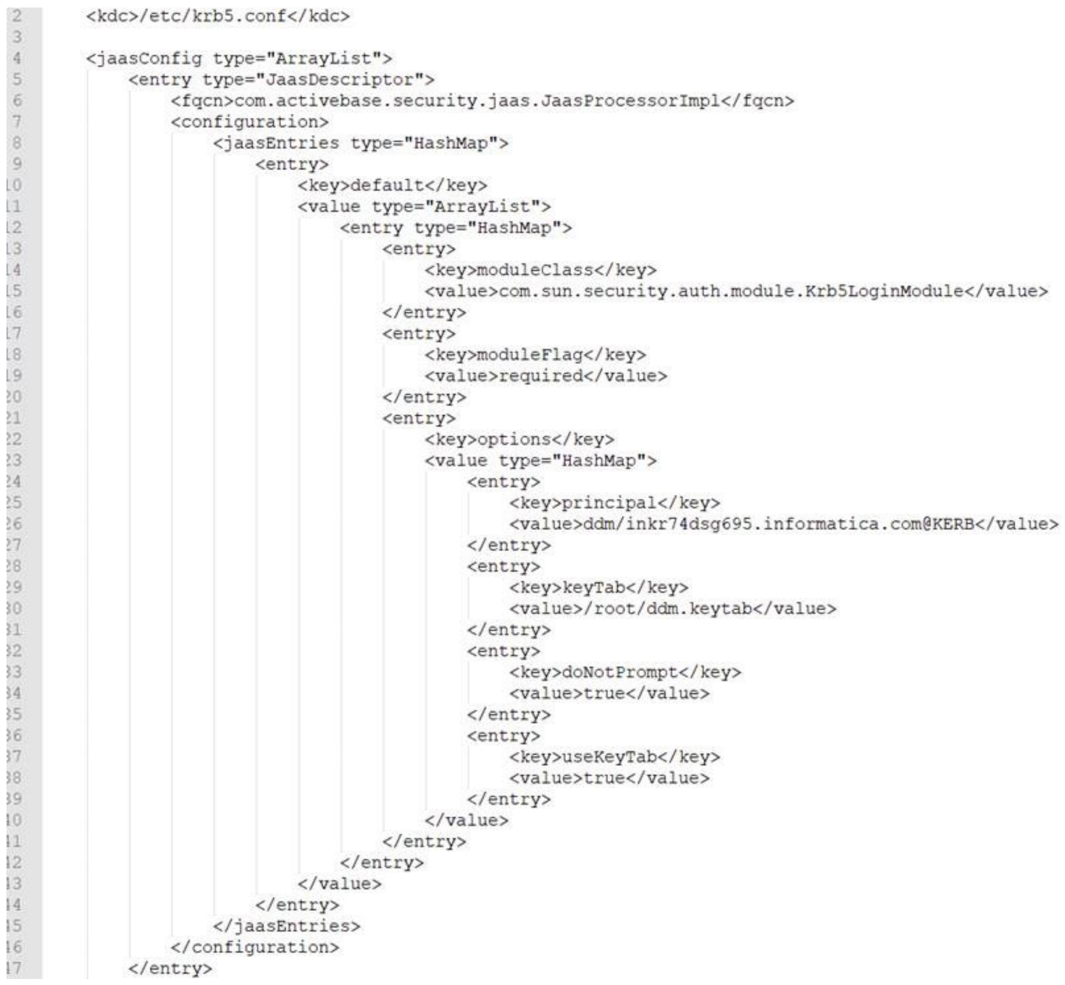
The following table describes the JAAS parameters supported in the `cfg/ddm.security` file:

Option	Description
KDC	The path to the configuration file that provides details of the Kerberos key distribution center.
jaasConfig	The start of JAAS configuration.
JaasDescriptor	The JAAS descriptor that provides JAAS processor and configuration entries for the processor.
fqcn	The full-qualified class name of the JAAS processor implementation. Dynamic Data Masking provides one implementation: com.activebase.security.jaas.JaasProcessorImpl
jaasEntries	The map of JAAS configuration entries.
entry	The JAAS configuration entry.
key	The name of the configuration entry. Dynamic Data Masking supports one mandatory entry called "default."
value	The list that contains the map of login modules and their configuration parameters.
entry	The configuration entry of the login module.
moduleClass	The login module class tag.
value (moduleClass)	The fully-qualified class name of the login module implementation. Dynamic Data Masking supports two implementations: 1. com.sun.security.auth.module.Krb5LoginModule 2. com.ibm.security.auth.module.Krb5LoginModule
moduleFlag	The login module flag tag.

Option	Description
value (moduleFlag)	Java supports the following standard options: REQUIRED, REQUISITE, SUFFICIENT, and OPTIONAL.
options	The configuration options of the login module as a map of the *option name* and the corresponding*value*. Dynamic Data Masking supports the following configuration options of the com.sun.security.auth.module.Krb5LoginModule login module: - principal - keyTab - useKeyTab - storeKey - doNotPrompt - isInitiator - useTicketCache - refreshKrb5Config - renewTGT - storePass - clearPass - useFirstPass - debug For more information on these options, refer to the following Oracle documentation: https://docs.oracle.com/javase/8/docs/jre/api/security/jaas/spec/com/sun/security/auth/module/Krb5LoginModule.html Dynamic Data Masking supports the following configuration options of the com.ibm.security.auth.module.Krb5LoginModule login module: - KRB5CCNAME - principal - UseDefaultCcache - ticketcache - credsType - both For more information on these options, refer to the following IBM documentation: https://www.ibm.com/docs/en/sdk-java-technology/7.1?topic=guide-jaas-login https://www.ibm.com/docs/en/sdk-java-technology/7.1?topic=guide-credential-caches

Sample ddm.security File

The following image is an example of the `cfg/ddm.security` file configured with the JAAS entries listed in the table above:



```
<kdc>/etc/krb5.conf</kdc>

<jaasConfig type="ArrayList">
    <entry type="JaasDescriptor">
        <fqcn>com.activebase.security.jaas.JaasProcessorImpl</fqcn>
        <configuration>
            <jaasEntries type="HashMap">
                <entry>
                    <key>default</key>
                    <value type="ArrayList">
                        <entry type="HashMap">
                            <entry>
                                <key>moduleClass</key>
                                <value>com.sun.security.auth.module.Krb5LoginModule</value>
                            </entry>
                            <entry>
                                <key>moduleFlag</key>
                                <value>required</value>
                            </entry>
                            <entry>
                                <key>options</key>
                                <value type="HashMap">
                                    <entry>
                                        <key>principal</key>
                                        <value>ddm/inkr74dsg695.informatica.com@KERB</value>
                                    </entry>
                                    <entry>
                                        <key>keyTab</key>
                                        <value>/root/ddm.keytab</value>
                                    </entry>
                                    <entry>
                                        <key>doNotPrompt</key>
                                        <value>true</value>
                                    </entry>
                                    <entry>
                                        <key>useKeyTab</key>
                                        <value>true</value>
                                    </entry>
                                </value>
                            </entry>
                        </entry>
                    </value>
                </entry>
            </jaasEntries>
        </configuration>
    </entry>
```

CHAPTER 4

Connection Management

This chapter includes the following topics:

Connection Management Overview

Use the **Add Database** window to add a database to the Management Console tree. Select a database type and define database parameters. Test the database connection to verify that the Dynamic Data Masking service can access the database.

Configuring the Target Database

Define the database parameters that the Dynamic Data Masking service uses to connect to the target database. The target database configuration specifies the database, user account, and connection settings.

Dynamic Data Masking uses the target database to retrieve metadata. For security rules that include column masking for `SELECT *` statements, Dynamic Data Masking first retrieves the list of columns. If the target database connection is not valid or if the DDM admin user credentials are not active, Dynamic Data Masking cannot retrieve the list of columns to apply the security rule. In such cases, Dynamic Data Masking generates and sends the following error to the client: `Required credentials are either invalid or missing. Contact your database administrator for assistance.`

The error ensures that unmasked data does not appear on the database client.

1. In the Management Console, click **Tree** > **Add Database**.

   The **Add Database** window appears.
2. Select the database type.
3. In the **DDM Database Name** field, enter a name for the database connection you are creating.
4. From the **Key Store** menu, select either the **Default** or **Custom** keystore option and enter the database credentials or keystore name and alias.

   The settings differ based on the type of keystore that you select.
5. Click **Test Connection** to validate the connection to the database.
6. Click **OK**.

Testing a Connection

When you test a connection, the Dynamic Data Masking service validates the database configuration and connection information.

Define all connection parameters before you test a database connection. If the Dynamic Data Masking service cannot connect to the database, check with the database administrator to verify the database configuration information.

1. Select a database node in the Management Console tree and click **Tree** > **Edit**.

   The **Edit** window appears.
2. Click **Test Connection**.

   The Management Console sends a request with the database object to the Dynamic Data Masking Server. The Dynamic Data Masking Server reads the database object from the request, and tests connection to that database. A confirmation message appears if the connection is valid. If the connection fails, an error message appears.

Data Vault Connection Management

To add an Informatica Data Vault connection node to the Management Console tree, select the FAS database type.

You can enable SSL communication between the Data Vault and the Dynamic Data Masking Server. For more information about enabling SSL communication, see the "Security" chapter. Communication is encrypted between the Dynamic Data Masking FAS service and the Data Vault, and between the Dynamic Data Masking FAS service and the Data Vault client. For information on how to enable SSL in the Data Vault, see the *Data Vault Administrator Guide*. To use SSL communication, you must have Data Archive version 6.4.3 or later installed.

Click **Test Connection** to verify that the Dynamic Data Masking service can access the database.

Data Vault Connection Parameters

Define the following connection parameters for a Data Vault connection:

DDM Database Name

Name for the database node that appears in the Management Console tree.

Server Address

Server host name or TCP/IP address for the Data Vault.

Note: Verify that there is no firewall that prohibits the Dynamic Data Masking Server from connecting to the Data Vault Server and port number.

Server Port

TCP/IP listener port for the Data Vault.

FAS Database Name

Database name for the Data Vault.

Keystore

Select **Custom** if you have configured a custom keystore. Select **Default** if you want to use the default keystore preconfigured for use with Dynamic Data Masking.

DBA Username

User name for the Data Vault user account to log in to the Data Vault. The Data Vault user must have the SELECT access privilege for all the tables to which the client user has the SELECT access privilege. This parameter is valid for the default keystore.

DBA Password

Password for the Data Vault user. This parameter is valid for the default keystore.

Key Store Name

Name of the custom keystore, defined in the `ddm.security` file. This parameter is valid for custom keystores.

Alias

Alias name for the custom keystore. For CyberArk accounts, the alias name was defined when the CyberArk account was created. This parameter is valid for custom keystores.

SSL

Select to enable SSL communication between the database and the Dynamic Data Masking Server. For more information on SSL configuration, see the chapter "Security."

Generic Database Connection Management

Select the Generic Database type to add a Generic Database connection node to the Management Console. Use the DDM for JDBC and DDM for ODBC services and a Generic Database node to mask data for a database that uses JDBC or ODBC connectivity.

You can use the Generic Database node for databases that do not have a dedicated database node. For example, use the Generic Database node with a Netezza or Greenplum database.

You provide information about the database and the database drivers. The client sends the request to Dynamic Data Masking. Dynamic Data Masking connects to the database to retrieve metadata in order to mask the SQL statement. Dynamic Data Masking masks the request and sends the masked request back to the client. The client then sends the masked request to the database. The database returns masked data.

Create a connection rule that uses the Switch to Database rule action to define the target database. Specify a database in the rule that corresponds to the Dynamic Data Masking Database Name parameter.

Click **Test Connection** to verify that the Dynamic Data Masking service can access the database.

Generic Database Connection Parameters

Define the following connection parameters for a database that uses an ODBC or JDBC connection:

DDM Database Name

Name for the database in the Management Console tree.

Driver Class Name

Fully qualified class name of the target database driver.

For example, you might enter the following text:

```
org.netezza.Driver
```

Connect String (URL)

JDBC connection string used to connect to the database.

For example, you might enter the following text:

```
jdbc:netezza://hostname:port/database_name
```

Optional Parameters

Optional parameters for the Informatica driver for the database.

DSN Name

Logical data source name used to connect to the database.

Unary Table

Name of the unary table for the database.

Bind Argument Representation

Representation of the bind argument for a PL/SQL stored function.

Supports ANSI Join

Select if the database supports ANSI join syntax.

Are Function Calls Allowed in Select

Select if the database allows a function call to be included in a SELECT statement.

Are Parentheses Allowed for Function Call with No Arguments

Select if the database allows parentheses in a function call that does not have any arguments.

Command to Get Data for Impersonation

The SQL command that Dynamic Data Masking uses to retrieve the data required for impersonation.

For example, you might enter the following command to retrieve the search_path for a Greenplum database:

```
SELECT COALESCE(substring( useconfig[1] from  '%=#"%#"' for '#' ),
substring( datconfig[1] from  '%=#"%#"' for '#'), 'PUBLIC') as AUTH_CURRENT_SCHEMA
from pg_user pguser,   pg_database pgdatabase where pguser.usename= '\
(AUTH_USERNAME)'   and  pgdatabase.datname= '\(AUTH_CATALOG)'
```

Execute Command to get Impersonation Data for Every Request

Select to indicate that Dynamic Data Masking must retrieve the data for impersonation for each request. If the check box is unchecked, Dynamic Data Masking retrieves the data once per session.

Impersonation Commands

Impersonation commands for the database, separated by a semicolon (;) and a line break. Words that are preceded by a backslash and left parenthesis and followed by a right parenthesis are Dynamic Data Masking symbols that Dynamic Data Masking replaces with symbol values. For example, \(SYMBOL).

Note: For a Greenplum database, you must set the AUTH_CURRENT_SCHEMA symbol. You can use the following command to set the symbol:

```
SET SEARCH_PATH = \(AUTH_CURRENT_SCHEMA)
```

Cleanup Commands

Cleanup commands for the database, separated by a semicolon (;) and a line break.

Sanity Check Script

Sanity check script to verify that the Dynamic Data Masking connection to the database is valid.

Keystore

Select custom if you have configured a custom keystore. Select default if you want to use the default keystore preconfigured for use with Dynamic Data Masking.

DBA Username

Username for the database user account to log in to the database. The database user must be a privileged user that has SELECT access to all the tables that the client user has SELECT access to.

DBA Password

Password for the database user.

Key Store Name

Name of the custom keystore, defined in the `ddm.security` file. This parameter is valid for custom keystores.

Alias

Alias name for the custom keystore. For CyberArk accounts, the alias name was defined during creation of the CyberArk account. This parameter is valid for custom keystores.

Greenplum Connections

Configure database privileges and impersonation and cleanup commands for a Greenplum database.

The required user privileges and commands for impersonation and cleanup vary based on the database and database version. The following privileges and commands are examples that work for some versions of the database.

Impersonation Commands

You might enter the following commands in the **Impersonation Commands** text box for a Greenplum database:

```
SET ROLE \(AUTH_USERNAME)
SET SEARCH_PATH = \(AUTH_CURRENT_SCHEMA)
```

Required Database Privileges

The Dynamic Data Masking administrator must have the following privilege:

```
GRANT <client_username> TO <ddm_username>
```

The database user that Dynamic Data Masking impersonates must have the following privilege:

```
GRANT SELECT ON TABLE <tablename> TO <client_username>;
```

Cleanup Commands

You might enter the following commands in the **Cleanup Commands** text box for a Greenplum database:

```
RESET ROLE;
RESET SEARCH_PATH;
```

MySQL Connections

Configure database privileges and impersonation and cleanup commands for a MySQL database.

The required user privileges and commands for impersonation and cleanup vary based on the database and database version. The following privileges and commands are examples that work for some versions of the database.

Impersonation Commands

You might enter the following command in the **Impersonation Commands** text box for a MySQL database:

```
USE \(AUTH_CATALOG)
```

Cleanup Commands

You might enter the following command in the **Cleanup Commands** text box for a MySQL database:

```
USE \(ADMIN_CATALOG)
```

Netezza Connections

Configure database privileges and impersonation and cleanup commands for a Netezza database.

The required user privileges and commands for impersonation and cleanup vary based on the database and database version. The following privileges and commands are examples that work for some versions of the database.

Impersonation Commands

You might enter the following commands in the **Impersonation Commands** text box for a Netezza database:

```
SET CATALOG \(AUTH_CATALOG)
SET SCHEMA \(AUTH_CURRENT_SCHEMA)
EXECUTE AS \(AUTH_USERNAME)
```

Required Database Privileges

The Dynamic Data Masking administrator must have the following privileges:

```
GRANT CONNECT ON <database_name> TYPE DATABASE  TO <ddm_username>;
GRANT CONNECT ON <schemaname> TYPE SCHEMA TO <ddm_username>;
GRANT EXECUTE AS ON <client_username> TYPE USER TO <ddm_username>
```

Cleanup Commands

You might enter the following commands in the **Cleanup Commands** text box for a Netezza database:

```
SET CATALOG \(ADMIN_CATALOG)
REVERT
```

Predefined Security Rule Sets

You can use predefined security rule sets with the generic database node if you want to run commands that alter the user context. The rule set captures the commands and updates the Dynamic Data Masking symbols for the session.

You must configure Dynamic Data Masking to direct the database request to the predefined rule set before the request goes to the user-defined masking rules.

Dynamic Data Masking includes predefined rule sets for the following databases:

Greenplum

You can find the predefined security rule set for a Greenplum database in the following location:

```
<Dynamic Data Masking installation>\Wrappers\ImpersonationRules\GreenplumRS.xml
```

MySQL

You can find the predefined security rule set for a MySQL database in the following location:

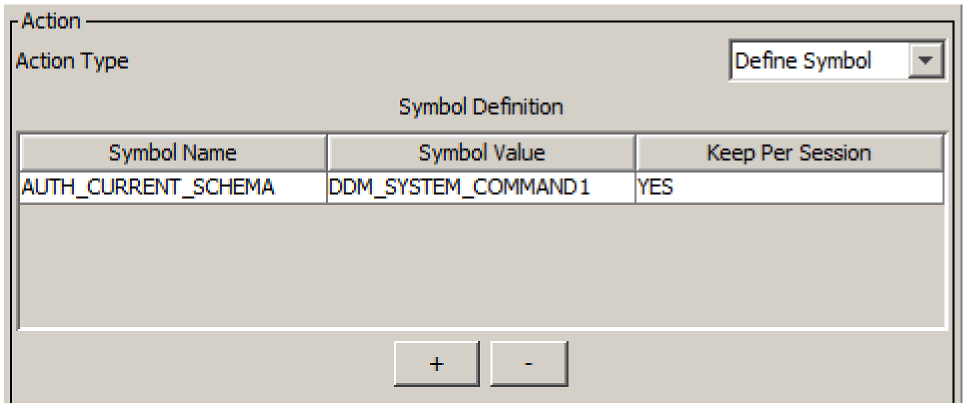
```
<Dynamic Data Masking installation>\Wrappers\ImpersonationRules\MySQL.xml
```

Netezza

You can find the predefined security rule set for a Netezza database in the following location:

```
<Dynamic Data Masking installation>\Wrappers\ImpersonationRules\Netezza.xml
```

If you execute commands that alter the user context, you might want Dynamic Data Masking to skip one or more of the impersonation commands. To skip an impersonation command, create a security rule that sets the symbol value to DDM_SYSTEM_COMMAND1. Dynamic Data Masking skips, and does not execute, any impersonation command that uses a symbol whose value is DDM_SYSTEM_COMMAND1. For example, on a Netezza database, if you want to skip the command that uses the AUTH_CURRENT_SCHEMA symbol, you would create a rule with the following rule action:



Symbol Name	Symbol Value	Keep Per Session
AUTH_CURRENT_SCHEMA	DDM_SYSTEM_COMMAND1	YES

Hive Connection Management

Select the Hive database type to add a Hive database connection node to the Management Console tree.

You can use the Hive database type to access Hadoop-compatible file systems. To connect to multiple Hive databases, create database nodes in the Management Console and enter different sets of driver files in the classpath parameter.

Click **Test Connection** to verify that the Dynamic Data Masking service can access the database.

Hive Connection Parameters

Define the following connection parameters for a Hive database:

DDM Database Name

Name for the database that appears in the Management Console tree.

Hive Database Name

Database name for the Hive database.

Driver Classpath

Classpath of the Hive database driver files on the Dynamic Data Masking Server. Use semicolons to separate multiple classpaths.

You can use an asterisk (*) to indicate all the jar files in a directory. Dynamic Data Masking ignores files in the directory that are not jar files. For example, you might enter the following location for Windows:

```
C:\JDBC_Drivers\Hive\hive\*;C:\JDBC_Drivers\Hive\hadoop\*
```

Driver Class Name

Fully qualified class name of the Impala driver specified in the **Driver Classpath** property.

Connection String (URL)

JDBC connection URL based on the driver.

If the Hive database has Kerberos authentication enabled, the URL properties **auth** and **kerberosAuthType** are mandatory.

For example: jdbc:hive2://hiveserver:10000/default;principal=hive/hiveserver@realm.com;
auth=kerberos;kerberosAuthType=fromSubject

Database Server Principal

When you connect to a Hive database with Kerberos authentication enabled, specify the server principal of Hive, even though it is included in the JDBC URL. Otherwise, leave this parameter blank.

Keystore

Select custom if you have configured a custom keystore. Select default if you want to use the default keystore preconfigured for use with Dynamic Data Masking.

DBA Username

Optional user name for the database user account to log in to the Hive database. The database user must be a privileged user that has SELECT access to all the tables that the client user has SELECT access to. This parameter is valid for the default keystore.

Leave this parameter blank if the Hive database has Kerberos authentication enabled.

DBA Password

Optional password for the database user. This parameter is valid for the default keystore.

Leave this parameter blank if the Hive database has Kerberos authentication enabled.

Key Store Name

Name of the custom keystore, defined in the `ddm.security` file. This parameter is valid for custom keystores.

Alias

Alias name for the custom keystore. For CyberArk accounts, the alias name was defined during creation of the CyberArk account. This parameter is valid for custom keystores.

IBM Db2 Connection Management

Select the Db2 database type to add an IBM Db2 database connection node to the Management Console tree.

The IBM Db2 connection request does not contain information about the database. You must define the target database that Dynamic Data Masking forwards the request to. Make a connection rule that uses the Switch to Database rule action to define the target database. Specify a database in the rule that corresponds to the Dynamic Data Masking Database Name parameter.

Click **Test Connection** to verify that the Dynamic Data Masking service can access the database.

IBM Db2 Connection Parameters

Define the following connection parameters for an IBM Db2 database:

DDM Database Name

Name for the database in the Management Console tree.

Server Address

Server host name or TCP/IP address for the IBM Db2 database.

Note: Verify that there is no firewall that prohibits the Dynamic Data Masking Server from connecting to the database server and port number.

Server Port

TCP/IP listener port for the IBM Db2 database.

DB2 Database Name

Database name for the IBM Db2 database.

Optional Parameters

Additional parameters for the Informatica driver for IBM Db2.

For example, if the IBM Db2 database is configured with the SERVER_ENCRYPT authentication method, you might enter the following parameter:

```
AuthenticationMethod=encryptedUIDPassword
```

Keystore

Select custom if you have configured a custom keystore. Select default if you want to use the default keystore preconfigured for use with Dynamic Data Masking.

DBA Username

User name for the database user account to log in to the IBM Db2 database. The database user must be a privileged user that has SELECT access to all the tables that the client user has SELECT access to. This parameter is valid for the default keystore.

DBA Password

Password for the database user. This parameter is valid for the default keystore.

Key Store Name

Name of the custom keystore, defined in the `ddm.security` file. This parameter is valid for custom keystores.

Alias

Alias name for the custom keystore. For CyberArk accounts, the alias name was defined during creation of the CyberArk account. This parameter is valid for custom keystores.

SSL

Select to enable SSL communication between the database and the Dynamic Data Masking Server. For more information on SSL configuration, see the chapter "Security."

IBM Db2 Dynamic Data Masking Administrator Required Privileges

The Dynamic Data Masking administrator must have privileges to access sensitive tables and columns.

Use the IBM Db2 Control Center to create a privileged database user, <DDM Admin>, that corresponds to an administrator user on your operating system or a standard user on your operating system.

If <DDM Admin> corresponds to an administrator user on your operating system, you do not need to grant the user additional privileges.

If <DDM Admin> corresponds to a standard user on your operating system, the user must have SYSMON authorization or higher. If you use the encrypted password option, you must also run the following commands:

- `GRANT SELECT ON SYSIBMADM.SNAPAPPL_INFO TO <DDM Admin>`
- `GRANT EXECTUE ON SYSPROC.SNAPAPPL_INFO_V9 TO <DDM Admin>`

If you use the encrypted password option and an ODBC driver, <DDM Admin> must be able to access the SYSIBMADM.SNAPAPPL_INFO database table.

Additional Privilege for SELECT * Statements

If your Dynamic Data Masking security rules need to support column masking on SELECT * statements, you must run the following command:

```
GRANT SETSESSIONUSER ON PUBLIC TO <DDM Admin>
```

Alternatively, you can run the following commands:

- `GRANT SELECT ON <table being queried> TO <DDM Admin>`
- `GRANT EXECUTE ON <user-defined function used in a PL/SQL action or stored program object> TO <DDM Admin>`

Define the CURRENT_SCHEMA Symbol

If you want to query a table with the `set schema` command and without a table prefix, you must create a security rule to define the CURRENT_SCHEMA symbol.

Create a security rule with the following properties to set the current schema:

Matcher

Select the Text matcher. In the Text matcher, select the Regular Expression Identification Method.

Enter the following regular expression in the Text field:

```
\s*SET\s+(CURRENT\s+)?SCHEMA\s*(=)?\s*('|")?(\w+)('|")?(.*)
```

Note: If you use this regular expression, a database request with the following format will fail:

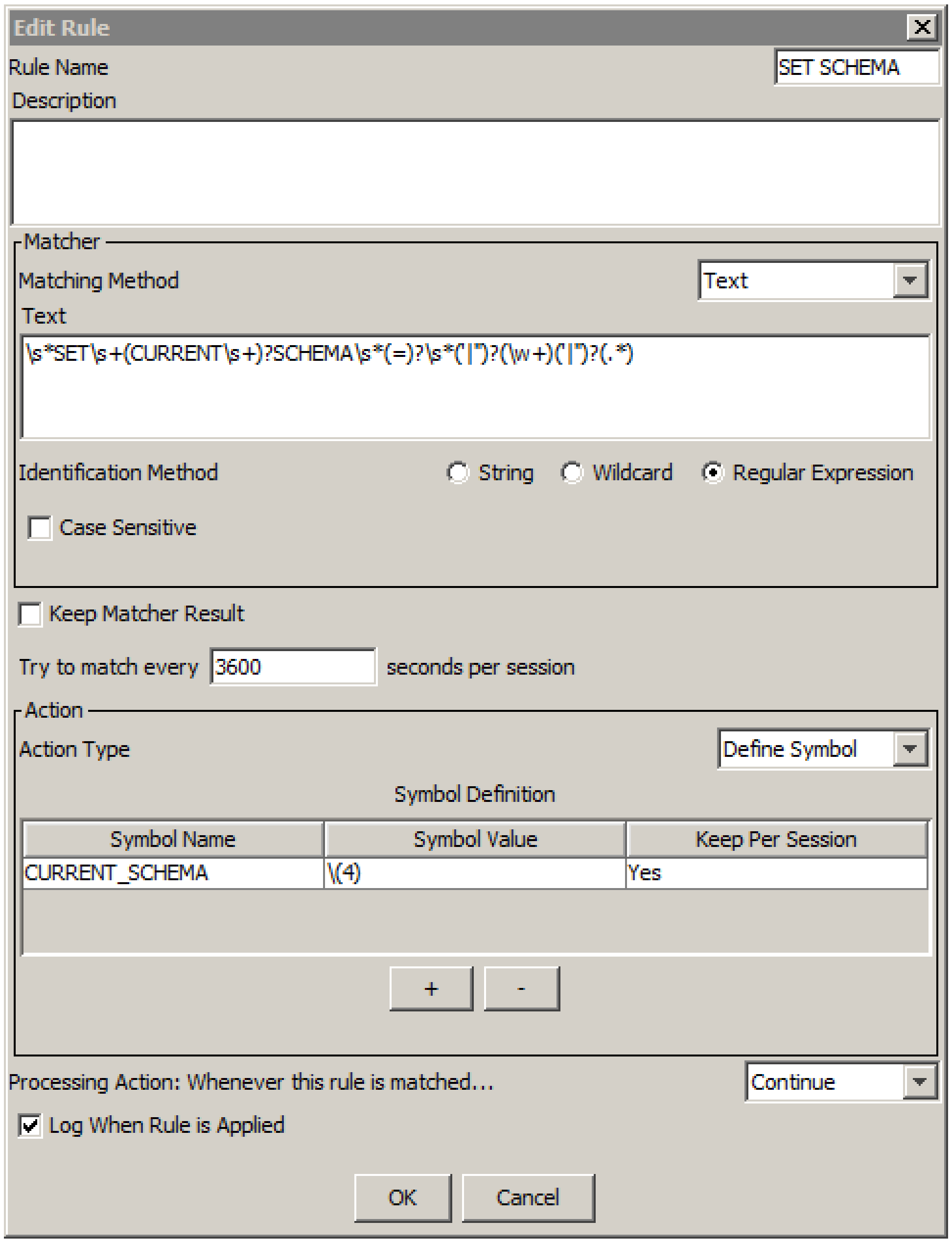
```
SET CURRENT_SCHEMA 'ABC'
```

Rule Action

Select the Define Symbol action. Define the following rule action parameters:

- Symbol Name: CURRENT_SCHEMA
- Symbol Value: \(4)
- Keep Per Session: Yes

The following image shows the rule properties:


Edit Rule

Rule Name: SET SCHEMA

Description:

Matcher

Matching Method: Text

Text:

```
\s*SET\s+(CURRENT\s+)?SCHEMA\s*(=)?\s*('|")?(\w+)('|")?(.*)
```

Identification Method: String Wildcard [✓ Regular Expression]

☐ Case Sensitive

☐ Keep Matcher Result

Try to match every 3600 seconds per session

Action

Action Type: Define Symbol

Symbol Definition

Symbol Name	Symbol Value	Keep Per Session
CURRENT_SCHEMA	\(4)	Yes

Processing Action: Whenever this rule is matched... Continue

☑ Log When Rule is Applied

Impala Connection Management

Select the Impala database type to add an Impala database connection node to the Management Console tree.

You can use the Impala database type to access Hadoop-compatible file systems.

Use **Test Connection** to verify that the Dynamic Data Masking service can access the database.

Impala Connection Parameters

Define the following connection parameters for an Impala database:

DDM Database Name

Name for the database that appears in the Management Console tree.

Impala Database Name

Database name for the Impala database.

Driver Classpath

Classpath of the Impala database driver files on the Dynamic Data Masking Server. Use semicolons to separate multiple classpaths.

You can use an asterisk (*) to indicate all the jar files in a directory. Dynamic Data Masking ignores files in the directory that are not jar files.

Example for Windows:

```
C:\JDBC_Drivers\impala\*
```

Example for Linux:

```
/root/cloudera/impala/*
```

Driver Class Name

Fully qualified class name of the Hive driver specified in the **Driver Classpath** property.

Example:

```
org.apache.hive.jdbc.HiveDriver
```

Connection String (URL)

JDBC connection URL based on the driver.

If the Impala database has Kerberos authentication enabled and the Hive driver is used to connect to Impala, the URL properties **auth** and **kerberosAuthType** are mandatory.

Example: `jdbc:hive2://impalaserver:21050/default;principal=impala/impalaserver@realm.com;auth=kerberos;kerberosAuthType=fromSubject`

Database Server Principal

When you connect to an Impala database with Kerberos authentication enabled, specify the server principal of Impala, even though it is included in the JDBC URL. Otherwise, leave this parameter blank.

Keystore

Select custom if you have configured a custom keystore. Select default if you want to use the default keystore that is preconfigured for use with Dynamic Data Masking.

Keystore Name

Name of the custom keystore that you defined in the `ddm.security` file. This parameter is valid for custom keystores.

Alias

Alias name for the custom keystore. For CyberArk accounts, the alias name was defined when the CyberArk account was created. This parameter is valid for custom keystores.

DBA Username

Optional user name for the database user account to log in to the Impala database. The database user must have SELECT access privileges for all the tables to which the client user has SELECT access privileges. This parameter is valid for the default keystore.

Leave this parameter blank if the Impala database has Kerberos authentication enabled.

DBA Password

Optional password for the database user. This parameter is valid for the default keystore.

Leave this parameter blank if the Impala database has Kerberos authentication enabled.

Informix Connection Management

Select the Informix database type to add an Informix database connection node to the Management Console tree.

The Informix connection request does not contain information about the database. You must define the target database that Dynamic Data Masking forwards the request to. Make a connection rule that uses the Switch to Database rule action to define the target database. Specify a database in the rule that corresponds to the Dynamic Data Masking Database Name parameter.

Click **Test Connection** to verify that the Dynamic Data Masking service can access the database.

Informix Connection Parameters

Define the following connection parameters for an Informix database:

DDM Database Name

Name for the database in the Management Console tree.

Server Address

Server host name or TCP/IP address for the Informix database.

Note: Verify that there is no firewall that prohibits the Dynamic Data Masking Server from connecting to the database server and port number.

Server Port

TCP/IP port of the listener receiving requests in Informix native protocol. The DDM for Informix service uses this port to communicate with the Informix database.

DRDA Port

TCP/IP port of the listener receiving requests in Informix DRDA protocol. The DDM for Informix (DRDA) service uses this port to communicate with the Informix database.

Informix Database Name

Database name for the Informix database.

Informix Server Name

The Informix server name specific to the database instance.

Keystore

Select **Custom** if you have configured a custom keystore. Select **Default** if you want to use the default keystore preconfigured for use with Dynamic Data Masking.

DBA Username

User name for the database user account to log in to the Informix database. The database user must have SELECT access privileges for all the tables to which the client user has SELECT access privileges. This parameter is valid for the default keystore.

DBA Password

Password for the database user. This parameter is valid for the default keystore.

Key Store Name

Name of the custom keystore, defined in the `ddm.security` file. This parameter is valid for custom keystores.

Alias

Alias name for the custom keystore. For CyberArk accounts, the alias name was defined during creation of the CyberArk account. This parameter is valid for custom keystores.

Microsoft Azure SQL Database Connection Management

Select the **Azure SQL Database** type to add a Microsoft Azure SQL Database connection node to the Management Console tree.

Click **Test Connection** to verify that the Dynamic Data Masking service can access the database.

Microsoft Azure SQL Database Connection Properties

Define the following connection parameters for a Microsoft Azure SQL Database:

DDM Database Name

Name for the database in the Management Console tree.

Server Address

Server host name or TCP/IP address for the database.

Note: Verify that there is no firewall that prohibits the Dynamic Data Masking Server from connecting to the database server and port number.

Server Port

TCP/IP listener port for the database.

Optional Parameters

Additional parameters for the Informatica driver for the database.

If you add the AD user as the DDM administrator, set the following property:

```
authenticationMethod=ActiveDirectoryPassword
```

Keystore

Select **Custom** if you have configured a custom keystore. Select **Default** if you want to use the default keystore that is preconfigured for use with Dynamic Data Masking.

DBA Username

User name for the database user account to log in to Microsoft Azure SQL Database. The database user must have SELECT access privileges for all the tables to which the client user has SELECT access privileges. This parameter is valid for the default keystore.

DBA Password

Password for the database user. This parameter is valid for the default keystore.

KeyStore Name

Name of the custom keystore defined in the `ddm.security` file. This parameter is valid for custom keystores.

Alias

Alias name for the custom keystore. For CyberArk accounts, the alias name was defined when the CyberArk account was created. This parameter is valid for custom keystores.

SSL

Select to enable SSL communication between the database and the Dynamic Data Masking Server. For more information on SSL configuration, see Chapter 3, "Security" on page 25.

Microsoft Azure SQL Database Dynamic Data Masking Administrator Required Privileges

The Dynamic Data Masking administrator must have privileges to access sensitive tables and columns.

If you have both Microsoft SQL Server authentication-based users and Active Directory authentication-based users, then use the Active Directory user as the Dynamic Data Masking administrator.

Note: Dynamic Data Masking allows you to log in to Microsoft Azure SQL Database through the following authentication methods:

- Microsoft SQL Server Authentication
- Active Directory Password

Microsoft SQL Server Authentication-Based User as Administrator

Log in as the system administrator in the master database and run the following command:

```
CREATE LOGIN <DDM_Admin_Login> WITH PASSWORD=<DDM_Admin_Login_password>
```

Log in as the system administrator in the target database and run the following commands:

- `CREATE USER <DDM_Admin_User> FOR LOGIN <DDM_Admin_Login>;`
- `ALTER ROLE db_datareader ADD MEMBER <DDM_Admin_User>;`
- `GRANT VIEW DATABASE STATE TO <DDM_Admin_User>;`

Additional Privileges for SELECT * Statements

If your Dynamic Data Masking security rules need to support column masking on SELECT * statements, you must also run the following command:

```
GRANT CONTROL TO <DDM_Admin_User>
```

Alternatively, you can run the following commands:

- `GRANT ALTER ANY USER TO <DDM_Admin_User>`
- `GRANT IMPERSONATE ON USER :: <DDM_User> TO <DDM_Admin_User>`

Active Directory Authentication-Based User as Administrator

Log in as the Active Directory administrator in the target database and run the following commands:

- `CREATE USER <Azure_Active_Directory_DDM_Admin> FROM EXTERNAL PROVIDER;`
- `ALTER ROLE db_datareader ADD MEMBER <Azure_Active_Directory_DDM_Admin>;`
- `GRANT VIEW DATABASE STATE TO <Azure_Active_Directory_DDM_Admin>`

Additional Privileges for SELECT * Statements

If your Dynamic Data Masking security rules need to support column masking on SELECT * statements, you must also run the following command:

```
GRANT CONTROL TO <Azure_Active_Directory_DDM_Admin>
```

Alternatively, you can run the following commands:

- `GRANT ALTER ANY USER TO <Azure_Active_Directory_DDM_Admin>`
- `GRANT IMPERSONATE ON USER :: <Azure_Active_Directory_DDM_Client> TO <Azure_Active_Directory_DDM_Admin>`

Note: If you have both Microsoft SQL Server authentication-based users and Active Directory authentication-based users, then use the Active Directory user as the Dynamic Data Masking administrator.

Microsoft SQL Server Connection Management

Select the Microsoft SQL Server database type to add a Microsoft SQL Server database connection node to the Management Console tree.

Click **Test Connection** to verify that the Dynamic Data Masking service can access the database.

Microsoft SQL Server Connection Parameters

Define the following connection parameters for a Microsoft SQL Server database:

DDM Database Name

Name for the database in the Management Console tree.

Server Address

Server host name or TCP/IP address for the Microsoft SQL Server database.

Note: Verify that there is no firewall that prohibits the Dynamic Data Masking Server from connecting to the database server and port number.

Server Instance Name

The instance name of the Microsoft SQL Server database.

If the Microsoft SQL Server database is configured to use dynamic port allocation, you can enter the Server Instance Name to identify the listener port. If you enter the instance name, you do not need to enter a Server Port number.

Server Port

TCP/IP listener port for the Microsoft SQL Server database. If you enter the Server Port number, you do not need to enter a Service Instance Name.

Optional Parameters

Additional parameters for the Informatica driver for Microsoft SQL Server.

If the database is SSL enabled, add the following property:

```
CryptoProtocolVersion=TLSv1,TLSv1.1,TLSv1.2
```

If you specify multiple protocols, the driver uses the latest version that the server supports. If the driver does not support any of the specified protocols that the server supports, the connection fails and the driver returns an error.

If you use TLSv1 or TLSv1.1, perform the steps documented in Java Security Protocol Configuration on page 41 to enable the protocols.

Keystore

Select custom if you have configured a custom keystore. Select default if you want to use the default keystore preconfigured for use with Dynamic Data Masking.

DBA Username

User name for the database user account to log in to the Microsoft SQL Server database. The database user must be a privileged user that has SELECT access to all the tables that the client user has SELECT access to. This parameter is valid for the default keystore.

DBA Password

Password for the database user. This parameter is valid for the default keystore.

Key Store Name

Name of the custom keystore, defined in the `ddm.security` file. This parameter is valid for custom keystores.

Alias

Alias name for the custom keystore. For CyberArk accounts, the alias name was defined during creation of the CyberArk account. This parameter is valid for custom keystores.

SSL

Select to enable SSL communication between the database and the Dynamic Data Masking Server. For more information on SSL configuration, see the chapter "Security."

Microsoft SQL Server Dynamic Data Masking Administrator Required Privileges

The Dynamic Data Masking administrator must have privileges to access sensitive tables and columns.

Log in as the system administrator and run the following command:

- `USE master;`
- `CREATE LOGIN <DDM Admin> WITH PASSWORD=<DDM Admin password>, DEFAULT_DATABASE = <default database>;`
- `GRANT CONTROL SERVER TO <DDM Admin>;`
- `USE <default database>;`
- `CREATE USER <database user> FOR LOGIN <DDM Admin>;`

Additional Privileges for SELECT * Statements

If your Dynamic Data Masking security rules need to support column masking on SELECT * statements, you must also run the following command:

```
GRANT CONTROL SERVER TO <DDM Admin>
```

Alternatively, you can run the following commands:

- `USE <client user catalog>`
- `CREATE USER <DDM Admin>`
- `FOR LOGIN <DDM Admin> WITH DEFAULT_SCHEMA=<default schema>`
- `USE master`
- `GRANT VIEW SERVER STATE TO <DDM Admin login>`
- `GRANT IMPERSONATE ON LOGIN :: <client user login> TO <DDM Admin login>`

Netezza Connection Management

You can use a Microsoft SQL Server connection to access a Netezza database. However, when you create a new database node for Netezza, Informatica recommends that you use the Generic Database node.

To use a Microsoft SQL Server connection to access Netezza, you must configure a server link between a Microsoft SQL Server database and the Netezza database. The client tool or application that you use to access the database must support Microsoft SQL Server.

When you send a request to the Microsoft SQL Server database, the request passes through the Dynamic Data Masking Server, which alters the request. The Dynamic Data Masking Server applies masking and auditing rules and uses OpenQuery to direct the request through the Microsoft SQL Server database to the Netezza database. The Netezza database returns masked data through the Microsoft SQL Server database to the Dynamic Data Masking Server.

To connect to a Netezza database, create a Microsoft SQL Server connection node in the Management Console.

Oracle Connection Management

Select the Oracle database type to add an Oracle database connection node to the Management Console tree.

If the Dynamic Data Masking service runs on the Oracle database server, you must switch the Oracle listener to a hidden port. Edit the `listener.ora` file to change the Oracle listener to a port that is not in use. When you change the Oracle listener to a hidden port, applications connect to the Dynamic Data Masking listener port instead of the database.

To route applications to the Dynamic Data Masking listener port, you must edit `tnsnames.ora` to add a database alias to `tnsnames.ora` for the Dynamic Data Masking service. Dynamic Data Masking uses the database alias to listen to incoming connections to the database.

Informatica recommends that you change the Oracle listener to a port that is not the default. If you do not change the port, it is possible for an unauthorized user to edit the connection properties from the client and bypass the Dynamic Data Masking Server. Also, when you change the Oracle listener and configure Dynamic Data Masking to listen on the default Oracle listener, you do not have to edit the client connection properties.

Click **Test Connection** to verify that the Dynamic Data Masking service can access the database. If a database defines multiple instances, **Test Connection** validates each Oracle instance cyclically. The test connection verifies each Oracle instance.

Oracle Connection Parameters

Define the following connection parameters for an Oracle database:

DDM Database Name

Logical name defined for the target database.

Instance Name

Instance name for the target database.

Listener Address

Server host name or TCP/IP address for the target database.

Listener Port

TCP/IP listener port for the target database.

Service Name

Service name for the target database. Dynamic Data Masking determines the target database based on the service name or SID in the client connection request.

Keystore

Select custom if you have configured a custom keystore. Select default if you want to use the default keystore preconfigured for use with Dynamic Data Masking.

DBA Username

User name for the database user account to log in to the Oracle database. The database user must be a privileged user that has SELECT access to all the tables that the client user has SELECT access to. This parameter is valid for the default keystore.

DBA Password

Password for database user. This parameter is valid for the default keystore.

Key Store Name

Name of the custom keystore, defined in the `ddm.security` file. This parameter is valid for custom keystores.

Alias

Alias name for the custom keystore. For CyberArk accounts, the alias name was defined during creation of the CyberArk account. This parameter is valid for custom keystores.

SSL

Select to enable SSL communication between the database and the Dynamic Data Masking Server. For more information on SSL configuration, see the chapter "Security."

Oracle Dynamic Data Masking Administrator Required Privileges

The Dynamic Data Masking administrator must have privileges to access sensitive tables and columns.

Run the following database commands to create a Dynamic Data Masking administrator user and grant the required privileges:

- `CREATE USER <DDM Admin> IDENTIFIED BY <XXXX>`
- `ALTER USER <DDM Admin> QUOTA UNLIMITED ON USERS`
- `GRANT BECOME USER`
- `GRANT CREATE SESSION TO <DDM Admin>`
- `GRANT ALTER SESSION`
- `GRANT SELECT ANY TABLE TO <DDM Admin>`
- `GRANT SELECT ANY DICTIONARY TO <DDM Admin>`
- `GRANT EXECUTE ANY PROCEDURE TO <DDM Admin>`

Additional Privileges for SELECT * Statements

If your Dynamic Data Masking security rules need to support column masking on SELECT * statements, you must also run the following commands:

- `CREATE USER <DDM Admin> IDENTIFIED BY <XXXX>`
- `ALTER USER <DDM Admin> QUOTA UNLIMITED ON USERS`
- `GRANT BECOME USER TO <DDM Admin>`
- `GRANT CREATE SESSION TO <DDM Admin>`
- `GRANT ALTER SESSION TO <DDM Admin>`
- `GRANT SELECT ANY TABLE TO <DDM Admin>`
- `GRANT SELECT ANY DICTIONARY TO <DDM Admin>`
- `GRANT EXECUTE ANY TYPE TO <DDM Admin>`
- `GRANT SELECT_CATALOG_ROLE TO <DDM Admin>`
- `GRANT GRANT ANY OBJECT PRIVILEGE TO <DDM Admin>`
- `GRANT CREATE ANY TABLE TO <DDM Admin>`
- `GRANT DROP ANY TABLE TO <DDM Admin>`
- `GRANT INSERT ANY TABLE TO <DDM Admin>`

Using DBLink

If you use DBLink to access the Oracle database, you must set DBLink to `PUBLIC`. If DBLink is not set to `PUBLIC`, Dynamic Data Masking will not be able to access the database.

Changing the Listener Port

If the Dynamic Data Masking service runs on a standalone server, you must modify the `tnsnames.ora` file to add a database alias for Dynamic Data Masking service. The `tnsnames.ora` file is a configuration file that defines the addresses that the client uses to connect to the database. When you add a database alias to the file, applications send requests to the Dynamic Data Masking listener instead of the database listener.

Before you change the `tnsnames.ora` file, back up the original copy.

1. Open `tnsnames.ora`. By default, this file is located in the following location: `<Oracle install directory>/app/oracle/product/<product version>/server/NETWORK/ADMIN`
2. Find the following entry in the `tnsnames.ora` file:

   ```
   DBNAME=(DESCRIPTION=
             (ADDRESS=(PROTOCOL=TCP)(HOST=dbServer)(PORT=1521))
                       (CONNECT_DATA=(SERVICE_NAME=prod.mycompany.com)))
   ```
3. Replace the service listener port and host with the Dynamic Data Masking host and listener port.
   The following tnsnames.ora file shows the entry updated with the updated host and listener port.

   ```
   DBNAME=(DESCRIPTION=
         (ADDRESS=(PROTOCOL=TCP)(HOST=DynamicDataMasking)(PORT=1525))
           (CONNECT_DATA=(SERVICE_NAME=prod.mycompany.com)))
   ```

Configuring the Oracle Target Database Example

The target database configuration defines the connection parameters that the Dynamic Data Masking service uses to connect to the database. The easiest way to identify the correct database connection settings is to use the information from `tnsnames.ora`. The following example shows how you can use the database parameters from `tnsnames.ora` to configure a database in the Database Editor.

The following entry is from `tnsnames.ora`:

```
ORA11G =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = 127.0.0.1)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = ora11g)
    )
  )
```

The following table shows how the parameters in `tnsnames.ora` correspond to the parameters in the Management Console:

Parameter Value in tnsnames.ora	Parameter Name in the Database Editor
ORA11G	Instance Name
localhost	Listener Address
1521	Listener Port
ora11g	Service Name

Define the CURRENT_SCHEMA Symbol

If you want to query a table with the `set current_schema` command and without a table prefix, you must create a security rule to define the CURRENT_SCHEMA symbol.

Create a security rule with the following properties to set the current schema:

Matcher

Select the Text matcher. In the Text matcher, select the Regular Expression Identification Method.

Enter the following regular expression in the Text field:

```
\s*ALTER\s+SESSION\s+SET\s+CURRENT_SCHEMA\s*=\s*(.*)
```

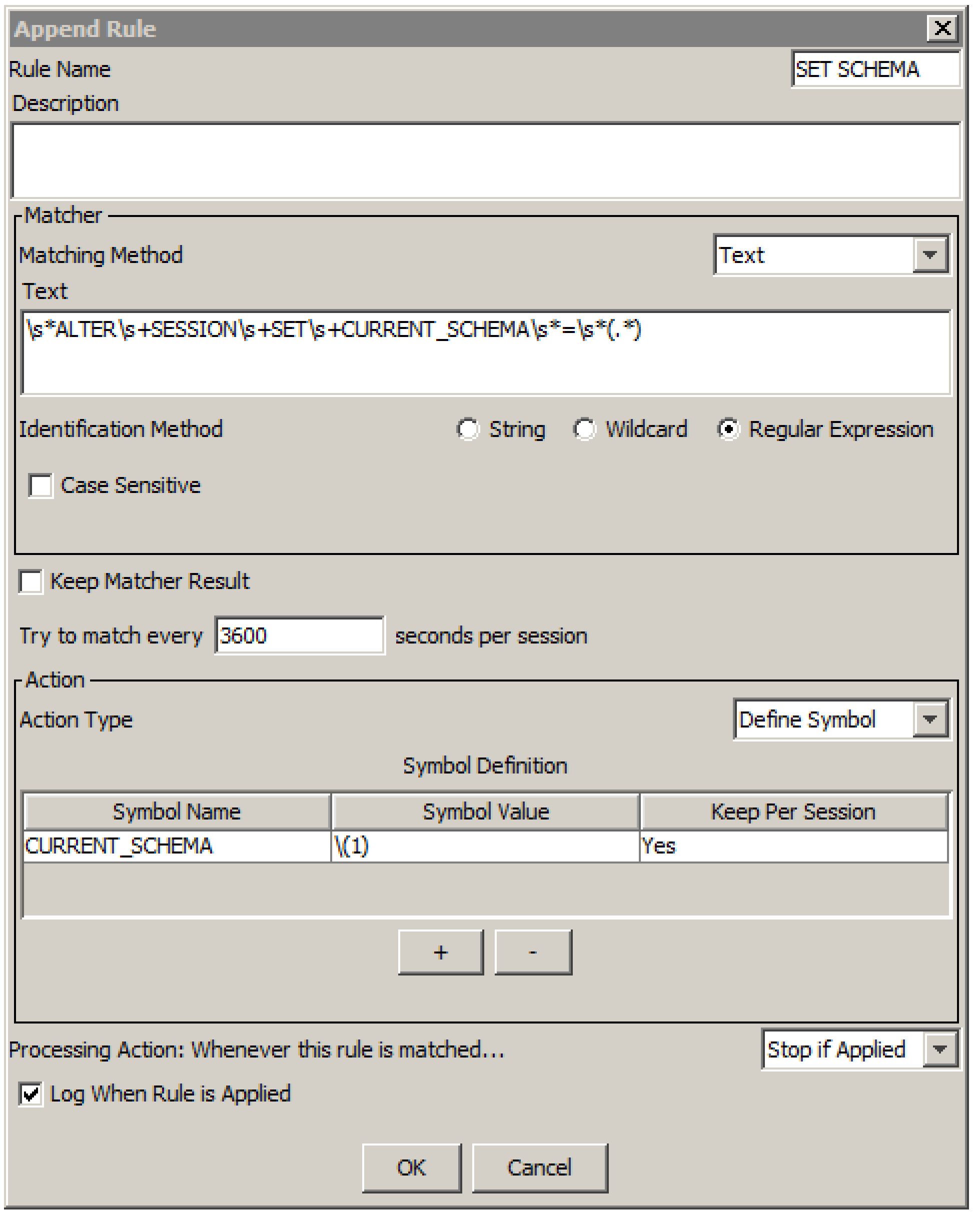
Rule Action

Select the Define Symbol action. Define the following rule action parameters:

- Symbol Name: CURRENT_SCHEMA
- Symbol Value: \(1)
- Keep Per Session: Yes

Note: For alternative syntaxes of SET SCHEMA, you can change the regular expression in the Text matcher based on the SQL syntax option that you use for the SET SCHEMA command.

The following image shows the rule properties:



Append Rule

Rule Name: SET SCHEMA

Description:

Matcher

Matching Method: Text

Text: \s*ALTER\s+SESSION\s+SET\s+CURRENT_SCHEMA\s*=\s*(.*)

Identification Method: String [✓ Regular Expression] Wildcard

☐ Case Sensitive

☐ Keep Matcher Result

Try to match every 3600 seconds per session

Action

Action Type: Define Symbol

Symbol Definition

Symbol Name	Symbol Value	Keep Per Session
CURRENT_SCHEMA	\(1)	Yes

Processing Action: Whenever this rule is matched... Stop if Applied

☑ Log When Rule is Applied

PostgreSQL Connection Management

Select the **PostgreSQL** database type to add a PostgreSQL database connection node to the Management Console tree.

Use **Test Connection** to verify that the Dynamic Data Masking service can access the database.

Configuring the PostgreSQL Driver

To access a PostgreSQL database, you must manually place the PostgreSQL driver in the Dynamic Data Masking installation directory.

1. Download the following driver from the PostgreSQL website:
   - `Postgresql-42.2.5.jar`
2. Stop the Dynamic Data Masking Server Windows Service.
3. Open a Server Control window and run the following commands:
   1. `server stop`
   2. `server remove`
4. Save the PostgreSQL driver in the following directory: `<Dynamic Data Masking installation>\lib\ext`
5. From the Server Control, run the following command: `server start`

   The **PostgreSQL** option appears in the **Database Type** drop-down menu on the Dynamic Data Masking Site Management Console.

PostgreSQL Connection Properties

Define the following connection parameters for a PostgreSQL database:

DDM Database Name

Name for the database in the Management Console tree.

Server Address

Server host name or TCP/IP address for the PostgreSQL database.

Note: Verify that there is no firewall that prohibits the Dynamic Data Masking Server from connecting to the database server and port number.

Server Port

TCP/IP listener port for the PostgreSQL database.

PostgreSQL Database Name

Database name for the PostgreSQL database.

Optional Parameters

Additional parameters for the PostgreSQL JDBC Driver.

Keystore

Select custom if you have configured a custom keystore. Select default if you want to use the default keystore that is preconfigured for use with Dynamic Data Masking.

DBA Username

User name for the database user account to log in to the PostgreSQL database. The database user must have SELECT access privileges for all the tables to which the client user has SELECT access privileges. This parameter is valid for the default keystore.

DBA Password

Password for the database user. This parameter is valid for the default keystore.

Keystore Name

Name of the custom keystore defined in the `ddm.security` file. This parameter is valid for custom keystores.

Alias

Alias name for the custom keystore. For CyberArk accounts, the alias name was defined when the CyberArk account was created. This parameter is valid for custom keystores.

PostgreSQL Dynamic Data Masking Administrator Required Privileges

The Dynamic Data Masking administrator must have privileges to access sensitive tables and columns.

Create a super user and the following properties to `true`:

- `rolsuper`
- `rolinherit`
- `rolcanlogin`

Use the super-user account to configure database nodes.

Defining the CURRENT_PATH Symbol

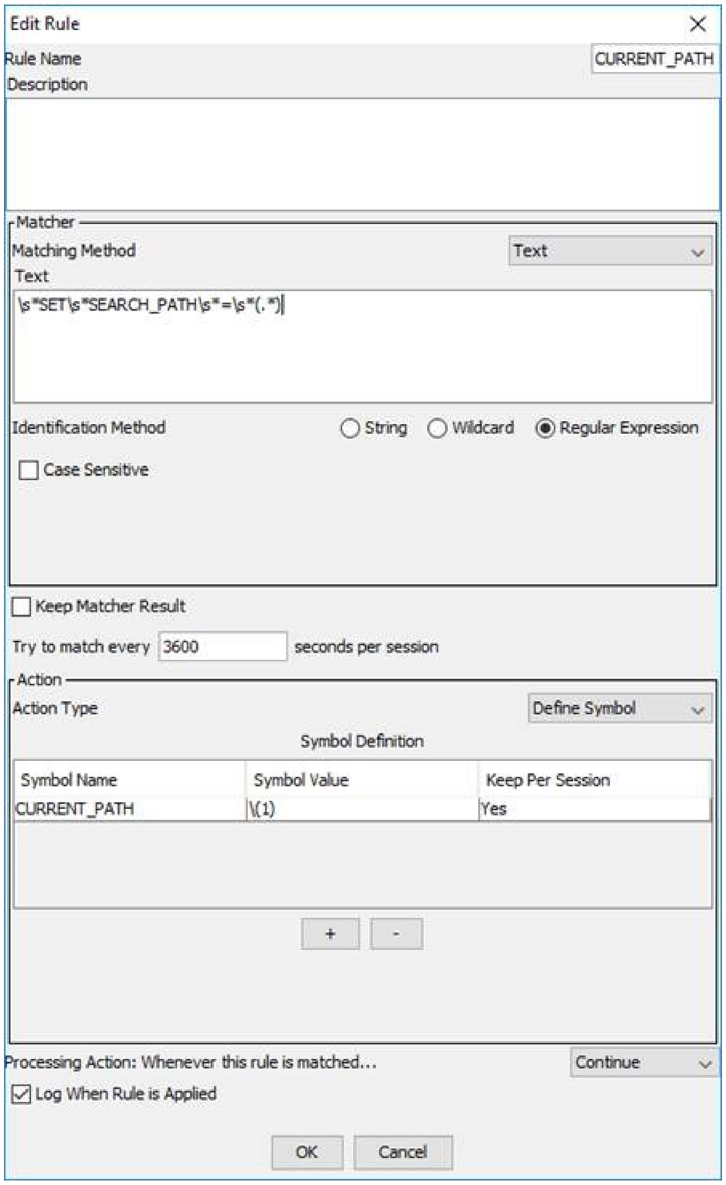
If you use the SET SEARCH_PATH = <schema name> command to switch between schema, define the CURRENT_PATH symbol to capture the former command.

Create a security rule with the following properties to set the current schema:

1. Go to the **Edit Rule** page.
2. In the **Rule Name** field, enter a name such as `CURRENT_PATH`.
3. In the **Matcher** section, perform the following steps:
   a. Set the **Matching Method** property to **Text**.
   b. Set the **Identification Method** property to **Regular Expression**.
   c. Enter the following text in the **Text** field:
      ```
      \s*SET\s*SEARCH_PATH\s*=\s*(.*)
      ```
4. In the **Action** section, perform the following steps:
   a. Set the **Action Type** field to **Define Symbol**.
   b. Create the CURRENT_PATH symbol.
   c. Enter `\(1)` in the **Symbol Value** column.
   d. Specify `Yes` in the **Keep Per Session** column.
5. Select the **Log When Rule is Applied** check box to create a line in the `rule.log` file when the Rule Engine applies the rule.

Example

The following image shows an example definition of the CURRENT_PATH symbol on the **Edit Rule** page:



Rule Name: CURRENT_PATH

Description:

Matching Method: Text

Text:

```
\s*SET\s*SEARCH_PATH\s*=\s*(.*)
```

Identification Method: String Wildcard [✓ Regular Expression]

☐ Case Sensitive

☐ Keep Matcher Result

Try to match every 3600 seconds per session

Action Type: Define Symbol

Symbol Name	Symbol Value	Keep Per Session
CURRENT_PATH	\(1)	Yes

Processing Action: Whenever this rule is matched... Continue

☑ Log When Rule is Applied

Sybase Connection Management

Select the Sybase database type to add a Sybase database connection node to the Management Console tree.

The Sybase connection request does not contain information about the database. You must define the target database that Dynamic Data Masking forwards the request to. Make a connection rule that uses the Switch to Database rule action to define the target database. Specify a database in the rule that corresponds to the Dynamic Data Masking Database Name parameter.

Click **Test Connection** to verify that the Dynamic Data Masking service can access the database.

Sybase Connection Parameters

Define the following connection parameters for a Sybase database:

DDM Database Name

Name for the database in the Management Console tree.

Server Address

Server host name or TCP/IP address for the Sybase database.

Note: Verify that there is no firewall that prohibits the Dynamic Data Masking Server from connecting to the database server and port number.

Server Port

TCP/IP listener port for the Sybase database.

Optional Parameters

Additional parameters for the Informatica driver for Sybase.

Keystore

Select custom if you have configured a custom keystore. Select default if you want to use the default keystore preconfigured for use with Dynamic Data Masking.

DBA Username

User name for the database user account to log in to the Sybase database. The database user must be a privileged user that has SELECT access to all the tables that the client user has SELECT access to. This parameter is valid for the default keystore.

DBA Password

Password for the database user account. This parameter is valid for the default keystore.

Key Store Name

Name of the custom keystore, defined in the `ddm.security` file. This parameter is valid for custom keystores.

Alias

Alias name for the custom keystore. For CyberArk accounts, the alias name was defined during creation of the CyberArk account. This parameter is valid for custom keystores.

Sybase Dynamic Data Masking Administrator Required Privileges

The Dynamic Data Masking administrator must have privileges to access sensitive tables and columns.

Log in to a Sybase client as an administrator that is not the Dynamic Data Masking administrator and run the following commands:

- `SP_ADDLOGIN <DDM Admin>, <DDM Admin password>`
- `GRANT ROLE SSO_ROLE TO <DDM Admin>`

Additional Privileges for SELECT * Statements

If your Dynamic Data Masking security rules need to support column masking on SELECT * statements, you must also run the following command from the master database:

```
GRANT SET PROXY TO <DDM Admin>
```

Search and Replace Rule

If you use Sybase with ODBC drivers and have the quoted_identifier option set to off or the query contains a table name with aliases, create a search and replace security rule.

If the quoted_identifier option is set to off or the query contains a table name with aliases, a query sent to the database through Dynamic Data Masking returns an error. You must create a security rule that removes double quotes from the query.

Queries to the database must use a masking rule with the following parameters:

Rule Name

The name of the rule. For example, Replace.

Matcher

Select the Any matcher. The rule applies to all queries sent to the database.

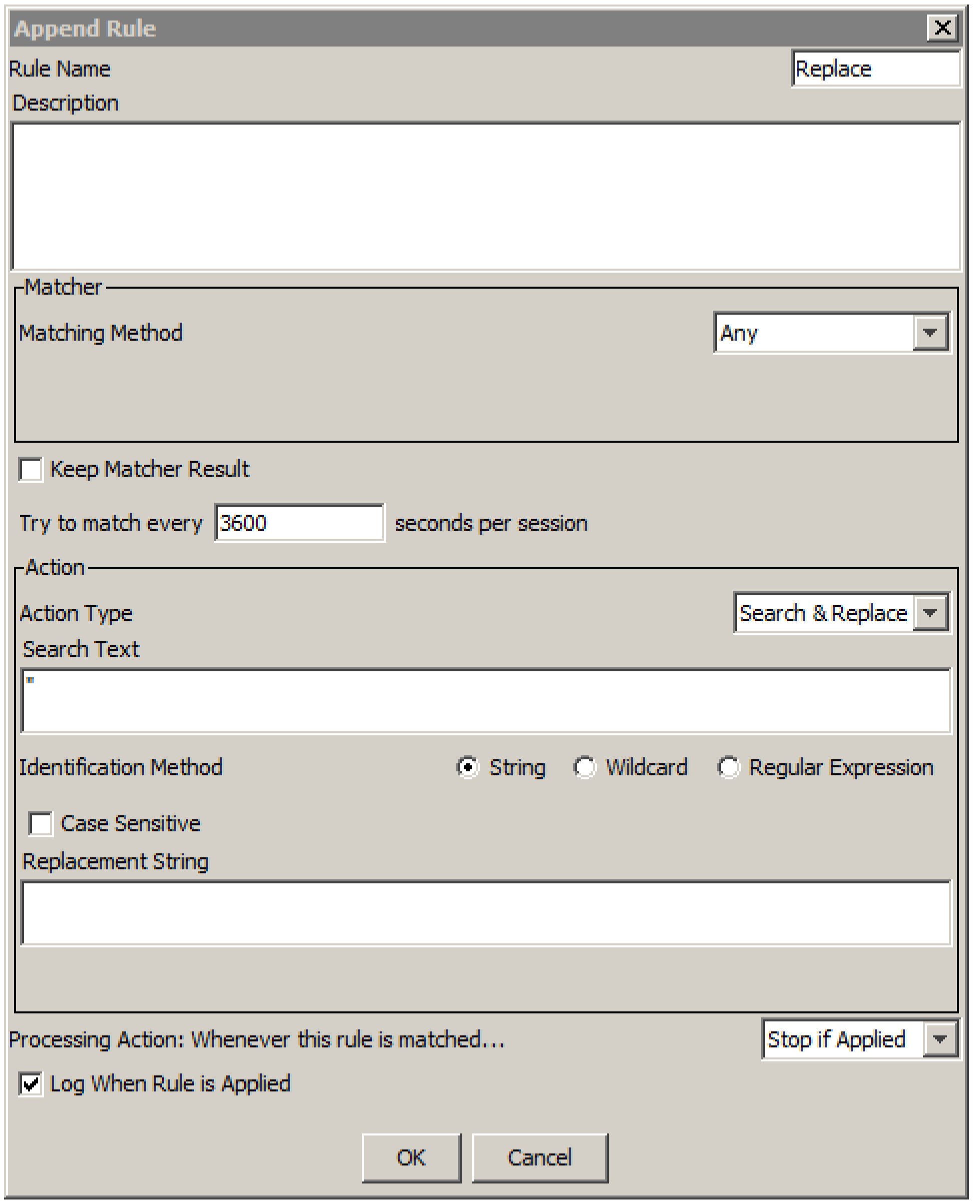
Rule Action

Select the Search and Replace rule action. In the Search Text field, enter a double quote ("). Leave the Replacement String field blank. In the query, the Rule Engine removes double quotes.

Log When Rule is Applied

Select the Log When Rule is Applied check box to create a line in the rule.log file when the Rule Engine applies the rule.

The following image is an example of the search and replace rule:


Append Rule

Rule Name: Replace

Description:

Matcher

Matching Method: Any

[ ] Keep Matcher Result

Try to match every 3600 seconds per session

Action

Action Type: Search & Replace

Search Text: *

Identification Method: [✓ String] Wildcard Regular Expression

[ ] Case Sensitive

Replacement String:

Processing Action: Whenever this rule is matched... Stop if Applied

[✓] Log When Rule is Applied

Teradata Connection Management

Select the Teradata database type to add a Teradata database connection node to the Management Console tree.

The Teradata connection request does not contain information about the database. You must define the target database that Dynamic Data Masking forwards the request to. Make a connection rule that uses the Switch to Database rule action to define the target database. Specify a database in the rule that corresponds to the Dynamic Data Masking Database Name parameter.

To access a Teradata database, you must manually add Teradata drivers into the Dynamic Data Masking installation directory.

Click **Test Connection** to verify that the Dynamic Data Masking service can access the database.

Teradata Connection Parameters

Define the following connection parameters for a Teradata database:

DDM Database Name

Name for the database in the Management Console tree.

Server Address

Server host name or TCP/IP address for the Teradata database.

Note: Verify that there is no firewall that prohibits the Dynamic Data Masking Server from connecting to the database server and port number.

Server Port

TCP/IP listener port for the Teradata database.

Keystore

Select custom if you have configured a custom keystore. Select default if you want to use the default keystore preconfigured for use with Dynamic Data Masking.

DBA Username

User name for the database user account to log in to the Teradata database. The database user must be a privileged user that has SELECT access to all the tables that the client user has SELECT access to. This parameter is valid for the default keystore.

DBA Password

Password for the database user account. This parameter is valid for the default keystore.

Key Store Name

Name of the custom keystore, defined in the `ddm.security` file. This parameter is valid for custom keystores.

Alias

Alias name for the custom keystore. For CyberArk accounts, the alias name was defined during creation of the CyberArk account. This parameter is valid for custom keystores.

Teradata Dynamic Data Masking Administrator Required Privileges

To run SELECT * commands, the Dynamic Data Masking administrator must have the required privileges.

The Dynamic Data Masking administrator must have the following privileges to run SELECT * statements:

- The administrator must be able to make the client catalog the default catalog. The administrator must also be able to make the client catalog the default catalog when the client's default catalog is set within the client's profile.
- The administrator must have all the SELECT and EXECUTE grants, or roles that contain the grants, that the client user has on objects that you want to mask.
- If the client session is set with query banding and one of the query bands is ProxyUser, the Dynamic Data Masking administrator must have the same CONNECT THROUGH grants as the client user.

Configuring the Teradata Drivers

To access a Teradata database, you must manually place the Teradata drivers in the Dynamic Data Masking installation directory.

1. Download the Teradata drivers from the Teradata website. You need the following drivers:
   - tdgssconfig.jar
   - terajdbc4.jar
2. Stop the Dynamic Data Masking Server Windows service.
3. Open a Server Control window and run the following commands:
   - `server stop`
   - `server remove`
4. Save the Teradata drivers in the following directory:
   ```
   <Dynamic Data Masking installation>\lib\ext
   ```
5. From Server Control, run the following command:
   ```
   server start
   ```

CHAPTER 5

JDBC Client Configuration

This chapter includes the following topics:

JDBC Client Configuration Overview

Before you use Dynamic Data Masking to mask data for a database that uses a JDBC connection, you must configure the client.

The Dynamic Data Masking installation contains a JAR file that you must save on the client machine. The JAR file contains a Java agent, a transformer that intercepts the method calls of JDBC objects, and proxy classes for JDBC classes. You can find the JAR file in the following location:

```
<Dynamic Data Masking installation>\Wrappers\jdbc\GenericJDBC.jar
```

You must perform additional configuration steps based on the client and operating system.

Note: Because the client looks for the Dynamic Data Masking service on the host and port, the client connection fails when the Dynamic Data Masking Server is down. To connect to the database, the Dynamic Data Masking Server must be running.

Apache Tomcat Configuration

Follow the configuration steps for the Apache Tomcat client based on the operating system.

Configure Apache Tomcat for Windows

Complete the following steps to configure Apache Tomcat for Windows:

1. Copy the GenericJDBC.jar file. You can find the file in the following location:

   ```
   <Dynamic Data Masking installation>\Wrappers\jdbc\GenericJDBC.jar
   ```

2. Save the GenericJDBC.jar file in the following location:

   ```
   <Apache Tomcat installation>\lib
   ```

3. Find the catalina.bat file. You can find the file in the following directory:

   ```
   <Apache Tomcat installation>\bin\catalina.bat
   ```

4. Save a backup of the catalina.bat file.
5. Open the catalina.bat file in a text editor and find the `setlocal` line. To append the -javaagent argument to the Java command line, enter one of the following options under the `setlocal` line:
   - If you want to enable logging on the client machine, enter the following text:

     ```
     set JAVA_OPTS=-  -DDDM_GENJDBC_LOG = <config file location and name>   -
     javaagent:..\lib\GenericJDBC.jar=host:<ddm_host>,port:<ddm_generic_service_port>
     ```

   - If you do not want to enable logging on the client machine, enter the following text:

     ```
     set JAVA_OPTS=-javaagent:..\lib
     \GenericJDBC.jar=host:<ddm_host>,port:<ddm_generic_service_port>
     ```

6. To set the classpath, add the following line under the line that you added in the previous step:

   ```
   set CLASSPATH=%CLASSPATH%;..\lib\GenericJDBC.jar;
   ```

7. Save catalina.bat.

Configure Apache Tomcat for Linux

Complete the following steps to configure Apache Tomcat for Linux:

1. Copy the GenericJDBC.jar file. You can find the file in the following location:

   ```
   <Dynamic Data Masking installation>/Wrappers/jdbc/GenericJDBC.jar
   ```

2. Save the GenericJDBC.jar file in the following location:

   ```
   <Apache Tomcat installation>/lib
   ```

3. Find the catalina.sh file. You can find the file in the following directory:

   ```
   <Apache Tomcat installation>/bin/catalina.bat
   ```

4. Save a backup of the catalina.sh file.
5. Open the catalina.sh file in a text editor and find the line that contains `#JAVA_OPTS=`. To append the -javaagent argument to the Java command line, enter one of the following options under the line that contains `#JAVA_OPTS=`:
   - If you want to enable logging on the client machine, enter the following text:

     ```
     export JAVA_OPTS=-  -DDDM_GENJDBC_LOG = <config file location and name>   -
     javaagent:..\lib\GenericJDBC.jar=host:<ddm_host>,port:<ddm_generic_service_port>
     ```

   - If you do not want to enable logging on the client machine, enter the following text:

     ```
     export JAVA_OPTS=-javaagent:../lib/
     GenericJDBC.jar=host:<ddm_host>,port:<ddm_generic_service_port>
     ```

6. To set the classpath, add the following line under the line that you added in the previous step:

   ```
   set CLASSPATH=$CLASSPATH:../lib/GenericJDBC.jar
   ```

7. Save catalina.sh.

Aqua Data Studio Configuration

Complete the following steps to configure the Aqua Data Studio client:

1. Copy the GenericJDBC.jar file. You can find the file in the following location:

   ```
   <Dynamic Data Masking installation>\Wrappers\jdbc\GenericJDBC.jar
   ```

2. Save the GenericJDBC.jar file in the following location:

   ```
   <Aqua Data Studio installation>\lib
   ```

3. If you want to use datastudio.bat to launch Aqua Data Studio, complete the following steps:

   a. Find the datastudio.bat file. You can find the file in the Aqua Data Studio installation directory.

   b. Save a backup of the datastudio.bat file.

   c. Open the datastudio.bat file in a text editor. To set the environment variable, add one of the following options to the last line of text in the file:

   - If you want to enable logging on the client machine, enter the following text:

     ```
     -DDDM_GENJDBC_LOG = <config file location and name>    -javaagent:.\lib
     \GenericJDBC.jar= host:<ddm_host>,port:<ddm_generic_service_port>
     ```

   - If you do not want to enable logging on the client machine, enter the following text:

     ```
     -javaagent:.\lib\GenericJDBC.jar=
     host:<ddm_host>,port:<ddm_generic_service_port>
     ```

   The line that you modified is similar to the following text:

   ```
   •    java -Dfile.encoding=UTF-8 -Xms512M - true -DDDM_GENJDBC_LOG = <config file
   location and name>     -javaagent:.\lib\GenericJDBC.jar=
   host:<ddm_host>,port:<ddm_generic_service_port> -cp ".\lib\ads.jar;%ADS_PATH%"
   com.aquafold.datastudio.DataStudio
   ```

   d. Save datastuido.bat.

4. If you want to use datastudio.exe to launch Aqua Data Studio, complete the following steps:

   a. Find the datastudio.cfg file. You can find the file in the Aqua Data Studio installation directory.

   b. Save a backup of the datastudio.cfg file.

   c. Open the datastudio.cfg file in a text editor. To append the -javaagent argument to the Java command line, add one of the following Java agent arguments before the `-cp` text:

   - If you want to enable logging on the client machine, enter the following text:

     ```
     -DDDM_GENJDBC_LOG = <config file location and name>     -javaagent:.\lib
     \GenericJDBC.jar= host:<ddm_host>,port:<ddm_generic_service_port>
     ```

   - If you do not want to enable logging on the client machine, enter the following text:

     ```
     -javaagent:.\lib\GenericJDBC.jar=
     host:<ddm_host>,port:<ddm_generic_service_port>
     ```

Oracle SQL Developer Configuration

Complete the following steps to configure the Oracle SQL Developer client:

1. Copy the GenericJDBC.jar file. You can find the file in the following location:
   ```
   <Dynamic Data Masking installation>\Wrappers\jdbc\GenericJDBC.jar
   ```
2. Save the GenericJDBC.jar file in the following location:
   ```
   <SQL Developer installation>\sqldeveloper\lib
   ```
3. Find the sqldeveloper.conf file. You can find the file in the in the following location:
   ```
   <SQL Developer installation>\sqldeveloper\bin
   ```
4. Save a backup of the sqldeveloper.conf file.
5. Open the sqldeveloper.conf file in a text editor. To append the -javaagent argument to the Java command line, add one of the following options to the file:
   - If you want to enable logging on the client machine, enter the following text:
     ```
     AddVMOption   -DDDM_GENJDBC_LOG = <config file location and name>
     AddVMOption   -javaagent:..\lib\GenericJDBC.jar=
     host:<ddm_host>,port:<ddm_generic_service_port>
     ```
   - If you do not want to enable logging on the client machine, enter the following text:
     ```
     Add VM Option -javaagent:..\lib\GenericJDBC.jar=
     host:<ddm_host>,port:<ddm_generic_service_port>
     ```
6. Save sqldeveloper.conf.
7. Use sqldeveloper.exe to launch Oracle SQL Developer.

SQuirreL Configuration

Complete the following steps to configure the SQuirreL SQL client:

1. Copy the GenericJDBC.jar file. You can find the file in the following location:
   ```
   <Dynamic Data Masking installation>\Wrappers\jdbc\GenericJDBC.jar
   ```
2. Save the GenericJDBC.jar file in the following location:
   ```
   <SQuirreL installation>\lib
   ```
3. Find the squirrel-sql.bat file. You can find the file in the SQuirreL installation directory.
4. Save a backup of the squirrel-sql.bat file.
5. Open the squirrel-sql.bat file in a text editor. To append the -javaagent argument to the Java command line, add one of the following options to the penultimate line of text in the file:
   - If you want to enable logging on the client machine, enter the following text:
     ```
     -DDDM_GENJDBC_LOG = <config file location and name>   -javaagent:.\lib
     \GenericJDBC.jar=host:<ddm_host>,port:<ddm_generic_service_port>
     ```
   - If you do not want to enable logging on the client machine, enter the following text:
     ```
     -javaagent:.\lib\GenericJDBC.jar= host:<ddm_host>,port:<ddm_generic_service_port>
     ```

   The line that you modified is similar to the following text:
   ```
   "%LOCAL_JAVA%"  -verbose -Xmx256m -Dsun.java2d.noddraw=true -DDDM_GENJDBC_LOG =
   <config file location and name>    -javaagent:.\lib\GenericJDBC.jar=
   host:<ddm_host>,port:<ddm_generic_service_port>  -cp %SQUIRREL_CP%    -
   ```

```
splash:"%SQUIRREL_SQL_HOME%/icons/splash.jpg"
net.sourceforge.squirrel_sql.client.Main %TMP_PARMS%
```

6. Save squirrel-sql.bat.

WebLogic Configuration

Follow the configuration steps for the WebLogic client based on the operating system.

Configure WebLogic for Windows

Complete the following steps to configure WebLogic for Windows:

1. Copy the GenericJDBC.jar file. You can find the file in the following location:
   ```
   <Dynamic Data Masking installation>\Wrappers\jdbc\GenericJDBC.jar
   ```
2. Save the GenericJDBC.jar file in the following location:
   ```
   ...\Middleware\user_projects\domains\base_domain\lib
   ```
   For example, you might save the file in the following location:
   ```
   C:\Oracle\Middleware\user_projects\domains\base_domain\lib
   ```
3. Find the startWebLogic.cmd file. You can find the file in the following directory:
   ```
   ...\Middleware\user_projects\domains\base_domain\bin
   ```
   For example, the file might be in the following location:
   ```
   C:\Oracle\Middleware\user_projects\domains\base_domain\bin\startWebLogic.cmd
   ```
4. Save a backup of the startWebLogic.cmd file.
5. Open the startWebLogic.cmd file in a text editor. To append the -javaagent argument to the Java command line, enter one of the following options below the comments section:
   - If you want to enable logging on the client machine, enter the following text:
     ```
     set JAVA_OPTIONS=% JAVA_OPTIONS %  -DDDM_GENJDBC_LOG = <config file location and
     name>   -javaagent:..\lib
     \GenericJDBC.jar=host:<ddm_host>,port:<ddm_generic_service_port>
     ```
   - If you do not want to enable logging on the client machine, enter the following text:
     ```
     set JAVA_OPTIONS=% JAVA_OPTIONS % -javaagent:..\lib
     \GenericJDBC.jar=host:<ddm_host>,port:<ddm_generic_service_port>
     ```
6. To set the classpath, add the following line after the classpath lines and before the echo statements:
   ```
   set CLASSPATH=%CLASSPATH%;..\lib\GenericJDBC.jar; ..\lib\informatica-jdbc-
   db2-5.1.2.HF1.jar;<additional drivers>
   ```
7. Save startWebLogic.cmd.

Configure WebLogic for Linux

Complete the following steps to configure WebLogic for Linux:

1. Copy the GenericJDBC.jar file. You can find the file in the following location:
   ```
   <Dynamic Data Masking installation>/Wrappers/jdbc/GenericJDBC.jar
   ```
2. Save the GenericJDBC.jar file in the following location:
   ```
   .../Middleware/user_projects/domains/base_domain/lib
   ```

For example, you might save the file in the following location:

```
C:/Oracle\Middleware/user_projects/domains/base_domain/lib
```

3. Find the startWebLogic.sh file. You can find the file in the following directory:

   ```
   ...\Middleware\user_projects\domains\base_domain\bin
   ```

   For example, the file might be in the following location:

   ```
   C:\Oracle\Middleware\user_projects\domains\base_domain\bin\startWebLogic.cmd
   ```

4. Save a backup of the startWebLogic.cmd file.
5. Open the startWebLogic.cmd file in a text editor. To append the -javaagent argument to the Java command line, enter one of the following options below the comments section:
   - If you want to enable logging on the client machine, enter the following text:

     ```
     set JAVA_OPTIONS=% JAVA_OPTIONS %  -DDDM_GENJDBC_LOG = <config file location and
     name>   -javaagent:..\lib
     \GenericJDBC.jar=host:<ddm_host>,port:<ddm_generic_service_port>
     ```

   - If you do not want to enable logging on the client machine, enter the following text:

     ```
     set JAVA_OPTIONS=% JAVA_OPTIONS % -javaagent:..\lib
     \GenericJDBC.jar=host:<ddm_host>,port:<ddm_generic_service_port>
     ```

6. To set the classpath, add the following line after the classpath lines and before the echo statements:

   ```
   set CLASSPATH=%CLASSPATH%;..\lib\GenericJDBC.jar; ..\lib\informatica-jdbc-
   db2-5.1.2.HF1.jar;<additional drivers>
   ```

7. Save startWebLogic.cmd.

CHAPTER 6

ODBC Client Configuration

This chapter includes the following topic:

ODBC Client Configuration Overview

Before you use the Dynamic Data Masking Generic ODBC approach to mask data for a database that uses an ODBC connection, you must configure the client machine.

The Dynamic Data Masking installation includes ODBC DLL files for 64-bit and 32-bit Windows. You must save the files in the Windows system directory and create Dynamic Data Masking host and port environment variables. Optionally, you can create a Data Source in the Windows ODBC Administrator to test the configuration.

1. Verify the requirements for ODBC client configuration.
2. Grant the required permissions to the user that performs the setup for the Windows Driver Manager files.
3. Set up the Dynamic Data Masking Driver Manager proxies.
4. Create the environment variables.
5. Optionally, create the Windows Data Source to test the client configuration.

Step 1. Verify the Requirements

Verify the following requirements before you configure the client:

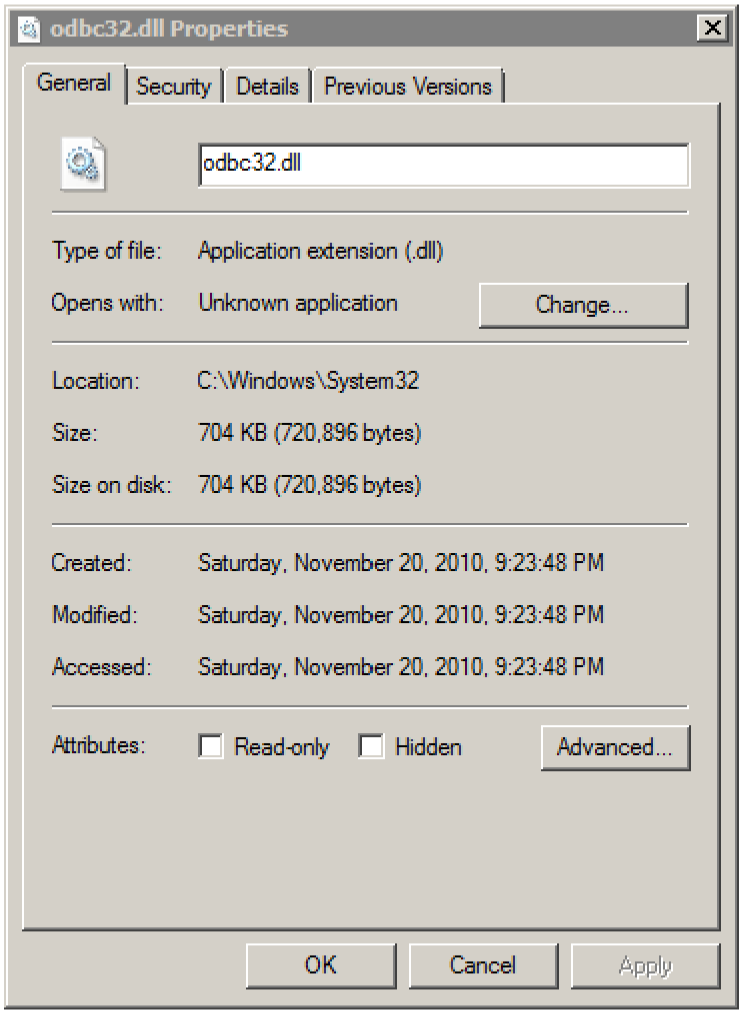
- The client must have a Windows 7 or Windows Server 2008 operating system.
- The Windows user that performs the setup must have administrator privileges to modify the ODBC DLLs or the user must have the rights to grant the required privileges or ownership.
- The Windows user that performs the setup must have the required permissions to rename the Windows ODBC Driver Manager (odbc32.dll).

Step 2. Grant File Permissions

Before you set up the Driver Manager proxies, you must identify the Driver Manager proxy that you want to install on the target application architecture and the ODBC 32-bit or 64-bit usage. Then you must change the owner of the Windows Driver Manager files and grant the required permissions.

1. Find the ODBC Driver Manager DLL file on the Windows machine based on the architecture of the target application. Complete the remaining steps for each of the required Driver Manager files on the machine.
   a. On a 64-bit Windows machine, you can find the following Windows Driver Manager files:
      - 64-bit Driver Manager: `<Windows installation>\System32\odbc32.dll`
      - 32-bit Driver Manager: `<Windows installation>\SysWOW64\odbc32.dll`
   b. On a 32-bit Windows machine, you can find the following Windows Driver Manager file:
      - 32-bit Driver Manager: `<Windows installation>\System32\odbc32.dll`
2. In Windows Explorer, right-click the file and click **Properties**.

   The **odbc32.dll Properties** window opens. The following image shows the window:



odbc32.dll

Type of file: Application extension (.dll)

Opens with: Unknown application

Location: C:\Windows\System32

Size: 704 KB (720,896 bytes)

Size on disk: 704 KB (720,896 bytes)

Created: Saturday, November 20, 2010, 9:23:48 PM

Modified: Saturday, November 20, 2010, 9:23:48 PM

Accessed: Saturday, November 20, 2010, 9:23:48 PM

Attributes: ☐ Read-only ☐ Hidden

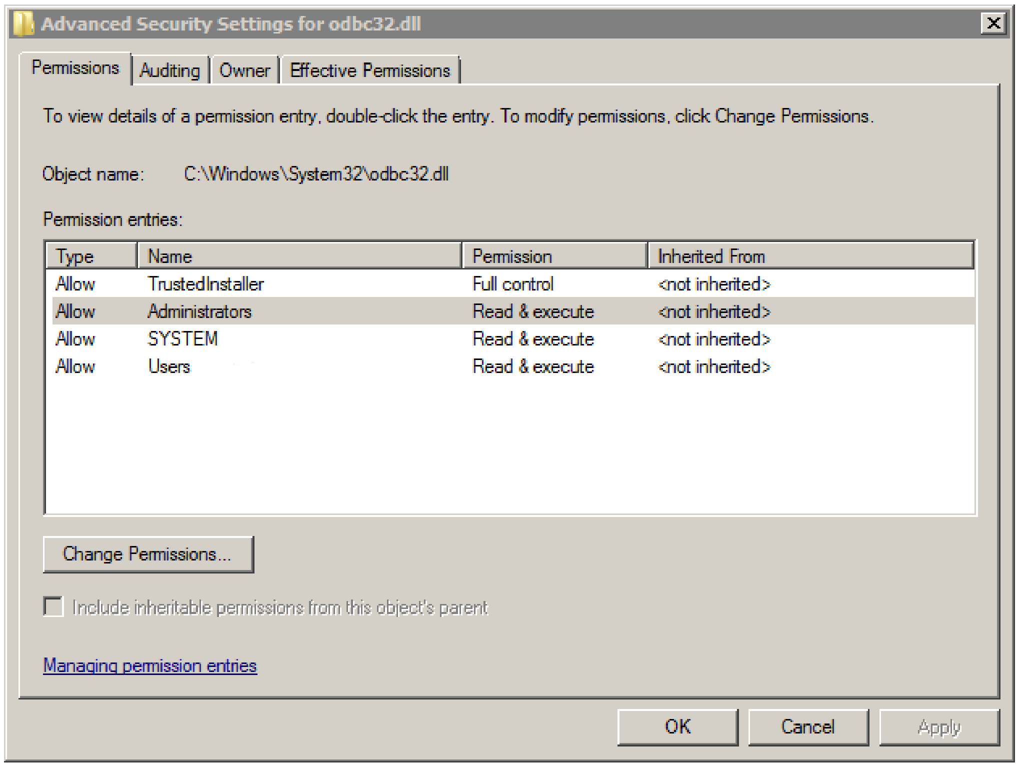
3. Select the **Security** tab and click **Advanced**.

   The **Advanced Security Settings for odbc32.dll** window opens.

4. Select the **Owner** tab and click **Edit**.
5. In the window that opens, select the current user and click **Apply**.
6. Click **OK** to close the window.
7. Click **OK** to close the **Advanced Security Settings for odbc32.dll** window.
8. In the **odbc32.dll Properties** window, click **OK** to close the window. You must close the properties window before you make additional changes to the file.
9. Reopen the **odbc32.dll Properties** window. Right-click the file and click **Properties**.
10. Select the **Security** tab and click **Advanced**.

    The **Advanced Security Settings for odbc32.dll** window opens.
11. In the **Permissions** tab, select the Administrators group and click **Change Permissions**.

    The following image shows the **Permissions** tab:


Advanced Security Settings for odbc32.dll
Permissions
Auditing
Owner
Effective Permissions

To view details of a permission entry, double-click the entry. To modify permissions, click Change Permissions.

Object name: C:\Windows\System32\odbc32.dll

Permission entries:

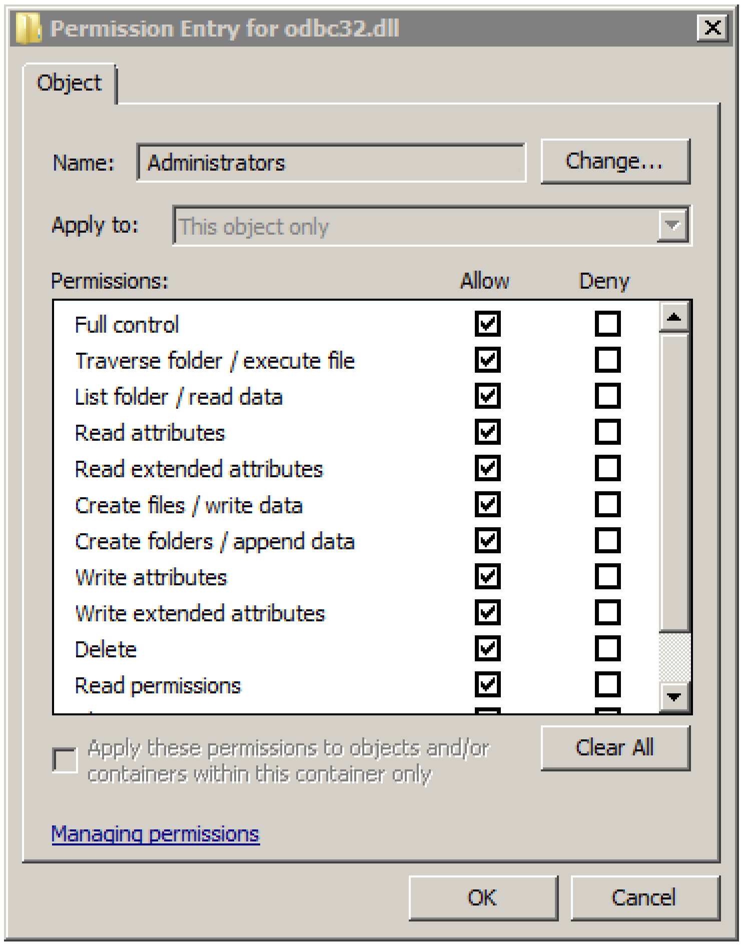
Type	Name	Permission	Inherited From
Allow	TrustedInstaller	Full control	<not inherited>
Allow	Administrators	Read & execute	<not inherited>
Allow	SYSTEM	Read & execute	<not inherited>
Allow	Users	Read & execute	<not inherited>

Change Permissions...
☐ Include inheritable permissions from this object's parent
Managing permission entries
OK
Cancel
Apply

12. In the window that opens, select the Administrators group again and click **Edit**.

    The **Permission Entry for odbc32.dll** window opens.
13. Click the **Allow** box in the **Full control** row to grant full permissions to the user.

    The following image shows the window with the **Full control** box selected:


Permission Entry for odbc32.dll

Object

Name: Administrators

Apply to: This object only

Permissions:	Allow	Deny
Full control	✓	
Traverse folder / execute file	✓	
List folder / read data	✓	
Read attributes	✓	
Read extended attributes	✓	
Create files / write data	✓	
Create folders / append data	✓	
Write attributes	✓	
Write extended attributes	✓	
Delete	✓	
Read permissions	✓	

Apply these permissions to objects and/or containers within this container only

14. Click **OK** to close the window.
15. Click **OK** to close the previous window. A dialog box asks if you want to continue. Click **Yes** to close the window.
16. Click **OK** to close the **Advanced Security Settings for odbc32.dll** window.
17. Click **OK** again to close the **odbc32.dll Properties** window.

    The permissions changes are saved.

Step 3. Set Up the Driver Manager Proxies

Save the Dynamic Data Masking Generic ODBC DLL files in the Windows system directory.

Complete the following steps for each Windows Driver Manager file that you edited in the previous step.

1. Rename the Windows Driver Manager file to odbc32.dll. Rename `odbc32.dll` to `odbc32o.dll`.

2. Find the Dynamic Data Masking Driver Manager proxy based on the architecture of the target client application.
    a. Find the following Dynamic Data Masking Generic ODBC DLL file for a 64-bit application:
        - `<Dynamic Data Masking installation>\Wrappers\odbc\Windows\odbc64\GenericOdbc64.dll`
    b. Find the following Dynamic Data Masking Generic ODBC DLL file for a 32-bit application:
        - `<Dynamic Data Masking installation>\Wrappers\odbc\Windows\odbc32\GenericOdbc32.dll`
3. Copy the Dynamic Data Masking Driver Manger file to the Windows system directory.
    a. Copy `GenericOdbc64.dll` to the following directory:

        `<Windows installation>\System32`
    b. Copy `GenericOdbc32.dll` to the following directory:
        - On a 64-bit machine: `<Windows installation>\SysWOW64`
        - On a 32-bit machine: `<Windows installation>\System32`
4. Rename the Dynamic Data Masking Generic ODBC DLL file to `odbc32.dll` for the 64-bit and 32-bit version.
    a. Rename `GenericOdbc64.dll` to `odbc32.dll`.
    b. Rename `GenericOdbc32.dll` to `odbc32.dll`.

Step 4. Create the Environment Variables

Create the Dynamic Data Masking host environment variables.

1. Open the Windows Start menu, right click **Computer**, and click **Properties**.

    The **Control Panel** opens.
2. On the right side of the Control Panel, click **Advanced system settings**.

    The **System Properties** window opens.
3. In the **System Properties** window, click the **Advanced** tab and click **Environment Variables** at the bottom of the window.

    The **Environment Variables** window opens.
4. Under the **System variables** box, click **New**.

    The **New System Variable** window opens.
5. In the **New System Variable** window, enter the following properties:

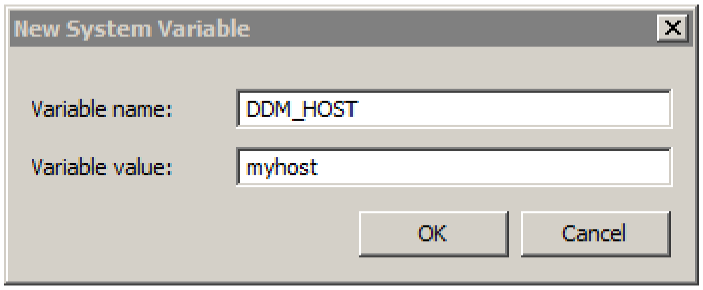
    Variable Name

        DDM_HOST

    Variable Value

        Dynamic Data Masking Server host name.

    The following image shows the **New System Variable** window:



Variable name: DDM_HOST

Variable value: myhost

6. Click **OK** to close the window.

   The DDM_HOST environment variable appears in the list of system variables.
7. In the System variables box, click New to add another variable.

   The **New System Variable** window opens.
8. In the **New System Variable** window, enter the following properties:

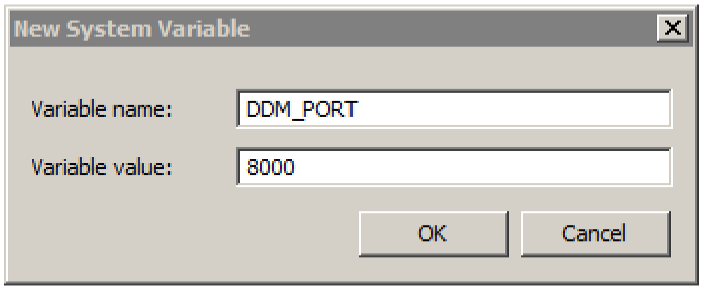
   Variable Name

   DDM_PORT

   Variable Value

   Dynamic Data Masking Generic ODBC service port.

   The following image shows the **New System Variable** window:



Variable name: DDM_PORT

Variable value: 8000

9. Click **OK** to close the window.

   The DDM_PORT environment variable appears in the list of system variables.
10. Click **OK** to close the **Environment Variables** window.
11. Click **OK** to close the **System Properties** window.

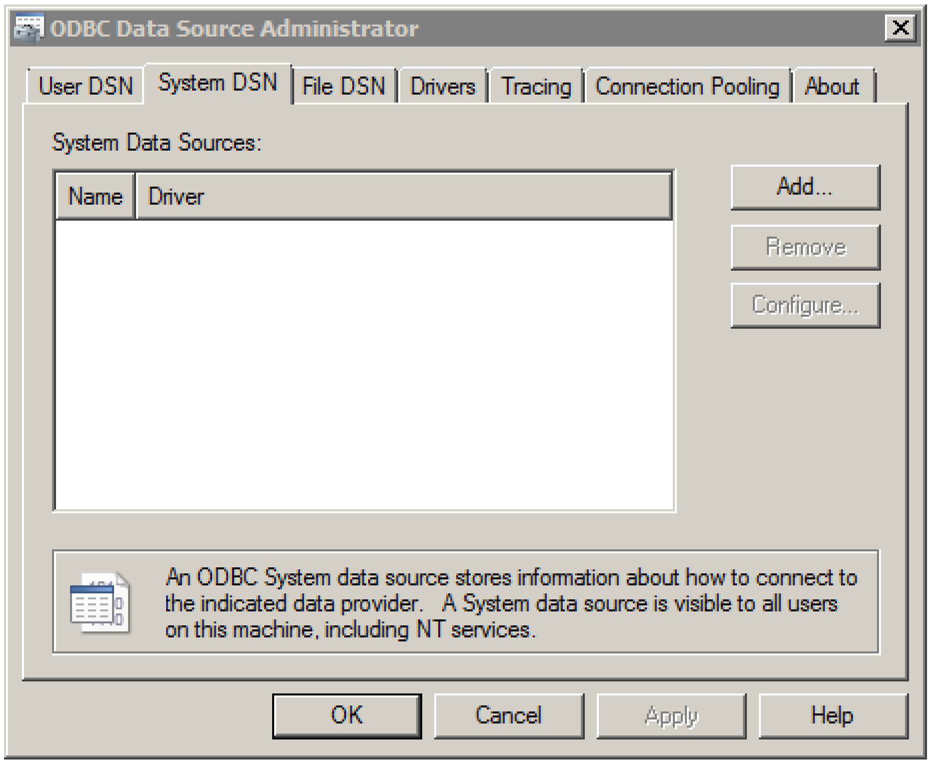
Step 5. Create a Windows Data Source (Test the Setup)

To test the client configuration, or if a System Data Source is not available on the client machine, you can create a System Data Source with the Windows ODBC Data Source Administrator, based on the architecture of the machine.

1. Open the Windows ODBC Data Source Administrator.
   a. On a 64-bit Windows machine, you can find the ODBC Data Source Administrator in the following locations:
      - 64-bit ODBC Administrator: `<Windows installation>\System32\odbcad32.exe`
      - 32-bit ODBC Administrator: `<Windows installation>\SysWOW64\odbcad32.exe`

   b. On a 32-bit Windows machine, you can find the ODBC Data Source Administrator in the following location:
      - 32-bit ODBC Administrator: `<Windows installation>\System32\odbcad32.exe`
2. To view the System Data Sources, click the **System DSN** tab at the top of the window.

   The following image shows the **System DSN** properties:



An ODBC System data source stores information about how to connect to the indicated data provider. A System data source is visible to all users on this machine, including NT services.

3. Click **Add**.

   The **Create New Data Source** window opens.
4. In the **Create New Data Source** window, click the name of the ODBC driver that you want to create a DSN for, based on the database that you want to connect to.
5. Click **Finish**.

   The ODBC setup window opens.

6. Provide the DSN details that the ODBC setup window requires and click **Test Connection**. When the successful connection dialog box appears, click **OK** to close the dialog box.
7. Click **OK** to close the ODBC setup window.

   The DSN appears in the **System DSN** tab of the **ODBC Data Source Administrator** window.
8. Note the System Data Source Name property value, which the client application requires to connect to the target database.
9. Click **OK** to close the ODBC Data Source Administrator.

CHAPTER 7

Configuration for MicroStrategy

This chapter includes the following topic:

Configuration for MicroStrategy Overview

MicroStrategy is a business intelligence tool that includes performance management, dashboards, analysis, and reporting capabilities. To capture MicroStrategy user context through Dynamic Data Masking, you must configure MicroStrategy and Dynamic Data Masking.

You configure MicroStrategy so that the name of the user running a report can be included in an SQL query. After you configure MicroStrategy and Dynamic Data Masking, you can take actions on Dynamic Data Masking SQL queries based on user context.

For information about capturing MicroStrategy user context through Dynamic Data Masking, see the H2L "How to Capture MicroStrategy User Context through Dynamic Data Masking."

Configure Dynamic Data Masking to Capture MicroStrategy User Context

To configure Dynamic Data Masking to capture MicroStrategy user context, create a Java Action and place the Java Action as the first rule in the rule set.

1. Create a Java Action with the following code.

```
/* GetMicroStrategyApplicationUserName.java*/
import com.activebase.content.mask.ContentProcessor;
import com.activebase.contentHandlers.statement.StatementContentHandler;
import com.activebase.logging.TraceFacility;
import com.activebase.rule.RuleContext;
import java.util.Map;
import java.util.regex.Matcher;
import java.util.regex.Pattern;


public class GetMicroStrategyApplicationUserName {

    private static final String regexforApplicationUserName = "(.*)
(MICROSTRATEGY_USERNAME\\s*=\\s*(.*?),)(.*)";
    private static final String MicroStrategy_USER = "MICROSTRATEGY_USERNAME";
    private static final int flags = Pattern.DOTALL | Pattern.MULTILINE |
Pattern.UNICODE_CASE | Pattern.CASE_INSENSITIVE;
    private static final Pattern patternforApplicationUserName =
Pattern.compile(regexforApplicationUserName, flags);
```

```
    public static String execute(RuleContext ctx) {

        StatementContentHandler stmtContenthandler = null;
        if (ctx == null) {
            return null;
        } else {
            ContentProcessor contentProcessor = (ContentProcessor) ctx;
            stmtContenthandler = contentProcessor.getStatementContentHandler();
            if (stmtContenthandler == null) {
                return null;
            }
        }


        String orginalStatement = stmtContenthandler.getStatement();
        Matcher applicationUserNameMatcher =
patternforApplicationUserName.matcher(orginalStatement);
        if (applicationUserNameMatcher.find()) {
            String microStrategyApplicationUserName =
applicationUserNameMatcher.group(3);
            Map symbolTable = stmtContenthandler.getSymbolTable();
            symbolTable.put(MicroStrategy_USER,
microStrategyApplicationUserName.trim());
            String newStatement = applicationUserNameMatcher.group(1) + " " +
applicationUserNameMatcher.group(4);
            return newStatement;
        }

        return null;
    }
}
```

2. Compile the code and create a JAR file. For example, `microStrategyAccelerator.jar`.
3. Configure the JAR file in a Java Action in Dynamic Data Masking.
4. Place the Java Action as the first rule in the rule set.

   After configuring the Java Action, a symbol value for MICROSTRATEGY_USERNAME populates the MicroStrategy user name each time a report runs.
5. Create masking rules based on the symbol MICROSTRATEGY_USERNAME.

CHAPTER 8

Access Control

This chapter includes the following topics:

Access Control Overview

Use Dynamic Data Masking access control to define the operations that a user can perform on Management Console tree nodes and Dynamic Data Masking configuration.

You can set permissions on domain, database, and security rule set nodes to define the users that can edit the nodes.

A Dynamic Data Masking user can be a privileged user or a non-privileged user. The type of user and type of permissions the user has on a node in the Management Console tree determines whether the user can view or make changes to the node.

Privileged User

A privileged user is a user that is connected to the the Dynamic Data Masking Server as the administrator or an LDAP user that belongs to the Dynamic Data Masking administration group.

Privileged users have full access control on the Management Console tree nodes. Privileged users can set privileges and perform any operation on Management Console tree nodes.

Non-Privileged User

A non-privileged user is a user that does not belong to the Dynamic Data Masking administration group.

In the Management Console tree, domain, database, and security rule set nodes have authorization properties. Authorization properties define which operations a non-privileged user can perform on Management Console tree nodes.

Non-privileged users cannot edit the Management Console Dynamic Data Masking Server node or the Server node children, such as service nodes, logger nodes, and appender nodes. A non-privileged user can have ownership, read, or read and write privileges on domain, database, and security rule set nodes in the Management Console tree.

The following table describes the authorizations a non-privileged user can have on a database, domain, or security rule set node in the Management Console tree:

Authorization	Description
Ownership	The LDAP user or group owns the node. A node owner has full access control to the node. A node owner can perform the following operations on the node: - Set Authorizations - Read operations - Write operations
Read	The LDAP user or group has read privileges on the node. A user with read privileges can perform the following operations on the node: - View the node details - View the children of the node Read authorizations are required on the source node for the copy node operation.
Read and Write	The LDAP user or group has read and write privileges on the node. A user with read and write privileges can perform the following operations: - View the node details - View the children of the node - The read privilege is required on the source node for the copy node operation. - Add - Edit database and security rule set node details - Copy (destination node) - Move (source and destination node) - Remove (parent and child nodes) - Edit domain, database, and security set names (parent and child nodes)

Authorization Properties of a Non-Privileged User

A non-privileged user must have authorizations to perform operations on Management Console tree nodes. Because the Management Console tree is a hierarchical tree, for some operations the user must have authorizations on multiple nodes.

The following table describes the operations a non-privileged user can perform on a database, domain, or security rule set node and the authorizations the user must have:

Operation	Authorizations
Add node	Ownership or read and write authorizations on the node. The user that creates the node is the owner of the node.
Copy node	Ownership or read authorizations on the source node and the descendants of the source node. Ownership or read and write privileges on the destination node. The user that creates the node is the owner of the node.
View authorizations	Ownership or read authorizations on the node.
Expand node	Ownership or read authorizations on the node.
View database details	Ownership or read authorizations on the node.
View security rule set details	Ownership or read authorizations on the node.
Move node	Ownership or read authorizations on the source node and the descendants of the source node. Ownership or read and write privileges on the destination node.
Remove node	Ownership or read and write authorizations on the parent node. Ownership or read and write authorizations on the child node.
Edit authorizations	Ownership.
Edit database details	Ownership or read and write authorizations on the node.
Edit security rule details	Ownership or read and write authorizations on the node.
Edit domain name	Ownership or read and write authorizations on the parent node. Ownership or read and write authorizations on the child node.
Edit security rule set name	Ownership or read and write authorizations on the parent node. Ownership or read and write authorizations on the child node.
Edit database name	Ownership or read and write authorizations on the parent node. Ownership or read and write authorizations on the child node.

Authorization Properties of a Newly Created Node

When you create a node in the Management Console tree, the node has default authorization properties.

You can add or copy a Management Console tree node to create a node.

The following table describes the default authorization properties:

Property	Default
Owner	User that creates the node. If the user is logged into Dynamic Data Masking as the administrator, the owner property is empty. The owner property is set to copied nodes and copied child nodes.
Read Privileges	Empty for an add operation. The read privileges property does not change for copied nodes and copied child nodes.
Read and Write Privileges	Empty for an add operation. The read and write privileges property does not change for copied nodes and copied child nodes.

Authorization Properties of a Moved Node

The authorization properties of moved Management Console tree nodes and child nodes do not change when you use the move operation.

Configuring Access Control

Configure access control on a node in the Management Console tree to allow users and LDAP groups to access the node.

1. In the Management Console, select a domain, database, or security rule set node.
2. Click **Tree** > **Authorization**.

   The **Authorize User or Group** window appears.
3. Define a node owner in the owner field and define read and write privileges for users and LDAP groups in the **Authorize User or Group** window.
4. Click **Ok**.

CHAPTER 9

Logs

This chapter includes the following topics:

Logs Overview

A log file maintains a history of events. Dynamic Data Masking creates log files and contains system loggers that record Dynamic Data Masking Server, service, and rule events. You can create custom loggers to log information that you specify in a security rule. Set log levels to determine which loggers send log information.

The Management Console Tree contains logger and appender nodes that create log files. The system loggers create the audit trail, rule, and server logs. You can add custom loggers to the Management Console tree that you use in security rules to specify events that you want to log. You can add appender nodes under loggers to specify how and where to log the event information. Set the Log Level property of the Dynamic Data Masking Server to specify the severity level of the events that you want to log.

You can use the log files to identify a problem with the Dynamic Data Masking Server, and to monitor and troubleshoot problems a Dynamic Data Masking service.

In addition to the system loggers, Dynamic Data Masking creates the following log files:

<year>_<month>.at

Detailed audit trail file. Contains detailed information about changes made within the Management Console. Dynamic Data Masking uses the year and the month that it creates the file to name the file.

DDMError.txt

Logs standard operating system process errors when they occur.

DDMOutput.txt

Logs standard operating system process output when they occur.

You can find log files in the following location: `<Dynamic Data Masking installation>/log`

Note: If you choose the Dynamic Data Masking Management Console installation without the Dynamic Data Masking Server, the installation does not create a log directory.

Audit Trail and Reporting

The Dynamic Data Masking general audit trail and detailed audit trail log files contain information that you can use to verify changes a user made to the Dynamic Data Masking configuration.

The `auditTrail.log` file contains general audit information about changes in the configuration.

The detailed audit trail log file contains comprehensive audit information about modifications to the Dynamic Data Masking configuration properties. Dynamic Data Masking names the detailed audit file according to the year and month that it creates the file. For example, if Dynamic Data Masking creates a detailed audit file in April 2019, it names the file `2019_04.at`. You can use the detailed audit trail log files as input to the audit command and generate audit trail reports. The audit trail report shows all changes made by users for selected objects in the specified time frame.

The detailed audit file contains information about audit trail operations and their sources (the Management Console or the Server Control command line program) for the following Dynamic Data Masking objects.

Database

The detailed audit file contains information about the following audit trail operations and their sources for the Dynamic Data Masking database object:

Audit Trail Operation	Management Console	Server Control
Add	Yes	-
Remove	Yes	-
Copy	Yes	-
Move	Yes	-
Edit	Yes	-
Import	No	Yes
Export	No	Yes

Domain

The detailed audit file contains information about the following audit trail operations and their sources for the Dynamic Data Masking domain object:

Audit Trail Operation	Management Console	Server Control
Add	Yes	-
Copy	Yes	-

Audit Trail Operation	Management Console	Server Control
Move	Yes	-
Edit	Yes	-
Remove	Yes	-

Service

The detailed audit file contains information about the following audit trail operations and their sources for the Dynamic Data Masking service:

Audit Trail Operation	Management Console	Server Control
Add	Yes	-
Remove	Yes	-
Import	Yes	Yes
Export	Yes	Yes
Start	Yes	Yes
Stop	Yes	Yes

Authorization

The detailed audit file contains information about the following audit trail operations and their sources for Dynamic Data Masking authorization:

Audit Trail Operation	Management Console	Server Control
Edit	Yes	-

Host Port Security

The detailed audit file contains information about the following audit trail operations and their sources for Dynamic Data Masking host port security:

Audit Trail Operation	Management Console	Server Control
Edit	Yes	Yes

Server

The detailed audit file contains information about the following audit trail operations and their sources for the Dynamic Data Masking Server:

Audit Trail Operation	Management Console	Server Control
Edit (One of these operations: edit server, edit log level, edit Loggers, or Appenders)	Yes	Yes
Move	Yes	No
Remove	-	Yes
Backup	-	Yes
Restore	-	Yes
Lock	-	Yes
Support	No	Yes
License	Yes	-
Reload	-	Yes
Shutdown	-	Yes
Rename	-	Yes
Password	-	Yes
Network	-	Yes
Port	-	Yes
Login	Yes	Yes
Logout	Yes	Yes

Import in the command line can create or modify an object, Dynamic Data Masking audits it as an import operation. Import in the Management Console can modify an object, Dynamic Data Masking audits it as an edit operation.

Dynamic Data Masking audits Export in the command line as an export operation. Dynamic Data Masking does not audit Export in the Management Console.

Dynamic Data Masking 9.9.1 audit trail reports do not show connection rule or security rule set changes.

Audit Trail Reports

To generate audit trail reports, you use the audit command. You can generate standard or compact audit trail reports that use the detailed audit files as input.

For information about the audit command, see “Audit” on page 142.

Standard Audit Trail Report

The following example shows a sample standard audit trail report:

```
Date                          Address    User Name          Type            Path  Object Name Operation
Result
Thu May 09 02:22:16 IST 2019 127.0.0.1  :: internal user :: OracleDatabase Site  o1          add
success

Attribute                Old Value                      New Value                             Notes
name                                                    o1
ssl                                                     false
ks.keyStore                                             DefaultKeyStore
path                                                    Site\o1
authorizations                                          Authorizations owner=null,
instances.1.instanceName                                read=[null],
instances.1.instanceName                                write=[null]
instances.1.listenerAddress                             ins
instances.1.listenerPort                                add
                                                        1521
```

The following table describes the standard audit trail report fields:

Field	Description
Date	Date of the report in the format: MM/dd/yyyy HH:mm:ss.SS
Address	User host or IP address.
User Name	User name.
Type	Type of object: - Domain - Database - Service - Host Port Security - Server - Appender - Logger
Path	Current location of the object.
Object name	Name of the object.

Field	Description
Operation	Audit trail operation: - Add - Backup - Copy - Edit - Export - Import - License - Lock - Login - Logout - Move - Network - Password - Port - Reload - Remove - Rename - Restore - Shutdown - Start - Stop - Support - Unlock
Result	Status of the operation: - success - failure
Attribute	Attribute that was changed by the operation.
Old value	Previous value of the attribute. Depending on the operation, an attribute might have an old value or a new value, or both.
New value	New value of the attribute. Depending on the operation, an attribute might have an old value or a new value, or both.
Notes	Contains data from the audit trail log reference field and any warning messages.

Compact Audit Trail Report

The following example shows a sample compact audit trail report:

```
    Date                         Address    User Name           Type           Path  Object Name Operation Result    Notes
    Thu May 09 02:22:16 IST 2019 127.0.0.1  :: internal user :: OracleDatabase Site  o1          add       success
```

The following table describes the compact audit trail report fields:

Field	Description
Date	Date of the report in the format: MM/dd/yyyy HH:mm:ss.SS
Address	User host or IP address.
User Name	User name.

Field	Description
Type	Type of object: - Domain - Database - Service - Host Port Security - Server - Appender - Logger
Path	Current location of the object.
Object name	Name of the object.
Operation	Audit trail operation: - Add - Backup - Copy - Edit - Export - Import - License - Lock - Login - Logout - Move - Network - Password - Port - Reload - Remove - Rename - Restore - Shutdown - Start - Stop - Support - Unlock
Result	Status of the operation: - success - failure
Notes	Contains data from the audit trail log reference field and any warning messages.

Loggers

A logger is a Management Console tree node that uses Apache log4j to create a log of events.

You can add logger nodes in the Management Console tree under the Loggers node.

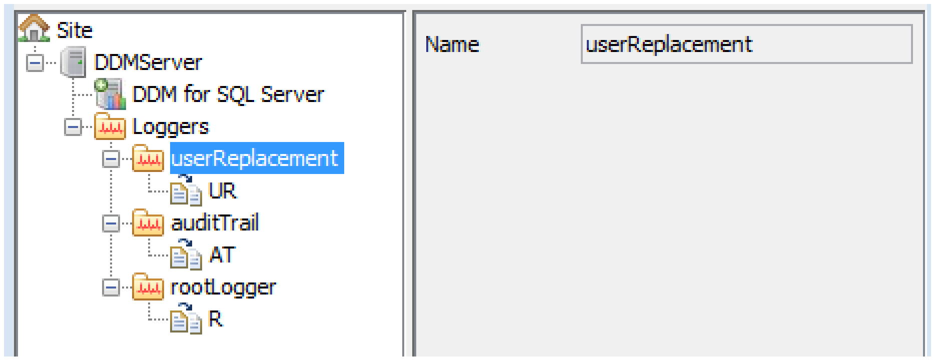
Use loggers to specify the events that you want to log and use appenders to define how to log the event. You can use pre-defined system loggers to log Dynamic Data Masking Server, service, and rule events. You can create custom loggers to log security rule events that you specify with the Log Message rule action.

Logger nodes can have multiple appender child nodes. When you use the logger in a security rule, the logger logs the event in each format specified by the child appender nodes.

Dynamic Data Masking contains pre-defined system loggers and appenders that log Dynamic Data Masking service events and rule events. You cannot edit or delete the system loggers.

The Loggers node is a child of the Dynamic Data Masking Server node in the Management Console tree. Because it is a child of the Dynamic Data Masking Server node, only Dynamic Data Masking administrators can edit and create child nodes of the Loggers node. Non-privileged users cannot edit or move logger and appender nodes and an administrator cannot delegate permissions to a non-privileged user.

The following image shows the Loggers node and the child system logger nodes:



Name: userReplacement

System Loggers

A system logger is a pre-defined logger node in the Management Console tree that logs Dynamic Data Masking Server, service, and rule events.

The Management Console tree contains userReplacement, auditTrail, and rootLogger system loggers. The system loggers use Rolling File appenders to create the audit trail, rule, and server logs. You cannot delete or move the system loggers or the appenders. You can edit the Max File Size and Max Backups properties of the system logger appenders, but you cannot edit the Type, Name, and File properties. If you edit a system logger appender, Dynamic Data Masking immediately reconfigures config.properties and saves the file.

You can add appenders to the system loggers to log the same information in different formats.

The system loggers create the following log files:

auditTrail.log

Logs changes made within the Management Console. The AT appender of the auditTrail logger creates the `auditTrail.log` file.

rule.log

Logs rules that the Rule Engine applies to incoming requests. In the Management Console, you can use the Log When Rule is Applied box in the **Edit Rule** window to specify whether an occurrence of the rule is logged. The UR appender of the userReplacement logger creates the `rule.log` file.

If multiple rule log files exist, Dynamic Data Masking appends each file name with a version number, such as `rule.log1`. Dynamic Data Masking stores 10 rule log files by default. Rule logs update cyclically and restart on `rule.log1` when the logs are full.

By default, each rule log file stores up to 20 MB of data for a total of 200 MB. You can configure file size and the maximum number of files in the UR appender.

Note: In high transaction volume applications, specify additional rule logs carefully due to the increased overhead.

You can use the Log Loader utility to load rule.log data into an Oracle, Db2, Informix, or Microsoft SQL Server database. See *Informatica Dynamic Data Masking Log Loader* for information on the Log Loader utility.

server.log

Logs server records, events, and error messages for internal troubleshooting of the Dynamic Data Masking Server operations. The R appender of the rootLogger logger creates the server.log file.

If multiple server log files exist, Dynamic Data Masking appends each file name with a version number, such as `server.log1`. Dynamic Data Masking stores up to 10 server log files at a time. Server logs update cyclically and restart on `server.log1` when the logs are full.

By default, each rule log file stores up to 20 MB of data for a total of 200 MB. You can configure file size and the maximum number of files in the UR appender.

Sample Rule.log File

The following excerpt is from the rule log:

```
05/30 16:55:39,240 [MASK@ERP-1] INFO  - Blocking Rule: Identify
Blocked Statement (user message: This request has been blocked.)
select * from customer
Done by ClientInfo:[User=Admin, Host=ADMIN-THINK, application=JDBC Thin Client] -
SessionID=74,2834 - SYSTEM - Instance 1

05/30 16:57:29,156 [sapiens@ERP-1] INFO  - None Rule: auditlog
BEGIN DBMS_OUTPUT.DISABLE; END;
Done by ClientInfo:[User=Admin, Host=ADMIN-THINK, application=C:\app\Admin\product
\11.2.0\dbhome_2\bin\sqlplus.exe] - SessionID=73,763 - system - Instance 1
```

Sample Server.log File

The following excerpt is from the server log:

```
05/23 13:16:00,075 [pool-1-thread-1] INFO  - Service started.
05/23 13:16:00,077 [main] INFO  - Service DDM for DB2 started
05/23 13:16:00,077 [main] WARN  - DDM for Oracle.configure: Invalid address provided:
null:0
05/23 13:16:00,079 [main] INFO  - Service DDM for Oracle started
05/23 13:16:00,116 [main] INFO  - Server started.
05/23 13:16:00,198 [Thread-3] INFO  - Service DDM for SQL Server started
05/23 18:36:43,989 [Thread-2] WARN  - ProcessService: ProcessService: restarting process.
05/23 18:36:44,005 [Thread-4] INFO  - Service DDM for SQL Server started
```

Custom Loggers

A custom logger is a logger that you create to use in a security rule to define events that you want to log.

Loggers have a Name property that you define when you create the logger. Logger names must be unique. When you create a security rule with the Log Message rule action, you specify the name of the logger in the rule and define the log level in the rule. The logger creates logs based on the appender child nodes of the logger. The log level that you define in the Send As parameter of the security rule must be equal to or higher than the log level that you define in the Dynamic Data Masking Server node. If the log level of the rule is a lower severity than the Dynamic Data Masking Server Log Level parameter, the logger will not log the event.

After you create a logger and add an appender, you must use the Log Message rule action in a security rule to define the events that you want the logger to log. If you do not use the logger in a security rule, it does not log events.

Because logger nodes are child nodes of the Dynamic Data Masking Server node, only a Dynamic Data Masking administrator can create or edit a logger node. An administrator cannot delegate privileges to a non-privileged user.

Note: Do not use system loggers with the Log Message rule action because you might not be able to perform log analysis on the logs if they contain information from security rules.

Creating a Custom Logger

To define the events that you want to log, create a custom logger that you use in a security rule .

1. In the Management Console tree, select the Loggers node and click **Tree** > **Add Logger**.

   The **Add Logger** window appears.
2. Enter the name of the logger. The logger name can contain alphabetic characters, numbers, and underscores (_), and must not exceed 60 characters.
3. Click **OK**.

   The logger appears in the Management Console tree.

After you create a logger, you must add appenders to the logger to specify the log output format. You must use the logger in a security rule with the Log Message rule action to define the events that you want to log.

Loggers Example

Your organization uses syslog to integrate log data from multiple types of systems. You want to add Dynamic Data Masking logs to the syslog repository.

You add appenders to the userReplacement, auditTrail, and rootLogger system loggers. The appenders use the syslog appender class. When Dynamic Data Masking writes to the auditTrail.log, rule.log, and server.log files, it also creates a syslog output.

You create custom loggers and you add syslog appenders to the loggers. You use the loggers in security rules based on the events that you want to log. Dynamic Data Masking sends the event data to the syslog repository.

Appenders

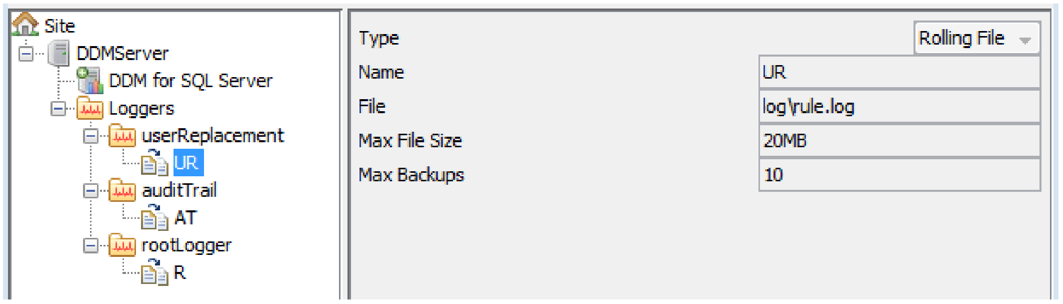
An appender is a node in the Management Console tree that uses log4j classes to define the output format of log information.

You can create appenders to log information in a format that is useful to your organization. You can use the built-in Rolling File, Syslog, SMTP, and SNMP appenders, or create a custom appender to store log information in any format. A logger can have multiple appenders.

Appenders are child nodes of logger nodes in the Management Console tree. Only administrators can create and edit appender nodes. An administrator cannot delegate appender permissions to a non-privileged user.

The Dynamic Data Masking system loggers use Rolling File appenders to create the rule.log, auditTrail.log, and server.log files. You cannot delete the system logger appenders. You can edit the system logger appender Max File Size and Max Backups properties, but you cannot edit the Type, Name, and File properties. You can add an appender to the system loggers to create an additional system log output.

The following image shows the system logger appenders in the Management Console tree and the UR appender properties:



Type	Rolling File
Name	UR
File	log\rule.log
Max File Size	20MB
Max Backups	10

Rolling File Appender

Create a Rolling File appender to log information to a text file.

Rolling File appenders create plain text output files. You can use any plain text extension, such as .log or .txt.

The File property contains the file path and name of the log file. If you do not specify a complete file path, the path originates in the Dynamic Data Masking installation directory. For example, the rule.log file has the following File property:

```
log\rule.log
```

The Max File Size and Max Backups properties prevent the log files from becoming too large. When the log reaches the Max File Size, the logger creates a new log file. When the number of files exceeds the Max Backups number, the logger overwrites the first log file that it created. You can specify Max File Size and Max Backups in megabytes or gigabytes by entering the number followed by MB or GB. Do not insert a space between the value and the unit of measure.

Rolling File Appender Properties

The following table describes the Rolling File appender properties:

Property	Description
Name	The name of the appender. The appender name can contain alphabetic characters, numbers, and underscores (_), and must not exceed 60 characters.
File	The file path and name of the log file. If you do not specify a complete file path, the path originates in the Dynamic Data Masking installation directory. The file path and name cannot exceed 60 characters.
Max File Size	The maximum size that the output file reaches before the appender creates a new file. Default is 10MB. **Note:** Do not enter a space between the value and the unit of measure.
Max Backups	The number of backup files the appender creates before it overwrites the oldest file. Max Backups must be a positive integer. Default is 20.

Syslog Appender

Create a Syslog appender to log information to a syslog repository.

If your organization uses a syslog system to store log information, you can use the Syslog appender to log events. You can add Syslog appenders to the system loggers to log standard Dynamic Data Masking output files to the syslog repository in addition to the Dynamic Data Masking log directory.

Syslog Appender Properties

The following table describes the Syslog appender properties:

Property	Description
Name	The name of the appender. The appender name can contain alphabetic characters, numbers, and underscores (_), and must not exceed 60 characters.
Syslog Host	The host name or IP address of the Syslog server.
Facility	The Syslog facility. A system administrator can set Facility to one of the following strings: KERN, USER, MAIL, DAEMON, AUTH, SYSLOG, LPR, NEWS, UUCP, CRON, AUTHPRIV, FTP, LOCAL0, LOCAL1, LOCAL2, LOCAL3, LOCAL4, LOCAL5, LOCAL6, LOCAL7. Default is USER.
Conversion Pattern	The conversion pattern for the Syslog appender that you can modify. By default, the pattern is: `%d{MM/dd HH\:mm\:ss,SSS} [%t] %-5p - %m%n`

SMTP Appender

Create an SMTP appender to send log information as an email.

When you create an SMTP appender, the logger sends log events as an email. You can specify multiple email addresses that you want to receive the email. For example, you can use the SMTP appender to send an email when an error or fatal error occurs.

Before you use the SMTP appender, you must put the mail.jar file into the `<Dynamic Data Masking installation>/lib/ext` directory. On Windows, you must remove and restart the Dynamic Data Masking Server to update the Server configuration in the Windows Registry. On Unix, you must restart the Dynamic Data Masking Server.

SMTP Appender Properties

The following table describes the SMTP appender properties:

Property	Description
Name	The name of the appender. The appender name can contain alphabetic characters, numbers, and underscores (_), and must not exceed 60 characters.
From	The email address of the sender.

Property	Description
To	List of recipient email addresses, separated by commas.
Subject	The subject line of the email.
SMTP Host	The SMTP relay mail server to use to send the email.
SMTP Username	The username to authenticate the SMTP server.
SMTP Password	The password to authenticate the SMTP server.
Debug	Enable Debug to analyze the standard system output if an error occurs, such as an SMTP login failure.

Configuring the SMTP Appender

Before you use the SMTP appender, you must place the SMTP jar file into the Dynamic Data Masking directory. On Windows, remove and restart the Dynamic Data Masking Server. On Unix, restart the Dynamic Data Masking Server.

To configure the SMTP appender, you need the following file:

- mail.jar version 1.4.2

1. Place the mail.jar file into the following directory:
   ```
   <Dynamic Data Masking installation>/lib/ext
   ```
2. Close the Management Console.
3. Open a Server Control window.
   - If the Dynamic Data Masking Server runs on Windows, enter the following commands:

     - `server stop`

     - `server remove`

     - `server start`
   - If the Dynamic Data Masking Server runs on Unix, enter the following command:

     - `server restart`
4. Open the Management Console and create the SMTP appender.

Note: If you add, remove, or update the files in the `<Dynamic Data Masking installation>/lib/ext` directory, you must configure the SMTP appender again.

SNMP Appender

Create an SNMP appender to send log information by using SNMP protocol.

Simple Network Management Protocol (SNMP) is a standard protocol for managing devices on IP networks. Devices that typically support SNMP are routers, switches, servers, workstations, printers, and modem racks.

When you create an SNMP appender, you define the Generic Trap Type as a numeric value. You can find information on trap types at http://publib.boulder.ibm.com/infocenter/zvm/v5r4/index.jsp?topic=/com.ibm.zvm.v54.kijl0/trp.htm.

Before you use the SNMP appender, you must put the SNMPTrapAppender and OpenNMS jar files into the `<Dynamic Data Masking installation>/lib/ext` directory. On Windows, you must remove and restart the

Dynamic Data Masking Server to update the Server configuration in the Windows Registry. On Unix, you must restart the Dynamic Data Masking Server.

SNMP Appender Properties

The following table describes the SNMP appender properties:

Property	Description
Name	The name of the appender. The appender name can contain alphabetic characters, numbers, and underscores (_), and must not exceed 60 characters.
Server Host	The host name or IP address of the SNMP server.
Server Trap Listen Port	The SNMP server port. Default is 161.
Enterprise OID	The object ID of the organization that sends the trap message. Set this parameter to any value to identify the message.
Local IP Address	The IP address of the local SNMP embedded agent. Default is 127.0.0.1.
Local Trap Send Port	The port of the local SNMP embedded agent.
Generic Trap Type	A number value that specifies the trap type. Set the Generic Trap Type to one of the following values: - 0. coldStart - 1. warmStart - 2. linkDown - 3. linkUp - 4. authenticationFailure - 5. egpNeighborLoss - 6. enterpriseSpecific
Specific Trap Type	The trap description.
SNMP Community	The name of the SNMP community. Default is public.
Forward Stack Trace with Trap	Specifies whether to include the stack trace as part of the log message.
Application Trap OID	The object ID of the application that sends the trap message. Enter the name of the application object.

Configuring the SNMP Appender

Before you use the SNMP appender, you must place the SNMP jar files into the Dynamic Data Masking directory. On Windows, remove and restart the Dynamic Data Masking Server. On Unix, restart the Dynamic Data Masking Server.

To configure the SNMP appender, you need the following files:

- SNMPTrapAppender_1_2_9.jar

- opennms-joesnmp-20031201-173122.jar

1. Place the SNMPTrapAppender and the OpenNMS jar file into the following directory:

   ```
   <Dynamic Data Masking installation>/lib/ext
   ```

2. Close the Management Console.
3. Open a Server Control window.
   - If the Dynamic Data Masking Server runs on Windows, enter the following commands:

     - `server stop`
     - `server remove`
     - `server start`
   - If the Dynamic Data Masking Server runs on Unix, enter the following command:

     - `server restart`
4. Open the Management Console and create the SNMP appender.

Note: If you add, remove, or update the files in the `<Dynamic Data Masking installation>/lib/ext` directory, you must configure the SNMP appender again.

Creating an Appender

Create an appender to specify the format of the log information.

1. In the Management Console tree, select a logger node and click **Tree** > **Add Appender**. You can add an appender to a system logger or to a custom logger.

   The **Add Appender** window appears.
2. Select the type of appender that you want to create. The appender properties change based on the appender that you choose.
3. Enter the appender properties and click **OK**.

   The appender appears as a child node of the logger node that you selected.

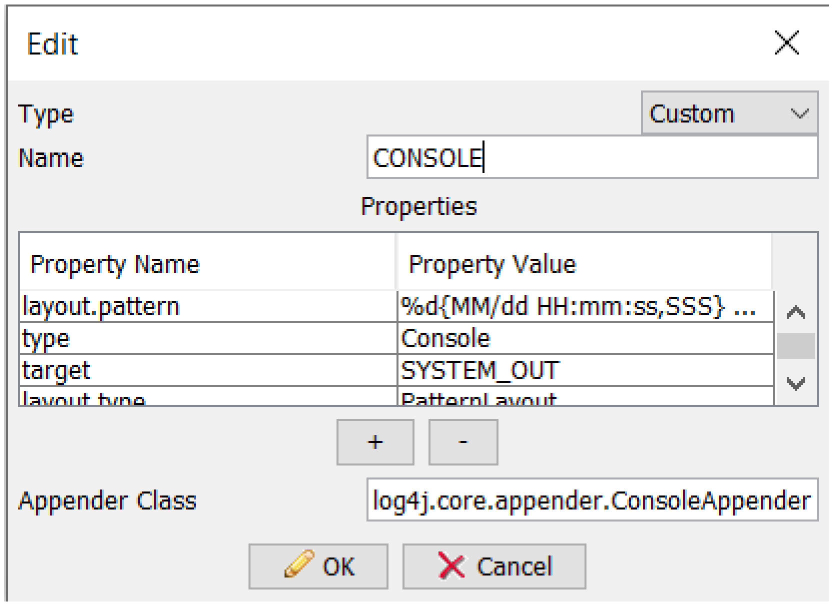
Custom Appender

Create a custom appender to log events in a custom format by using log4j classes.

A custom appender can use any log4j appender class and you can specify multiple properties for the appender.

The built-in appenders have hidden properties that you cannot modify. A custom appender is an appender that either has a class that does not match a built-in appender class or a property that does not match the hidden properties of the built-in class. For example, the Rolling File appender has a hidden encoding property set to UTF-8. You can create a custom rolling file appender that has a different encoding property.

The following image shows a custom appender that uses the console appender class:


Edit

Type	Custom
Name	CONSOLE

Properties

Property Name	Property Value
layout.pattern	%d{MM/dd HH:mm:ss,SSS} ...
type	Console
target	SYSTEM_OUT
layout.type	PatternLayout

Appender Class	log4j.core.appender.ConsoleAppender

You can specify the name, layout pattern, type, target, layout type, and appender class properties in the custom appender.

Custom Appender Properties

The following table describes the Custom appender properties:

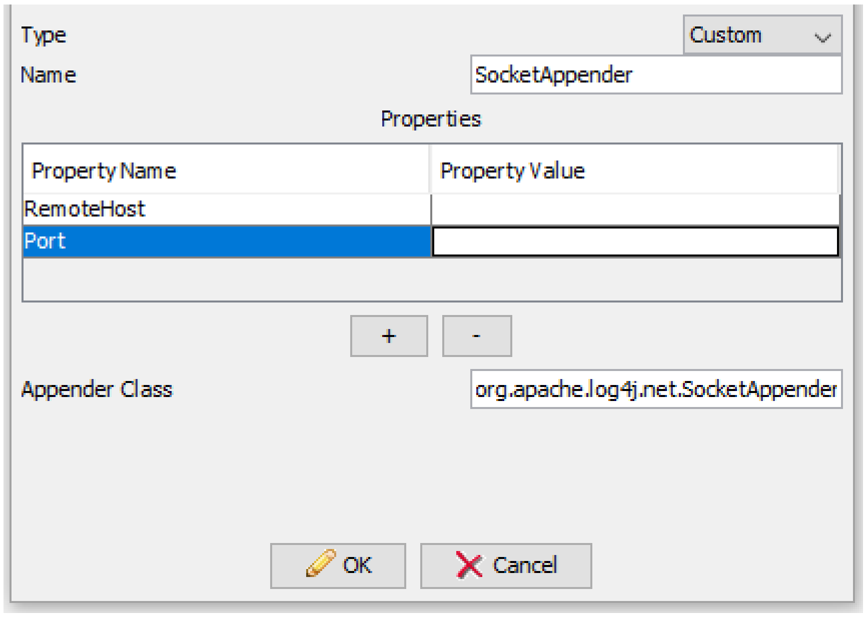
Property	Description
Name	The name of the appender. The appender name can contain alphabetic characters, numbers, and underscores (_), and must not exceed 60 characters.
Property Name	The name of the property.
Property Value	The value of the property. Property Value can be any value that is represented as a string.
Appender Class	The log4j2 appender class and package.

Sending User Activity to Secure@Source

To send user activity to Secure@Source, create a custom appender using the SocketAppender class. After you create the custom appender, create a security rule using the Any Matcher and the Log Message Action.

1. Create an appender and specify the type as Custom. Specify the RemoteHost, Port, and Appender Class.

The following image shows a custom appender that uses the SocketAppender class:


Type: Custom
Name: SocketAppender

Properties

Property Name	Property Value
RemoteHost	
Port	

Appender Class: org.apache.log4j.net.SocketAppender

The following table describes the required properties:

Property	Description
RemoteHost	The host name of the server.
Port	The port where the server is waiting for connection.
Appender Class	The log4j appender class and package. To send user activity to Secure@Source, use the SocketAppender class: org.apache.log4j.net.SocketAppender

2. Click OK.

Create a security rule using the Any Matcher and the Log Message Action. Specify the Logger Message using the CEF format. For more information, see the *Informatica Dynamic Data Masking 9.9.1 User Guide*.

Log Levels

The log level that you define in the Dynamic Data Masking Server node determines the severity of the event that you want to log. You specify a log level in a security rule to define the severity of individual events.

The Dynamic Data Masking Server node contains a Log Level property that the Dynamic Data Masking administrator can set to Information, Warning, or Error. The log level in the Dynamic Data Masking Server node corresponds to the Send As property of the Log Message security rule action. If the Send As property is an equal or greater severity than the Log Level property, the logger logs the event. If the Send As property is a lower severity than the Log Level property, the logger does not log the event.

For example, you have the Log Level property of the Dynamic Data Masking Server set to Warning. You have three security rules that use the Log Message rule action. The Send As property of Rule_1 is set to

Information. Rule_2 has the property set to Warning, and Rule_3 has the property set to Error. The loggers will create logs for Rule_2 and Rule_3, but not for Rule_1.

The following table describes the log levels:

Log Level	Description
Information	Provides information about the normal behavior of the Dynamic Data Masking Server. Information logs can include information about when a service starts or stops and when a user logs in or if the log in fails.
Warning	Provides information that you can use to find and analyze non-fatal abnormal states in the Dynamic Data Masking Server. Warning logs can include information about a Dynamic Data Masking service start or stop failure, or an error that occurs when you add a node in the Management Console tree.
Error	Provides only error messages. Use the Error log level in production because it provides the best Dynamic Data Masking performance.

Setting the Log Level

Set the log level to specify the severity of the events that you want to log.

1. In the Management Console tree, select the Dynamic Data Masking Server node and click **Tree** > **Edit**.

   The **Edit** window appears.
2. Configure the **Log Level** property to the level that you want to log.
3. Click **OK**.

JDBC Logging

If you use the DDM for JDBC service, you can configure logging on the client machine to debug the JDBC wrapper.

The Dynamic Data Masking installation contains a template configuration file for JDBC logging. You can find the template in the following location:

```
<Dynamic Data Masking installation>\Wrappers\jdbc\template.jdbcLogConfig.properties
```

The template.jdbcLogConfig.properties file contains the following properties:

ddm.logfile.name

Specifies the file name of the log. Modifiable.

Default value: %h/DDM_GENJDBC.log

ddm.logging.level

Specifies the log level. Modifiable.

You can specify the following log levels:

- ERROR. Provides the complete details of all errors and exceptions that affect the Dynamic Data Masking processes.

- WARN. Provides information about exceptions that do not affect the Dynamic Data Masking processes, but do affect Dynamic Data Masking functionality.
- INFO. Provides information about the normal behavior of the Dynamic Data Masking Server, such as when a service starts or stops and when a user logs in.
- DEBUG. Provides information for debugging issues, such as client connection details to the database and the Dynamic Data Masking Server, connection rule results, and original and modified requests to the database.

Default value: INFO

ddm.logfile.limit

Specifies the maximum size of the log file before the logger creates a new file. For example, you might enter 500KB or 200MB. A value of zero (0) indicates that there is no file size limit. Modifiable.

Default value: 0

ddm.logfile.count

The number of backup files the appender creates before it overwrites the oldest file. Modifiable.

Default value: 1

ddm.logfile.append

Specifies whether you want to overwrite the log file when you start the application. A value of false indicates that you want to overwrite the log file with a new file. A value of true indicates that you want to add new log information to the existing log file.

ddm.logfile.encoding

The log file character encoding system. Modifiable.

Default value: UTF-8

ddm.log.format

Specifies the logging format. Not modifiable.

Default value: %1$tm/%1$td %1$tH:%1$tM:%1$tS,%1$tL [%2$s] %4$s - %5$s %6$s %n

The following properties are used in the configuration of the log file name and are replaced at runtime:

- %t. The system temporary directory. For example, `%t/DDM_GENJDBC.log` causes the logs on Solaris to be written to `/var/tmp/DDM_GENJDBC.log` and the logs on Windows 7 to be written to `C:\Users\<username>\AppData\Local\Temp`.
- %h. The value of the user.home system property.
- %g. The generation number to distinguish rotated logs. If the file count is greater than one, the generation number is added to the end of the file name.
- %u. A unique number used to resolve conflicts. If FileHandler tries to open the file when the file is in use by another process, it increments the unique number and tries again until it finds a file name that is not in use.
- %%. Translates to a percent sign (%).

Note: When you modify the JdbcLogConfig.properties file, you must restart the application for the new properties to take effect.

The following text is an example of a JDBC configuration file:

```
########################   description of special components for log file name starts
########################

#   %t     The system temporary directory     "%t/DDM_GENJDBC.log" would typically cause
```

```
log files to be written on Solaris to /var/tmp/DDM_GENJDBC.log whereas on Windows 7 they
would be typically written to C:\Users\<username>\AppData\Local\Temp
#   %h     The value of the "user.home" system property
#   %g     The generation number to distinguish rotated logs. If the file count is
greater than one, then the generation number will be added to the end of the generated
filename.
#   %u     A unique number to resolve conflicts. FileHandler tries to open the filename
and finds the file is currently in use by another process it will increment the unique
number field and try again. This will be repeated until FileHandler finds a file name
that is not currently in use.
#   "%%     %%" translates to a single percent sign "%"

########################    description of special components for log file name
ends      ########################


# log file name and location
ddm.logfile.name=%h/DDM_GENJDBC.log
# level for ddm log file
ddm.logging.level=INFO
# max size of log file (in bytes). 0 means unlimited
ddm.logfile.limit=0
# max number of log files ( minimum is 1)
ddm.logfile.count=1
# whether log file gets appended after restart of JVM. (false means old logs gets
deleted)
ddm.logfile.append=true
#encoding of log file
ddm.logfile.encoding=UTF-8
# log format
ddm.log.format=%1$tm/%1$td %1$tH:%1$tM:%1$tS,%1$tL  [%2$s]  %4$s - %5$s  %6$s   %n
```

Configure JDBC Logging

To enable logging for an application or process that uses the DDM for JDBC service, you must save the configuration file and create a VM argument.

1. Copy the template.jdbcLogConfig.properties file. You can find the file in the following location:

   ```
   <Dynamic Data Masking installation>\Wrappers\jdbc\template.jdbcLogConfig.properties
   ```

2. Save the file to a directory that is accessible to the JVM.
3. Rename the file that you saved in the previous step to the following name:

   ```
   JdbcLogConfig.properties
   ```

4. Create a VM argument and set the value of the argument to the file path and file name of the JdbcLogConfig.properties file you saved in the previous step.

   Create the following VM argument:

   ```
   o    -DDM_GENJDBC_LOG= <config file name and location>
   ```

   For example, you might create the a VM argument with the following value:

   ```
   -DDDM_GENJDBC_LOG =C:\DDM_GenJDBC\ JdbcLogConfig.properties
   ```

ODBC Logging

If you use the DDM for ODBC service, you can configure logging on the client machine to debug the Driver Manager proxy.

Dynamic Data Masking ODBC logging uses Apache log4cxx. The log4cxx library is consumed as a static library and is packed inside the Drive Manager proxy DLL to avoid conflict with the client application libraries. To enable logging, set up a user-level environment variable that points to a log4cxx configuration file.

The Dynamic Data Masking installation contains a template configuration file for ODBC logging. You can find the template in the following location:

```
<Dynamic Data Masking installation>\Wrappers\odbc\template.odbcLogConfig.properties
```

The following text is an example of an ODBC configuration file:

```
##########    Begin of non-modifiable properties section    ##########

log4j.appender.ddmGenODBCFileAppender=org.apache.log4j.RollingFileAppender
log4j.appender.ddmGenODBCFileAppender.layout=org.apache.log4j.PatternLayout
log4j.appender.ddmGenODBCFileAppender.layout.ConversionPattern=%d{MM/dd HH\:mm\:ss,SSS}
%-10X{processID} %-10t %X{app} (%F:%L) %-5p - %m%n


##########    End of non-modifiable properties section    ##########


##########    Begin of modifiable properties section    ##########

#only permitted to change the log level from INFO to either OFF or DEBUG, do not modify/
remove the ddmGenODBCFileAppender name.
log4j.rootLogger=INFO, ddmGenODBCFileAppender

#point the File property to the complete file path including the folder location and
prefixing the name provided to the log file name and adding the process id as a suffix
to the file name in order to support different instances of the same application running
at same time. Hence, do not remove the ${processID} suffix.
log4j.appender.ddmGenODBCFileAppender.File=C:\\logs\\Aquadata-${processID}.log

#Specify the size of each log file for rolling purpose.
log4j.appender.ddmGenODBCFileAppender.MaxFileSize=20MB

#Specify the max number of bacup rolling files to maintain.
log4j.appender.ddmGenODBCFileAppender.MaxBackupIndex=5

##########    End of modifiable properties section    ##########
```

Set the odbcLogConfig.properties Parameters

You can set the log level, log file location, log file size, number of log files in the odbcLogConfig.properties file.

The file contains the following properties:

log4j.appender.ddmGenODBCFileAppender

Specifies the rolling file appender type. Not modifiable.

Default value: org.apache.log4j.RollingFileAppender

log4j.appender.ddmGenODBCFileAppender.layout

Specifies that the rolling file appender uses a pattern layout. Not modifiable.

Default value: org.apache.log4j.PatternLayout

log4j.appender.ddmGenODBCFileAppender.layout.ConversionPattern

Specifies the logging format for the file appender. Not modifiable.

Default value: =%d{MM/dd HH\:mm\:ss,SSS} %-10X{processID} %-10t %X{app}(%F:%L) %-5p - %m%n

log4j.rootLogger

Specifies the log level and the appender for the root logger. Do not change the file appender name.

You can specify the following log levels:

- INFO. Provides information about the normal behavior of the Dynamic Data Masking Server. Information logs can include information about when a service starts or stops and when a user logs in or if the login fails.
- DEBUG. Provides information for debugging issues, such as client connection details to the database and the Dynamic Data Masking Server, connection rule results, and original and modified requests to the database.
- OFF. Turns off logging.

Default value: INFO, ddmGenODBCFileAppender

Example value: DEBUG, ddmGenODBCFileAppender

log4j.appender.ddmGenODBCFileAppender.File

Specifies the file name and path of the log file. Modifiable.

Default value: <Folder path>\\<File name prefix>-${processID}.log

Example value: C:\\Drive\\Work\\Log4cxx\\Aquadata-${processID}.log

Note: When you enter the path, you must include the -${processID}.log text after the file name. In the example above, the file name is Aquadata.

log4j.appender.ddmGenODBCFileAppender.MaxFileSize

Specifies the maximum size of the log file before the logger creates a new file. Modifiable.

Default value: <Max log file size ex: 500KB, 20MB etc>

Example value: 500KB

log4j.appender.ddmGenODBCFileAppender.MaxBackupIndex

The number of backup files the appender creates before it overwrites the oldest file.

Example value: 20

Configure ODBC Logging

To enable logging for an application or process that uses the DDM for ODBC service, you must set an environment variable before the application or process launches.

If you can launch the application through a .bat file, add the following line to the .bat file:

```
SET DDM_GENODBC_LOG=<log configuration file path>
```

For example, you might add the following text:

```
SET DDM_GENODBC_LOG= C:\logs\logConfig.properties
```

If you cannot launch the application through a .bat file, complete the following steps before you launch the application:

1. Define the DDM_GENODBC_LOG user-level environment variable. Set the value to the application-specific configuration file that contains the log file path and log configuration for the application. The environment variable has the following properties:

   Variable name

   DDM_GENODBC_LOG

   Variable value

   The file path of the logging configuration file for the application. For example, you might enter the following text:

   ```
   C:\DDMLoggingConfigs\ CognosLog4cxxConfig.properties
   ```
2. Start the application and test the ODBC connectivity. Verify that odbc32.dll is used and the configuration file is read.
3. Remove the DDM_GENODBC_LOG environment variable or delete the variable value to ensure that the variable does not affect applications that you start later.

CHAPTER 10

High Availability

This chapter includes the following topics:

High Availability Overview

High availability refers to the uninterrupted availability of computer system resources. When you configure Dynamic Data Masking, you might configure high availability for the Dynamic Data Masking Server, high availability for the database that the Dynamic Data Masking Server connects to, or both.

Informatica recommends that you use standard vendor software to implement high availability. However, if you do not have high availability configured for your database, you can configure Dynamic Data Masking high availability. To send requests to a secondary database if the primary database is unavailable, configure high availability for the database. To send requests through a secondary Dynamic Data Masking Server if the primary Dynamic Data Masking Server is unavailable, configure high availability for the Dynamic Data Masking Server.

Database High Availability

Database high availability sends requests to a database based on whether a connection to the database exists. If you do not have a standard database high availability solution configured for the database, you can configure high availability in Dynamic Data Masking.

Create connection rules that use the Check Database Connection matcher to send requests to a primary database if a connection exists, and to a secondary database if the primary database connection does not exist. When you create the connection rule, you specify a database in the rule matcher and Dynamic Data Masking verifies whether a connection to the database exists.

For more information about the Check Database Connection Matcher, see the *Dynamic Data Masking User Guide*.

Configuring Database High Availability

Configure database high availability to send requests to a secondary database if the primary database is unavailable.

1. In the Management Console, create a database node with a connection to the primary database and a database node with a connection to the secondary database.
2. Add the Dynamic Data Masking service for the database type.
3. Open the connection rule set for the Dynamic Data Masking service and create a connection rule that sends requests to the primary database if a connection to the database exists. Configure the following properties for the connection rule:

   Matcher

   Select the Check Database Connection matcher. The Check Database Connection matcher verifies whether a connection to the database exists.

   Database

   Enter the name of database node for the primary database.

   Rule Action

   Select the Switch to Database action. The Switch to Database action sends the request to the database if the matcher identified a connection.

   Database

   Enter the name of database node for the primary database.

   Processing Action

   Select Stop if Applied. The Rule Engine does not continue to the next rule in the tree if Dynamic Data Masking applied the Switch to Database rule action.
4. In the connection rule set, create a rule that sends the request to the secondary database if a connection to the primary database does not exist. Configure the following properties for the connection rule:

   Matcher

   Select the Check Database Connection matcher.

   Database

   Enter the name of the database node for the secondary database.

   Rule Action

   Select the Switch to Database action.

   Database

   Enter the name of the database node for the secondary database.

   Processing Action

   Select Stop if Applied.
5. Save the connection rule set.

If a connection to the primary database exists, Dynamic Data Masking sends the request to the primary database. If a connection to the primary database does not exist, Dynamic Data Masking sends the request to the secondary database.

Dynamic Data Masking Server High Availability

Dynamic Data Masking Server high availability sends requests through a secondary Dynamic Data Masking Server if the primary Dynamic Data Masking Server is unavailable.

Informatica recommends that you use standard vendor solutions to provide Dynamic Data Masking high availability. For example, you might use failover clustering for Microsoft SQL Server.

To configure Dynamic Data Masking Server high availability for Db2, you can create connection rules that use the Load Control rule action. You must have at least two Dynamic Data Masking Server installations and the database client must use the IBM JDBC driver for Db2. If the Dynamic Data Masking Server installations are on the same machine, they must use different listener ports. After you configure Dynamic Data Masking Server high availability, you can connect to the database through either of the Dynamic Data Masking Servers. If one of the servers is unavailable, the request goes through the other server.

Configuring Dynamic Data Masking Server High Availability for Db2

Configure Dynamic Data Masking Server high availability to send requests through a secondary Dynamic Data Masking Server if the primary Dynamic Data Masking Server is unavailable.

Verify the following prerequisites before you configure Dynamic Data Masking Server high availability for Db2:

- You must have a Db2 database.
- You must have two Dynamic Data Masking Servers installed.
- You must have the IBM JDBC driver.

1. In the Management Console for the primary Dynamic Data Masking Server, create a connection to the Db2 database. The database client must use the IBM JDBC driver to connect to the listener port that you define for the connection.

   Note: If the Dynamic Data Masking Server installations are on the same machine, you must configure different listener port numbers for the primary and secondary servers.
2. Add the Dynamic Data Masking service for DB2.
3. Open the connection rule set for the Dynamic Data Masking service and create a rule folder that identifies requests to the database. Configure the following properties for the rule folder:

   Matcher

   Select the Incoming DDM Listener Port matcher. The Incoming DDM Listener Port matcher identifies requests based on the incoming listener port.

   Incoming Port

   Enter the listener port number for the primary Dynamic Data Masking Server.

   Rule Action

   Select the Folder rule action. The Folder rule action creates a rule folder.

   Processing Action

   Select the Stop if Matched processing action to process only the connection rules in the rule folder.
4. In the rule folder, create a connection rule that sets the priority levels of the Dynamic Data Masking Servers. Configure the following properties for the connection rule:

Matcher

Select the All Incoming Connections matcher. The All Incoming Connections matcher applies the rule action to all SQL requests.

Rule Action

Select the Load Control rule action. The Load Control rule action identifies the Dynamic Data Masking Servers and port numbers, and sets the server priority level. Configure the following Load Control properties:

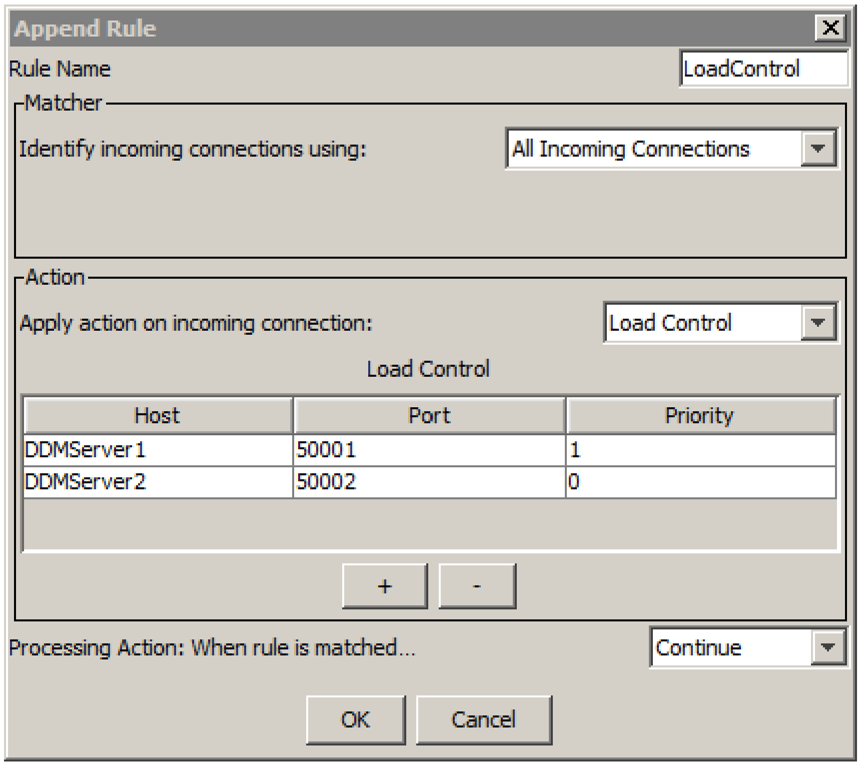
- **Host**. Enter the names of the Dynamic Data Masking Servers. Click the plus sign (+) to add additional servers.
- **Port**. Enter the port number for each of the Dynamic Data Masking Servers.
- **Priority**. Enter a priority number for each of the Dynamic Data Masking Servers. The value of the Priority property corresponds to the frequency that the client sends the request through the Dynamic Data Masking Server.

Note: For more information about the Load Control action and how to set priority levels, see the *Dynamic Data Masking User Guide*.

Processing Action

Select the Continue processing action. The Continue processing action sends the request to the next rule in the tree.

The following image shows an example of the connection rule:


Append Rule

Rule Name: LoadControl

Matcher

Identify incoming connections using: All Incoming Connections

Action

Apply action on incoming connection: Load Control

Load Control

Host	Port	Priority
DDMServer1	50001	1
DDMServer2	50002	0

Processing Action: When rule is matched... Continue

5. In the connection rule folder, create another connection rule that sends the request to the database. Configure the following properties for the connection rule:

Matcher

Select the All Incoming Connections Matcher.

Rule Action

Select the Switch to Database action. The Switch to Database action sends the request to the database that you specify.

Database

Enter the name of the database node in the Management Console tree.

Processing Action

Select the Continue processing action.

6. Save the connection rule set for the primary Dynamic Data Masking Server.
7. In the Management Console for the secondary Dynamic Data Masking Server, create a connection to the Db2 database. The database client must use the IBM JDBC driver to connect to the listener port that you define for the connection.

   Note: If the Dynamic Data Masking Server installations are on the same machine, you must configure different listener port numbers for the primary and secondary servers.
8. Add the Dynamic Data Masking service for DB2.
9. Open the connection rule set for the Dynamic Data Masking service and create connection rules that are identical to the connection rules that you created for the primary Dynamic Data Masking Server except that the Load Control action priority levels are switched.
10. Configure the following properties for the rule folder:

    Matcher

    Select the Incoming DDM Listener Port matcher.

    Incoming Port

    Enter the listener port number for the secondary Dynamic Data Masking Server.

    Rule Action

    Select the Folder rule action.

    Processing Action

    Select the Stop if Matched processing action to process only the connection rules in the rule folder.
11. Configure the following properties for the first rule in the rule folder:

    Matcher

    Select the All Incoming Connections matcher.

    Rule Action

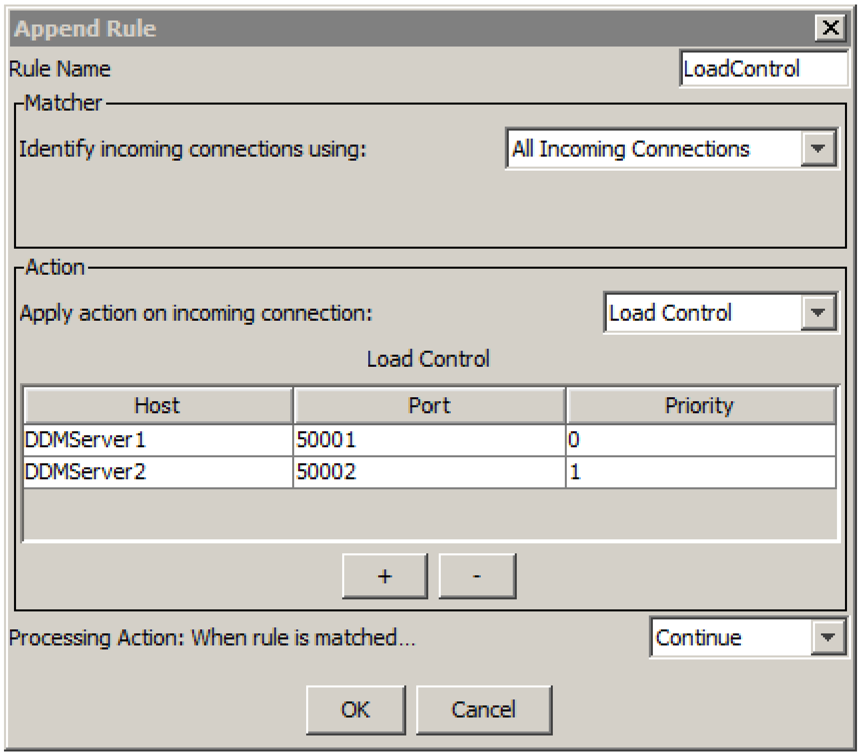
    Select the Load Control rule action. Configure the following Load Control properties:

    - **Host**. Enter the names of the Dynamic Data Masking Servers. Click the plus sign (+) to add additional servers.
    - **Port**. Enter the port number for each of the Dynamic Data Masking Servers.
    - **Priority**. Enter a priority number for each of the Dynamic Data Masking Servers. Enter one (1) for the secondary Dynamic Data Masking Server and zero (0) for the primary Dynamic Data Masking Server.

Processing Action

Select the Continue processing action.

The following image shows an example of the connection rule:



Rule Name: LoadControl

Matcher

Identify incoming connections using: All Incoming Connections

Action

Apply action on incoming connection: Load Control

Load Control

Host	Port	Priority
DDMServer1	50001	0
DDMServer2	50002	1

Processing Action: When rule is matched... Continue

12. Configure the following properties for the second rule in the rule folder:

    Matcher

    Select the All Incoming Connections Matcher.

    Rule Action

    Select the Switch to Database action.

    Database

    Enter the name of the database node in the Management Console tree.

    Processing Action

    Select the Continue processing action.

13. Save the connection rule set for the secondary Dynamic Data Masking Server.

You can connect to the Db2 database through either of the Dynamic Data Masking Servers. If one of the Dynamic Data Masking Servers is unavailable, the request goes through the other Dynamic Data Masking Server.

To verify which Dynamic Data Masking Server receives requests, you can run Dynamic Data Masking in debug mode and check the log files to see which server provides debug information. You can also define database nodes for different databases on each server and check which database Dynamic Data Masking sends the request to.

Teradata COP Discovery

You can configure a Teradata environment with more than one Teradata database node. Each Teradata Database node in the system that runs the Teradata Database Gateway process is called a communications processor, or COP. You can configure Dynamic Data Masking for a procedure called COP discovery, which identifies COP host names in the Teradata environment and checks their availability before making a connection to an available COP host. COP discovery is useful when you want to use all of the Teradata database nodes depending on their availability.

COP Discovery through the Teradata Driver

In a typical Teradata environment, the Teradata JDBC driver handles COP discovery. COP discovery through the Teradata driver identifies all of the COP host IP addresses.

Internally, when the Teradata JDBC Driver runs COP discovery, it appends "cop1" to the database hostname, then proceeds to cop2, cop3, cop4. The driver will continue running DNS lookups sequentially until it encounters an unknown COP hostname. Alternatively, if you have enabled the COPLAST connection parameter, the driver runs DNS lookups until it finds a COP hostname whose IP address matches the coplast hostname. For more information on the COPLAST connection parameter and the coplast hostname, see the Teradata connectivity documentation.

When a client connects to a Teradata database through a JDBC URL, the client provides the hostname without the COP suffix and the driver selects a COP entry to attempt to connect. If the first COP entry fails, the driver tries another.

COP Discovery in Dynamic Data Masking

Dynamic Data Masking always performs COP discovery. It is important to note that the COP discovery process in Dynamic Data Masking differs from the process used by the Teradata driver.

To perform COP discovery and connect to a Teradata database node, Dynamic Data Masking completes the following process:

1. Dynamic Data Masking performs COP discovery once for all Teradata databases, when the Dynamic Data Masking Server starts. This is in contrast to the Teradata driver, which performs COP discovery each time a client connects to the database. If you add or change a Teradata database, Dynamic Data Masking performs COP discovery again in that specific Dynamic Data Masking database. Dynamic Data Masking uses the same naming convention as the Teradata driver to discover COP hosts.
2. Dynamic Data Masking discovers COP host names, unlike the Teradata driver, which discovers COP IP addresses. During COP discovery Dynamic Data Masking generates a list of available COP hosts, which could be empty. Dynamic Data Masking looks up COP hosts in the network DNS.
3. Internally, Dynamic Data Masking uses one high availability thread to periodically check connections to all Teradata COP hosts and mark them as available or not available. The default time to periodically check connections to COP hosts is three minutes. You can specify this amount of time in the `jvm.params` file with the verify.connection.time parameter, which is configured in milliseconds.
4. When a client connects to a Teradata database through Dynamic Data Masking, Dynamic Data Masking randomly connects to a COP discovered host that the high availability thread marked as available. Before Dynamic Data Masking makes the connection, it checks that the COP host is still available by creating a socket to the COP host and port. The process of creating a socket to the host and port takes a few milliseconds, if the COP host is up and running. The process takes no longer than twice the amount of time specified in the verify.connection.time parameter for the high availability thread. You can also set

the maximum amount of time for this process by configuring the max.connection.time parameter in the `jvm.params` file. The default maximum time is 500 milliseconds.

5. If the COP host is available, Dynamic Data Masking makes the connection. If either the COP host or port is not available at that moment, Dynamic Data Masking picks the next available COP host from the list of COPs. Dynamic Data Masking again creates a socket to the COP host and port to check its availability.

To optimize performance, you can install Dynamic Data Masking on the COP hosts. This reduces the amount of time taken to check the host availability and make the connection.

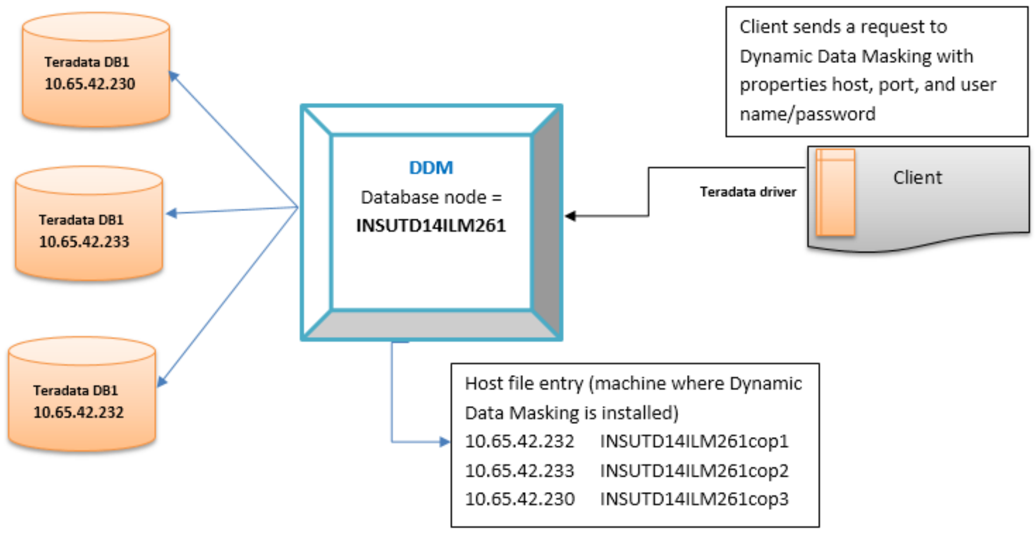
Dynamic Data Masking stores the names of COP hosts and not the IP addresses. If an IP address changes in the DNS table, Dynamic Data Masking resolves a known COP host to a new IP address. However, if the COP host is down, Dynamic Data Masking is unable to resolve the COP host to an IP address in the DNS. If no other COP hosts are available, the connection might fail.

You cannot disable COP discovery in Dynamic Data Masking. If COP discovery does not return any COP hosts, Dynamic Data masking connects directly to the host specified in the Teradata database form.

Implementation Scenarios

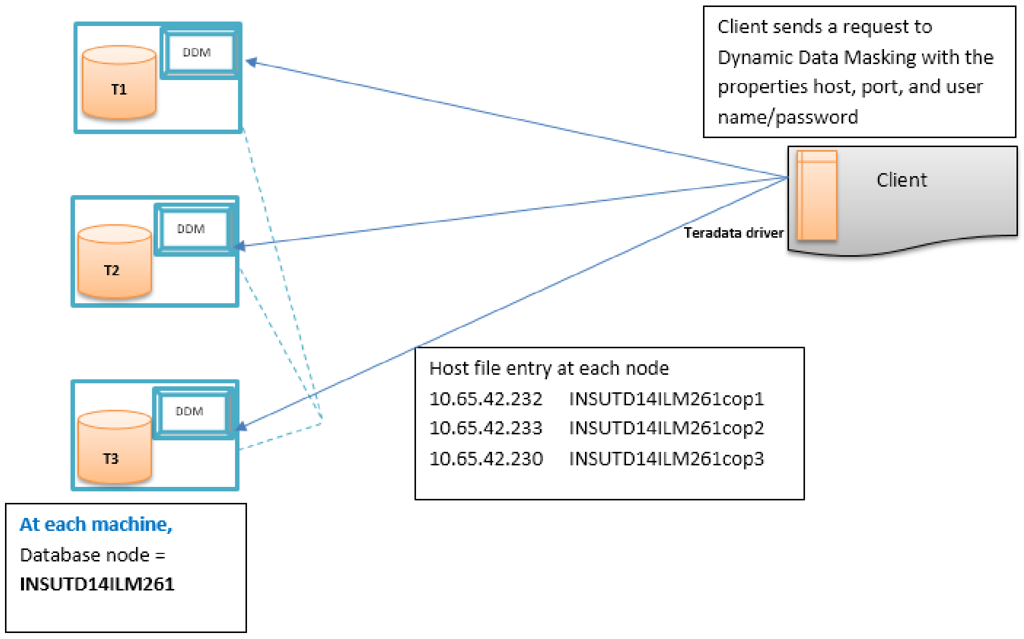
Dynamic Data Masking as a Single Point of Failure

The following diagram shows an example implementation where the Teradata instance details are configured on the same machine where Dynamic Data Masking is installed. In this scenario, the client sends a request to Dynamic Data Masking and Dynamic Data Masking creates the InetSocketAddress object.


Teradata DB1
10.65.42.230
Teradata DB1
10.65.42.233
Teradata DB1
10.65.42.232
DDM
Database node =
INSUTD14ILM261
Client sends a request to Dynamic Data Masking with properties host, port, and user name/password
Teradata driver
Client
Host file entry (machine where Dynamic Data Masking is installed)
10.65.42.232 INSUTD14ILM261cop1
10.65.42.233 INSUTD14ILM261cop2
10.65.42.230 INSUTD14ILM261cop3

Dynamic Data Masking and Teradata Installed Together

The following diagram shows an example of an implementation where the Teradata instance and Dynamic Data Masking are installed together at each node. The client sends a request to Dynamic Data Masking and Dynamic Data Masking creates the InetSocketAddress object.


Client sends a request to Dynamic Data Masking with the properties host, port, and user name/password
DDM
T1
DDM
T2
DDM
T3
Client
Teradata driver
Host file entry at each node
10.65.42.232 INSUTD14ILM261cop1
10.65.42.233 INSUTD14ILM261cop2
10.65.42.230 INSUTD14ILM261cop3
At each machine,
Database node =
INSUTD14ILM261

You can create the host configuration file, which contains details of the COP host names and corresponding IP addresses, either with or without the domain name. For example:

```
10.65.42.233 INSUTD14ILM261cop2
```

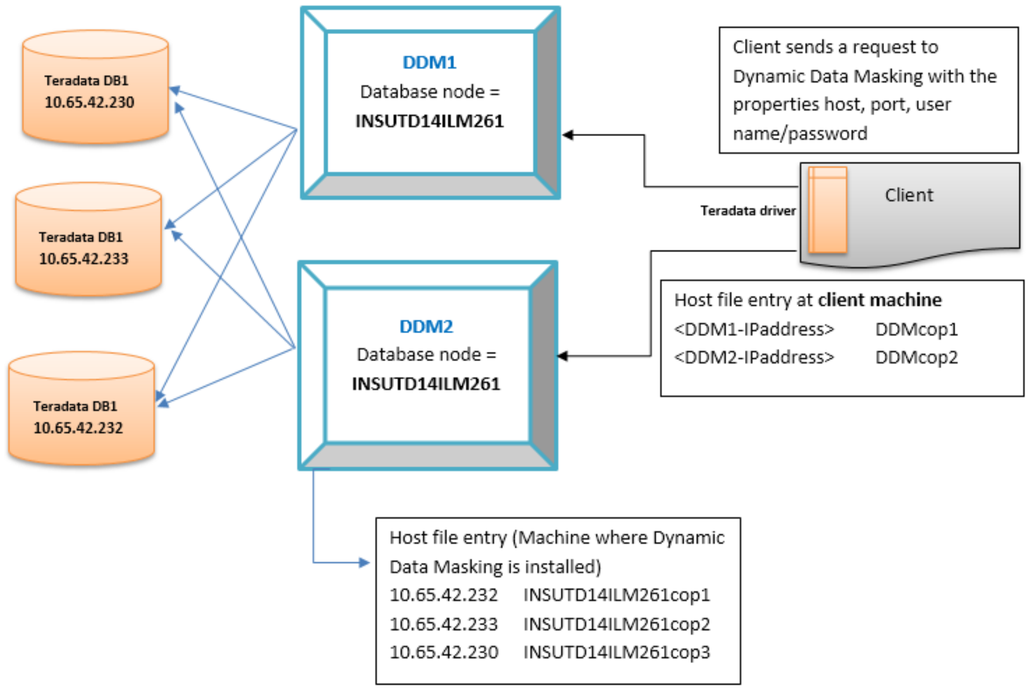
and

```
10.65.42.233 INSUTD14ILM261cop2.informatica.com
```

are equivalent entries in the host configuration file.

Multiple Dynamic Data Masking Nodes

The following diagram shows an example of an implementation with multiple installations of Dynamic Data Masking to handle failover. Each Dynamic Data Masking instance contains details of multiple Teradata instances.


Teradata DB1
10.65.42.230
Teradata DB1
10.65.42.233
Teradata DB1
10.65.42.232
DDM1
Database node =
INSUTD14ILM261
DDM2
Database node =
INSUTD14ILM261
Client sends a request to Dynamic Data Masking with the properties host, port, user name/password
Client
Teradata driver
Host file entry at client machine
<DDM1-IPaddress> DDMcop1
<DDM2-IPaddress> DDMcop2
Host file entry (Machine where Dynamic Data Masking is installed)
10.65.42.232 INSUTD14ILM261cop1
10.65.42.233 INSUTD14ILM261cop2
10.65.42.230 INSUTD14ILM261cop3

Hive Connectivity with Apache ZooKeeper and Dynamic Data Masking

Apache ZooKeeper is a reliable, highly available, persistent, and distributed coordination service that provides a centralized infrastructure and services that enable synchronization across a cluster. You can configure Apache ZooKeeper in combination with a Hive database driver to load balance database connections.

To achieve high availability for the Dynamic Data Masking servers and utilities in a distributed environment, configure Dynamic Data Masking as a node in ZooKeeper. When a server or service is down, ZooKeeper continues to provide uninterrupted service by switching queries to a server, or service, that is running.

Configuring Dynamic Data Masking for Apache ZooKeeper

To group the Dynamic Data Masking servers in a cluster in ZooKeeper, create a parent node. The parent node is a persistent node and will continue to exist after you restart ZooKeeper.

1. Use the Dynamic Data Masking Server control commands to create the parent node in ZooKeeper.
   a. Start the Dynamic Data Masking Server.
   b. Connect to ZooKeeper with the following command:

```
server zookeeper connect -url '<zookeeper_ensemble>'
```

Example:

```
server zookeeper connect -url 'server1.informatica.com:2181,
server2.informatica.com:2181, server3.informatica.com:2181'
```

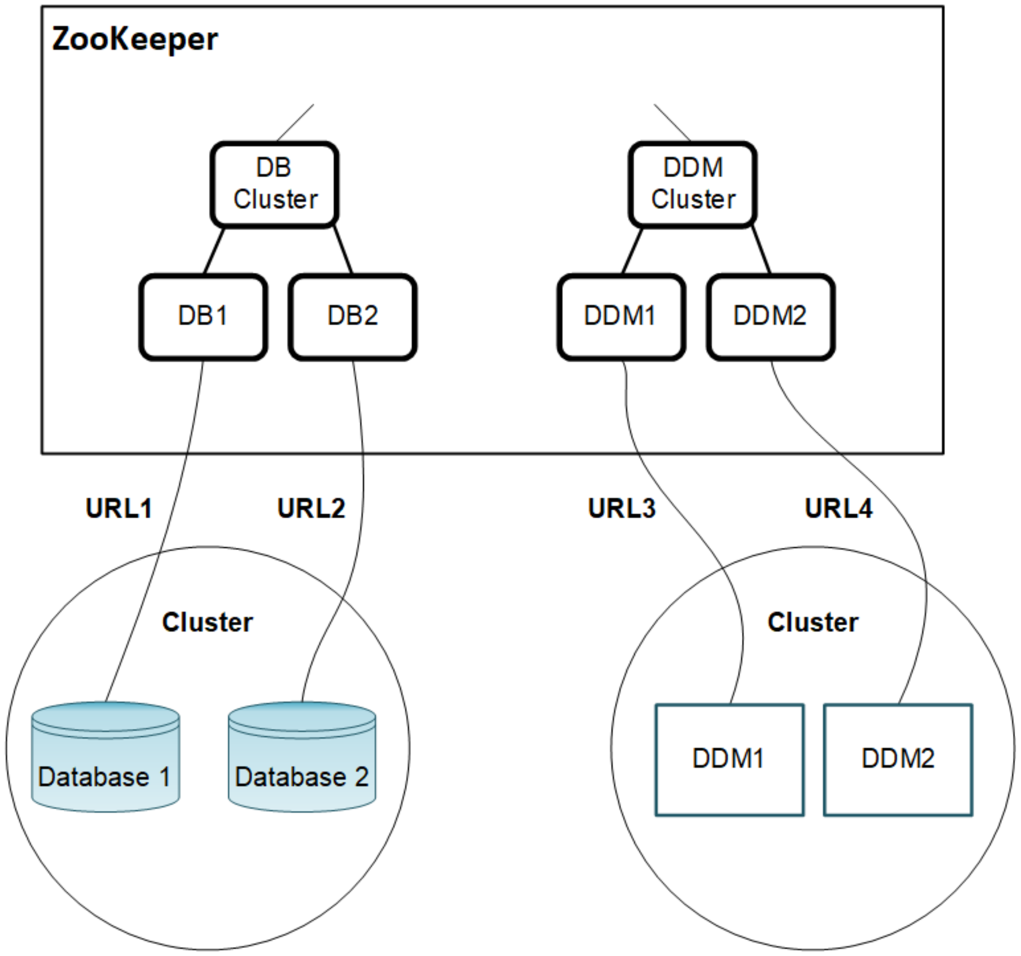
c. Create a parent node in ZooKeeper with the following command:

```
server zookeeper create -path /<name of parent node>
```

Example:

```
server zookeeper create -path /DDM
```

The following image shows a database cluster with two database nodes and a Dynamic Data Masking cluster with two server nodes:



2. Create an ephemeral node for each Dynamic Data Masking Server in the cluster. Initialize the ephemeral node with the URL copied from the database node of *hiveserver2*. Replace the host and port values of the database URL with the corresponding host and port values of the Dynamic Data Masking Server.

   If the environment is enabled for Kerberos, replace the Hive service principal with Dynamic Data Masking service principal defined in the `/cfg/ddm.security` file.

   ```
   server zookeeper create -path /<DDM cluster path> -type EPHEMERAL -data
   <driver_url_toddm_cluster>
   ```

   Example:

   ```
   server zookeeper create -type EPHEMERAL -path /DDM/ddm1 -data
   "hive.server2.instance.uri=<ddmhost1>:<DDM hive service
   port>;hive.server2.authentication=KERBEROS;hive.server2.transport.mode=binary;hive.se
   rver2.thrift.sasl.qop=auth;hive.server2.thrift.bind.host=<ddmhost1>;hive.server2.thri
   ```

```
ft.port=<DDM hive service
port>;hive.server2.use.SSL=false;hive.server2.authentication.kerberos.principal=<Dyna
mic Data Masking service principal for ddmhost1>"
```

```
server zookeeper create -type EPHEMERAL -path /DDM/ddm2 -data
"hive.server2.instance.uri=<ddmhost2>:<DDM hive service
port>;hive.server2.authentication=KERBEROS;hive.server2.transport.mode=binary;hive.se
rver2.thrift.sasl.qop=auth;hive.server2.thrift.bind.host=<ddmhost2>;hive.server2.thri
ft.port=<DDM hive service
port>;hive.server2.use.SSL=false;hive.server2.authentication.kerberos.principal=<Dyna
mic Data Masking service principal for ddmhost2>"
```

Note: A persistent node can have child nodes. An ephemeral node cannot have child nodes.

Configuring the Hive Database for ZooKeeper on the Dynamic Data Masking Server

Configure a connection to the Hive database in Dynamic Data Masking and test the connection.

1. From the Management Console, click **Tree** > **Add Database**.

   The **Add Database** window appears.
2. Select **Hive** as the database type.
3. In the **DDM Database Name** field, enter the name of the Dynamic Data Masking database.

   Example: `hive2_serv`
4. In the **Driver Classpath** field, enter the absolute path of each driver.

   Example: `C:\hive_jdbc_driver\*"`
   Verify that the folder for each driver contains the required JAR files.

   For example, the Hortonworks distribution Hive driver must contain the following files:
   - `hadoop-auth-xxx.jar`
   - `hadoop-common-xxx.jar`
   - `hive-jdbc-xxx-standalone.jar`
   - `zookeeper-xxx.jar`
   - `commons-collections-xxx.jar`
   - `commons-configurationxxx.jar`
   - `commons-lang-xxx.jar`
   - `guava-xxx.jar`
   - `slf4j-api-xxx.jar`
   - `slf4j-log4jxxx.jar`
5. In the **Driver Class Name** field, enter the fully qualified class name of the Hive driver.

   Example: `org.apache.hive.jdbc.HiveDriver`
6. In the **Connection String (URL)** field, enter the JDBC connection URL based on the driver.

   If the Hive database is Kerberos-enabled, the URL must include the **auth** and **kerberosAuthType** properties.

   ```
   jdbc:hive2://<zookeeper_ensemble>/
   <databaseName>;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=<hive_cluster>;auth=
   kerberos;kerberosAuthType=fromSubject
   ```

   Example: `jdbc:hive2://server1.informatica.com:2181,server2.informatica.com:2181,server3.informatica.com:2181/`

`hiveDatabase;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=hiveserver2;auth=kerberos;kerberosAuthType=fromSubject`

Note: If you do not provide the *databaseName*, Dynamic Data Masking connects to the default Hive database.

For Kerberos-enabled Hive databases, replace the *_HOST* value of the property `hive.server2.authentication.kerberos.principal=hive/_HOST@KERB` with the Hive server hostname in the Hive configuration.

After you replace the *_HOST* value, the property value appears as: `_HOST`
`hive.server2.authentication.kerberos.principal=hive/your.host.com@KERB`

7. In the **Database Server Principal** field, specify the server principal of Hive.

   Example: `hive/hiverserver@REALM.COM`

   Do not enter a value in the **DBA Username** and **DBA Password** fields.

8. Click **Test Connection**.

   A JDBC connection to the Dynamic Data Masking service opens. The services uses the defined database connection parameters. A confirmation message appears if the connection is valid.

Configuring the JDBC Client

To configure the JDBC client, specify the direct URL and the Dynamic Data Masking URL.

1. Specify the direct URL:

   ```
   jdbc:hive2://<zookeeper_ensemble>/
   <datbaseName>;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=<hive_cluster>;
   ```

2. Specify the Dynamic Data Masking URL:

   ```
   jdbc:hive2://<zookeeper_ensemble>/
   <databaseName>;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=<ddm_cluster>;
   ```

   Replace `<ddm_cluster>` with the name of the Dynamic Data Masking cluster in ZooKeeper.

Connecting the Dynamic Data Masking Server to ZooKeeper on Server Startup

Create a script file that connects the Dynamic Data Masking Server to ZooKeeper when the Dynamic Data Masking Server starts.

1. Create a parent node in ZooKeeper.

   Example: `server zookeeper create -path /DDM`
   The parent node contains the Dynamic Data Masking clusters in ZooKeeper. The parent node is a permanent node that continues to exist even after ZooKeeper is restarted.

2. Create a regular parent node for each database cluster if none exists.

   ```
   server zookeeper create -path /DDM
   ```

3. To automatically connect the Dynamic Data Masking Server to Zookeeper and create ephemeral nodes, create a script file with the name `serverStarting.bat` or `serverStarting.sh`.

   Otherwise, you can manually run the ZooKeeper commands after you start the server.

4. Provide the following information in the `serverStarting` script file:

   a. The `connect` command and parameters to connect the Dynamic Data Masking Server and ZooKeeper.

b. The `create` command and parameters to create an ephemeral node for each database server.

c. Initialize the ephemeral nodes with specific URLs. The URL must include the host and port of the Dynamic Data Masking Server.

   The database driver uses the URL to connect to the database cluster through Dynamic Data Masking.

Linux Example:

```
#!/bin/sh
cd `dirname $0`
. ./server zookeeper connect -url 'zookeeper_ensemble'
. ./server zookeeper create -type EPHEMERAL -path /<DDM cluster>/<name of ephemeral
node> -data
hive.server2.instance.uri=<ddmhost1>:10001;hive.server2.authentication=KERBEROS;hive.
server2.transport.mode=binary;hive.server2.thrift.sasl.qop=auth;hive.server2.thrift.b
ind.host=<ddmhost1>;hive.server2.thrift.port=<DDM Hive service
port>;hive.server2.use.SSL=false;hive.server2.authentication.kerberos.principal=<hive
 service principal for ddmhost1>
```

Microsoft Windows Example:

```
@echo off
REM *********************************************
REM Start DDM Server
REM *********************************************
CALL server start

REM *********************************************
REM Connect DDM Server to Zookeeper
REM *********************************************
CALL server zookeeper connect -url '<zookeeper_ensemble>'



REM *********************************************
REM Create a new ephemeral node for the DDM Server
REM *********************************************
CALL server zookeeper create -type EPHEMERAL -path /<DDM cluster>/<name of ephemeral
node> -data "hive.server2.instance.uri=<ddmhost1>:<DDM Hive service
port>;hive.server2.authentication=KERBEROS;hive.server2.transport.mode=binary;hive.se
rver2.thrift.sasl.qop=auth;hive.server2.thrift.bind.host=<ddmhost1>;hive.server2.thri
ft.port=<DDM Hive service
port>;hive.server2.use.SSL=false;hive.server2.authentication.kerberos.principal=<hive
 service principal for ddmhost1>"
```

5. Save the `serverStarting` script file in the Dynamic Data Masking installation directory.

   When the Dynamic Data Masking Server starts, the server runs the `serverStarting` script file.

6. Repeat steps 3-5 for other Dynamic Data Masking servers in the cluster.

Dynamic Data Masking Server Commands for ZooKeeper

Use the Dynamic Data Masking Server commands to connect to and manage ZooKeeper.

Use the `connect` command to connect to the ZooKeeper server.

After you connect to the ZooKeeper server, you can manage Dynamic Data Masking servers and services on ZooKeeper with the following commands:

- `create`
- `delete`
- `disconnect`
- `get`

- `list`
- `update`

Some commands must include parameters such as data, the path to the node, the node type, and the URL to ZooKeeper. Use the following table to determine which parameters (-data, -path, -type, -url) are required and optional for each command:

Command	-data	-path	-type	-url	Description
connect	-	Optional	-	Required	Connects to ZooKeeper using the URL. If you specify the path, ZooKeeper treats it as the root directory. Nodes above this root will not be available for this connection.
create	Optional	Required	Optional	-	Creates a new node. If you do not specify the data parameter, the default value is null. If you do not specify type, the default value is PERSISTENT. You can specify one of the following types: - EPHEMERAL - EPHEMERAL_SEQUENTIAL - PERSISTENT - PERSISTENT_SEQUENTIAL
delete	-	Required	-	-	Deletes the parent node and children nodes recursively.
disconnect	-	-	-	-	Disconnects from ZooKeeper and deletes the ephemeral nodes.
get	-	Required	Optional	-	Prints data from the specified path. If you do not specify type, the default value is String. You can specify one of the following types: - string - short - int - integer - float - long - double - byte - boolean (true | false) - date (a long number)
list	-	Optional	-	-	Lists all nodes recursively. If you do not specify path, the default value is `/`.
update	Optional	Required	-	-	Updates the existing node in the specified path.

Use the following syntax:

```
server zookeeper [create | update | delete | get | list | connect | disconnect] -path
<path> -data <data> -type <type> -url <url>
```

If you want to create data nodes for a remote Dynamic Data Masking Server, you can connect to ZooKeeper from your local Dynamic Data Masking Server with the following commands:

```
server zookeeper connect -url <url of the zookeeper ensemble>  -targets user/
password@host:port[;SSL];
```

```
server zookeeper create -path <path> -targets user/password@host:port[;SSL]
```

Restoration of Broken Connections to ZooKeeper

If the connection to ZooKeeper is lost, the Dynamic Data Masking Server automatically attempts to restore the connection and re-create the ephemeral node in ZooKeeper.

The following process describes how the Dynamic Data Masking Server attempts to restore the connection and re-create the ephemeral node:

1. During the connection, Dynamic Data Masking provides the session timeout value to ZooKeeper. ZooKeeper manages session expiration.
2. If the connection is lost, Dynamic Data Masking attempts to restore the broken connection within the specified timeout period in order to retain the ephemeral nodes.
3. When ZooKeeper does not receive a response from Dynamic Data Masking within the specified session timeout period, it expires the Dynamic Data Masking session. When the session expires, ZooKeeper removes all ephemeral nodes owned by the session and immediately notifies all connected clients, which are watching the nodes, about the change.
4. The Dynamic Data Masking instance associated with the expired session remains disconnected from ZooKeeper, and is not notified of the session expiration until it reconnects with ZooKeeper. When the connection with ZooKeeper is reestablished, Dynamic Data Masking receives a Session Expired notification. At this point, Dynamic Data Masking attempts to re-create all ephemeral nodes that were created during the expired session.

By default, Dynamic Data Masking provides a session timeout value of 10,000 milliseconds (10 seconds) to ZooKeeper. To modify the default value, set the **zookeeper.timeout** system property in the `jvm.params` file. For example, to set a session timeout value of 12 seconds specify:

```
zookeeper.timeout=12000
```

For the minimum and maximum session timeout values that ZooKeeper allows the client to negotiate, see the ZooKeeper documentation. For more information, contact your ZooKeeper administrator.

Dynamic Data Masking monitors the ephemeral nodes that it creates so it can restore the nodes and node content as follows:

- When reestablishing connection with Zookeeper, Dynamic Data Masking restores the ephemeral nodes and the content.
- If a third-party tool is used to remove the ephemeral nodes, or remove the parent cluster node and then create it again, Dynamic Data Masking restores the ephemeral nodes.
- If a third-party tool is used to modify the ephemeral nodes, Dynamic Data Masking restores the node content.

CHAPTER 11

Server Control

This chapter includes the following topics:

Server Control Overview

The Dynamic Data Masking Server Control program is a command line interface that you use to manage Dynamic Data Masking Servers and services. Server Control reads administrative requests and writes the results of the requests to the standard system output and error streams.

Server Control has a set of commands that simplify management and configuration of local and remote Dynamic Data Masking Servers. Server Control is installed with the Dynamic Data Masking Server. Run Server Control on the machine where you installed the Dynamic Data Masking Server.

You can run the following types of commands from Server Control:

server

Use `server` commands to configure the local Dynamic Data Masking Server and Dynamic Data Masking services. For example, you can start and stop the Dynamic Data Masking Server and services, set the Dynamic Data Masking listener port, and view the Dynamic Data Masking Server version.

You can run the equivalents of the following Server Control server commands from shell scripts:

- start
- stop
- restart
- startDDMService
- stopDDMService
- restartDDMService
- status

server config

Use `server config` commands to perform configuration tasks on local and remote Dynamic Data Masking Servers. For example, you can set Dynamic Data Masking database passwords, synchronize Dynamic Data Masking Server configurations, and export and import security rule sets and Dynamic Data Masking databases.

server service

Use `server service` commands to manage Dynamic Data Masking services. For example, you can import and export Dynamic Data Masking services.

Important: Server Control stores information in the config.properties file. You must not modify the config.properties file.

Running Server Control

Run Server Control to manage the Dynamic Data Masking Server from a command line interface.

If you run the Dynamic Data Masking Server on Windows, Server Control runs as a batch file with a .bat extension.

If you run the Dynamic Data Masking Server on Linux, Server Control runs as a shell script with no extension. You can run the `server` shell script to use the Server Control commands. Alternatively, you can run a subset of the Server Control commands from individual shell scripts. For example, to start the Dynamic Data Masking Server, you can run the `server` shell script with the start command or you can run the `start` shell script directly.

Running Server Control on Windows

On Windows, run the Server Control command line program from the Start Menu.

1. Select **Start** > **Informatica** > **Dynamic Data Masking** > **Server Control**.
2. At the command line, enter commands with the following syntax:

   ```
   server <command name> <parameter>
   ```

For example, the following command sets the port for the Dynamic Data Masking Server to 8195:

```
server setPort 8195
```

Running Server Control on Linux or UNIX

On Linux, use the `server` shell script to run Server Control commands. Alternatively, run Server Control commands from individual shell scripts.

1. Open a terminal, and navigate to the Dynamic Data Masking Server installation directory.

   For example, you might navigate to the following directory:

   ```
   /home/Informatica/DDM
   ```
2. Run the shell script for the Server Control command that you want to use.
   - If you run the `server` shell script, enter commands with the following syntax:

     ```
     ./server <command name> <parameter>
     ```
   - If you run a shell script for a specific command, run the shell script with the following syntax:

     ```
     ./<shell script name> <parameter>
     ```

For example, the following command uses the `server` shell script to start the DDM for Oracle service:

```
./server startDDMService "DDM for Oracle"
```

Alternatively, the following command uses the `startDDMService` shell script to start the DDM for Oracle service:

```
./startDDMService "DDM for Oracle"
```

Server Control Commands

Enter commands and parameters in the Server Control command line program to manage the Dynamic Data Masking Servers and services.

Use the following rules when you enter commands and parameters:

- The first word after a command is the parameter.
- If a parameter contains spaces, enclose the parameter in double quotes.
- Server Control commands are not case sensitive.

Syntax Notation

Before you use Server Control, review the syntax notation.

The following table describes the Server Control syntax notation:

Convention	Description
<x>	Required parameter. If you omit a required parameter, Server Control returns an error message.
[x]	Optional parameter. The command runs whether or not you enter optional parameters.
-x	Parameter placed before an argument that designates the argument that you enter. For example, to enter a target Dynamic Data Masking Server, type `-targets` followed by the Dynamic Data Masking Server name.

Server Commands

Use Server Control `server` commands to configure the local Dynamic Data Masking Server and services.

Server Control has the following `server` commands:

- audit
- checkPort
- displayAddress
- encrypt

- help
- log
- remove
- removeAddress
- rename
- restart
- restartDDMService
- services
- setAddress
- setInternalPassword
- setPort
- start
- startDDMService
- status
- stop
- stopDDMService
- support
- version

Audit

Generates audit trail reports to help you analyze specific data. The audit trail report shows all changes made by users for selected objects in the specified time frame. The command also verifies that audit trail entries were not altered. You can write the report to the console or to a CSV file.

Note: Use the audit command only with files generated by Dynamic Data Masking 9.9.1 or later.

The command uses the following syntax:

```
server audit  [-report <compact | standard>] [-file <file0 file1…> | [-start <yyyy_MM>]
[-end <yyyy_MM>]] [-out <console | file.csv>]
```

Example 1:

```
server audit -report standard -file 2019_03.at  -out  D:\Audit\AuditReport.csv
```

Example 2:

```
server audit -report compact  -start 2019_03 -end 2019_05 -out console
```

The `audit` command uses the following parameters:

report

Generate an audit trail report. You must specify a report type. Choose from a standard or a compact report. Default is compact.

- **standard**. Contains the Date, Address, User name, Type, Path, Object name, Operation, Result, Attribute, Old value, New value, and Notes.
- **compact**. Contains the Date, Address, User name, Type, Path, Object name, Operation, Result, and Notes.

file

Read the specified audit log file or files. Insert a space to separate file names.

start

Date when you want Dynamic Data Masking to start reading audit log files. Dynamic Data Masking reads all audit log files between the start and end dates you specify. If you do not specify a start date, Dynamic Data Masking uses the audit log file for the current date.

end

Date when you want Dynamic Data Masking to stop reading audit log files. Dynamic Data Masking reads all audit log files between the start and end dates you specify. If you do not specify an end date, Dynamic Data Masking uses the audit log file for the current date.

out

Specify how to write the audit report. To write the report to the console, specify console. To write the report to a file, specify a path to a CSV file. Default is console.

For sample reports, see “Audit Trail Reports” on page 103.

CheckPort

Checks if a port is available or locked. Use the command to identify ports that you can use for the Dynamic Data Masking Server.

The `checkPort` command uses the `port` parameter. The value of the `port` parameter is the port number that you want to check for the Dynamic Data Masking Server. The command uses the following syntax:

```
server
  checkPort <port>
```

For example, you might enter the following command:

```
server checkPort 6002
```

DisplayAddress

The `displayAddress` command returns a message indicating whether the Dynamic Data Masking Server network address has been set with the `setAddress` command. If the address has been set, it displays the address.

The command uses the following syntax:

```
server
  displayAddress
```

Encrypt

Encrypts a file. Informatica Global Customer Support can decrypt files that you encrypt with the `encrypt` command.

You can send an encrypted screenshot or log file that contains sensitive information to Informatica Global Customer Support. For example, if you saved an unencrypted log archive with the `support` command, you can use the `encrypt` command to encrypt the file before you send it to Informatica Global Customer Support. Notify Informatica Global Customer Support that you encrypted the file with the Server Control `encrypt` command.

The command uses the following syntax:

```
server
  encrypt [/y] <filename.zip> <filename-enrypted.zip>
```

For example, you might enter the following command:

```
server encrypt SUPPORT.zip SUPPORT_ENCRYPT.zip
```

The `encrypt` command uses the following parameters:

/y

Forces the command to continue without user confirmation. If you use the `/y` parameter, Server Control does not require confirmation to overwrite an existing file.

filename

The filename of the unencrypted file that you want to encrypt.

filename-encrypted

The filename of the encrypted file. If you have a case open with Informatica Global Customer Support, Informatica recommends that you include the case number in the name of the file.

Help

Displays descriptions and parameters for each Server Control command.

The command uses the following syntax:

```
server
  help
```

Log

Sets and displays the Dynamic Data Masking Server log level.

If you do not include a `Log_Level` parameter in the command, Server Control displays the current log level. You can set the following log levels:

- INFO
- DEBUG
- WARN
- ERROR

The command uses the following syntax:

```
server
  log <Log_Level>
```

For example, you might enter the following command:

```
server log warn
```

Remove

Removes the service or daemon for the Dynamic Data Masking Server. The Dynamic Data Masking Server must be stopped to run the command.

The command uses the following syntax:

```
server
  remove
```

RemoveAddress

Removes the Dynamic Data Masking Server network address that you added with the `setAddress` Server Control command. The Dynamic Data Masking Server must be shut down to run the command.

The command uses the following syntax:

```
server
  removeAddress
```

Rename

Renames the Dynamic Data Masking service on Windows or the daemon on Linux. The Dynamic Data Masking Server must be shut down to run the command.

The `rename` command uses the `name` parameter. The value of the `name` parameter is the name that you want to set for the Dynamic Data Masking Server. The command uses the following syntax:

```
server
  rename <name>
```

For example, you might enter the following command:

```
server rename Informatica_DDM
```

On Linux and UNIX, Dynamic Data Masking does not check to verify that the Dynamic Data Masking Server name you choose is not in use. Before you change the Server name, make sure that there is not a Dynamic Data Masking Server in the same environment with the same name.

Restart

Restarts the Dynamic Data Masking Server.

The command uses the following syntax:

```
server
  restart
```

On Linux, you can also run the `restart` shell script to restart the Dynamic Data Masking Server.

RestartDDMService

Restarts a Dynamic Data Masking service configured in the Management Console. The Dynamic Data Masking Server must be running.

The `restartDDMService` command uses the `DDMService_Name` parameter. The value of the `DDMService_Name` parameter is the name of the Dynamic Data Masking service that you want to restart.

The command uses the following syntax:

```
server
  restartDDMService <DDMService_Name>
```

For example, you might enter the following command:

```
server restartDDMService "DDM for SQL Server"
```

On Linux, you can also run the `restartDDMService` shell script to restart a Dynamic Data Masking service.

Services

Lists Dynamic Data Masking services in the Management Console Tree. The Dynamic Data Masking Server must be running.

The command uses the following syntax:

```
server
  services
```

SetAddress

Sets the network address of the Dynamic Data Masking Server to enable IP address binding. The Dynamic Data Masking Server must be shut down to run the command.

Use the `setAddress` command if you want multiple Dynamic Data Masking Servers to run on the same machine and use the same listener port or if you want to use the same listener port for multiple Dynamic Data Masking services when the same service is defined on different Dynamic Data Masking Servers. The command creates a `server.address=<IP address>` property in the `config.properties` file and allows Dynamic Data Masking to identify the server or service by `<IP address>:<port>` or `<hostname>/<IP address>:<port>`.

You can use the `displayAddres` command to view the current network address and the `removeAddress` command to remove the address.

The command uses the following syntax:

```
server
  setAddress <IP address>
```

For example, you might enter the following command:

```
server setAddress 123.45.678.9
```

SetInternalPassword

Sets a value for the Dynamic Data Masking Server internal password. The Dynamic Data Masking Server must be shut down to execute the command.

The `setInternalPassword` command uses the `password` parameter. The value for the `password` parameter is the internal password that you want to set for the Dynamic Data Masking Server.

The command uses the following syntax:

```
server
  setInternalPassword <password>
```

For example, you might enter the following command:

```
server setInternalPassword SpF_dPHx
```

SetPort

Sets the value of the server management port. The Dynamic Data Masking Server must be shut down to run the command.

The `setPort` command uses the `port` parameter. The value of the `port` parameter is the port number that you want to set for the Dynamic Data Masking Server.

The command uses the following syntax:

```
server
  setPort <port>
```

For example, you might enter the following command:

```
server setPort 6002
```

Start

Starts the Dynamic Data Masking Server. Creates an operating system service in Windows if the service does not exist. The Dynamic Data Masking Server must be shut down to run the command.

The command uses the following syntax:

```
server
  start
```

On Linux, you can also run the `start` shell script to start the Dynamic Data Masking Server.

StartDDMService

Starts a Dynamic Data Masking service configured on the Dynamic Data Masking Server. The Dynamic Data Masking Server must be running.

The `startDDMService` command uses the `DDMService_Name` parameter. The value of the `DDMService_Name` parameter is the name of the Dynamic Data Masking service that you want to start.

The command uses the following syntax:

```
server
  startDDMService <DDMService_Name>
```

For example, you might enter the following command:

```
server startDDMService "DDM for Oracle"
```

On Linux, you can also run the `startDDMService` shell script to start a Dynamic Data Masking service.

Status

Shows whether the Dynamic Data Masking Server is running. The command also displays the name of the Dynamic Data Masking Server and the number of the server management port.

The command uses the following syntax:

```
server
  status
```

On Linux, you can also run the `isStarted` shell script to show the status of the Dynamic Data Masking Server.

Stop

Stops the Dynamic Data Masking Server.

The command uses the following syntax:

```
server
  stop
```

On Linux, you can also run the `stop` shell script to stop the Dynamic Data Masking Server.

StopDDMService

Stops a Dynamic Data Masking service configured on the Dynamic Data Masking Server. The Dynamic Data Masking Server must be running.

The `stopDDMService` command uses the `DDMService_Name` parameter. The value of the `DDMService_Name` parameter is the name of the Dynamic Data Masking service that you want to stop.

The command uses the following syntax:

```
server
  stopDDMService <DDMService_Name>
```

For example, you might enter the following command:

```
server stopDDMService "DDM for DB2"
```

On Linux, you can also run the `stopDDMService` shell script to stop a Dynamic Data Masking service.

Support

Creates a .zip archive of Dynamic Data Masking logs. Send the encrypted log archive to Informatica Global Customer Support to troubleshoot issues with Dynamic Data Masking.

The `support` command includes the following files in the archive:

- All files in the `<Dynamic Data Masking installation>/cfg` directory with the exception of the following files:
  - `config.cfg`. The archive contains `config.cfg.xml`, which is `config.cfg` in XML format. The XML file does not contain user passwords.
  - `config.pbk`
- All files in the `<Dynamic Data Masking installation>/lib` directory.
- All files in the `<Dynamic Data Masking installation>/log` directory.
- jvm.params

The following files are generated at runtime and included in the archive:

- DDM for <database>.txt. Contains statistics for running the Dynamic Data Masking services.
- `environment.txt`. Contains operating system environment parameters.
- `java_properties.txt`. Contains Java system properties.

Note: The `support` command does not save user passwords. Dynamic Data Masking replaces encrypted user passwords with new encrypted passwords.

The command uses the following syntax:

```
server
  support [/y] [-noencrypt] <filename.zip> [-source user/password@host:port]
```

For example, you might enter the following command:

```
server support SUPPORT.zip
```

The `support` command uses the following parameters:

/y

Forces the command to continue without user confirmation. If you use the `/y` parameter, Server Control does not require confirmation to overwrite an existing file.

-noencrypt

Saves the log archive unencrypted.

filename

The name of the log archive file. If you have a case open with Informatica Global Customer Support, Informatica recommends that you include the case number in the name of the file.

-source

Indicates that the following argument is a source. You can indicate a source to create a log archive of a remote Dynamic Data Masking Server.

user

The Dynamic Data Masking user name that you use to log in to the Dynamic Data Masking Server.

password

The password for the Dynamic Data Masking user.

host

The Dynamic Data Masking Server host name or IP address.

port

The Dynamic Data Masking Server listener port.

Note: You can encrypt an unencrypted log archive with the `encrypt` command.

Version

Displays the name and version of the Dynamic Data Masking Server. The Dynamic Data Masking Server must be running.

The command uses the following syntax:

```
server
  version
```

Server Config Commands

Use Server Control `server config` commands to perform configuration tasks on local and remote Dynamic Data Masking Servers.

Server Control has the following `server config` commands:

- export
- import
- setDBPassword
- sync
- setKeyStore

Export

Exports a Dynamic Data Masking database or security rule set. The Dynamic Data Masking Server must be running.

Use the `export` command to export a Dynamic Data Masking database or security rule set from the Dynamic Data Masking Server. You can then import the file into one or more Dynamic Data Masking Servers.

If you do not specify a source Dynamic Data Masking Server, the `export` command exports the object from the local Dynamic Data Masking Server.

The command uses the following syntax:

```
server config
  export [/y] <object full path> <file>
  [[-source] user/pwd@host:port]
```

For example, if you want to export a database named DEV_SYBASE_DB from the local Dynamic Data Masking Server and name the file DEV_SYBASE_DB_1, you might enter the following command:

```
server config export "Site\DEV_SYBASE_DB" "DEV_SYBASE_DB_1"
```

If you want to export the DEV_SYBASE_DB_1 database from a remote Dynamic Data Masking Server, you might enter the following command:

```
server config export "Site\DEV_SYBASE_DB" "DEV_SYBASE_DB_1" -source admin/
admin@ABC12345:7000
```

The `export` command uses the following parameters:

/y

Forces the command to continue without user confirmation. If you use the `/y` parameter, Server Control does not require confirmation to overwrite an existing file. For example, you might use the `/y` parameter in a script.

object full path

The path in the Management Console of the object that you want to export.

file

The name of the file that the object exports to.

-source

Indicates that the following argument is a source.

user

The Dynamic Data Masking user name that you use to log in to the Dynamic Data Masking Server.

pwd

The password for the Dynamic Data Masking user.

host

The Dynamic Data Masking Server host name or IP address.

port

The Dynamic Data Masking Server listener port.

Import

Imports a Dynamic Data Masking database or security rule set. The Dynamic Data Masking Server must be running.

Use the `import` command to import a Dynamic Data Masking database or security rule set into one or more Dynamic Data Masking Servers. The type of object that you import must be the same type of object that you specify as the location to import the object. For example, you can import a security rule set into a security rule set. You can import an Oracle database into an Oracle database node, but you cannot import an Oracle database into a Sybase database node.

If you do not specify a target Dynamic Data Masking Server, the `import` command imports the object into the local Dynamic Data Masking Server. Use the `-targets` parameter to specify multiple target Dynamic Data Masking Servers.

The command uses the following syntax:

```
server config
  import <object full path> <file>
  [[-targets] user/pwd@host:port [user/pwd@host:port user/pwd@host:port]]
```

For example, if you exported a database named DEV_SYBASE_DB and you named the file DEV_SYBASE_DB_1, and you want to import the database into the local Dynamic Data Masking Server, you might enter the following command:

```
server config import "Site\DEV_SYBASE_DB" "DEV_SYBASE_DB_1"
```

If you want to import the DEV_SYBASE_DB_1 file into multiple Dynamic Data Masking Servers, you might enter the following command:

```
server config import "Site\DEV_SYBASE_DB" "DEV_SYBASE_DB_1" -targets admin/
admin@ABC12345:1111 admin/admin@XYZ98765:2222
```

The `import` command uses the following parameters:

object full path

The full path in the Management Console where you want to import the object. If the parent path does not exist in the Management Console tree, the command returns an error. If the object does not exist in the path, Dynamic Data Masking creates the object. If the object exists in the path, the object must be the same object type as the object that you want to import.

For example, you want to import a database into the following location in the Management Console tree:

```
Site\backup\SQL_SERVER_DB
```

The parent path is Site\backup and the object is SQL_SERVER_DB. The parent path must exist in the Management Console tree. If the object does not exist, Dynamic Data Masking creates a database node named SQL_SERVER_DB.

file

The file name of the object that you want to import.

-targets

Indicates that the following arguments are targets.

user

The Dynamic Data Masking user name that you use to log in to the Dynamic Data Masking Server.

pwd

The password for the Dynamic Data Masking user.

host

The Dynamic Data Masking Server host name or IP address.

port

The Dynamic Data Masking Server listener port.

SetDBPassword

Sets the password for the Dynamic Data Masking database. The Dynamic Data Masking Server must be running.

If you do not specify a target Dynamic Data Masking Server, the `setDBPassword` command updates the password of the database on the local Dynamic Data Masking Server. Use the `-targets` parameter to specify multiple target Dynamic Data Masking Servers.

The command uses the following syntax:

```
server config
  setDBPassword <dbpath> <oldDBPassword> <newDBPassword>
  [[-targets] user/pwd@host:port [user/pwd@host:port user/pwd@host:port]]
```

For example, if you have a database named DEV_SYBASE_DB_1 under a root node named Site on the local Dynamic Data Masking Server, you might enter the following command:

```
server config setDBPassword "Site\DEV_SYBASE_DB_1" ddmadmin ddmadmin1
```

If you want to change the password for the database on multiple Dynamic Data Masking Servers, you might enter the following command:

```
server config setDBPassword "Site\DEV_SYBASE_DB_1" ilmuser ilmuser1 -targets admin/
admin@ABC12345:1111 admin/admin@XYZ9876:2222
```

The `setDBPassword` command uses the following parameters:

dbpath

The path to the database in the Management Console tree.

oldDBPassword

The password that Dynamic Data Masking uses to connect to the database.

newDBPassword

The new password that you want Dynamic Data Masking to use to connect to the database.

-targets

Indicates that the following arguments are targets.

user

The Dynamic Data Masking user name that you use to log in to the Dynamic Data Masking Server.

pwd

The password for the Dynamic Data Masking user name.

host

The Dynamic Data Masking Server host name or IP address.

port

The Dynamic Data Masking Server listener port.

Sync

Synchronizes the Management Console tree of one or more target Dynamic Data Masking Servers with the Management Console tree of a source Dynamic Data Masking Server. The source and target Dynamic Data Masking Servers must be running.

The `sync` command synchronizes databases, and security rule sets. You cannot synchronize the Dynamic Data Masking Server, Dynamic Data Masking services, connection rules, and loggers. To copy a Dynamic Data Masking service, you must export the service from the source location and import the service into the target location.

You can specify one source Dynamic Data Masking Server and one or more target Dynamic Data Masking Servers. You must specify at least one target or source. If you specify a source and not a target, the `sync` command uses the local Dynamic Data Masking Server as the target. If you specify a target and not a source, the `sync` command uses the local Dynamic Data masking Server as the source.

The command uses the following syntax:

```
server config
  sync [[-source] user/pwd@host:port]
  [-targets user/pwd@host:port [user/pwd@host:port user/pwd@host:port]]
```

For example, if you want to copy the Management Console tree of a remote Dynamic Data Masking Server into the local Dynamic Data Masking Server, you might enter the following command:

```
server config sync -source admin/admin@ABC12345:1111
```

If you want to copy the Management Console tree of a remote Dynamic Data Masking Server into multiple remote Dynamic Data Masking Servers, you might enter the following command:

```
server config sync -source admin/admin@ABC12345:1111 -targets admin/admin@XYZ9876:2222
admin/admin@DEF1234:3333
```

The `sync` command uses the following parameters:

-source

Indicates that the following argument is a source.

user

The Dynamic Data Masking user name that you use to log in to the Dynamic Data Masking Server.

pwd

The password for the Dynamic Data Masking user name.

host

The Dynamic Data Masking Server host name or IP address.

port

The Dynamic Data Masking Server listener port.

-targets

Indicates that the following arguments are targets.

SetKeyStore

Changes a default keystore to a custom keystore, or a custom keystore to a default keystore.

To change the keystore from default to custom, you provide the custom keystore name and alias in the command. To change the keystore from custom to default, you provide the database user name and password within the command. In both scenarios you must provide the database path.

The command uses the following syntax:

```
server config setKeyStore -path <path> [-storeName <storeName> -alias <alias>] | [-user
<user> -password <password>]
```

Path is a required parameter. If you want to change the keystore from default to custom, both keystore name and alias are required parameters. If you want to change the keystore from custom to default, both user name and password are required parameters. Do not use both sets of parameters in the same command invocation.

You can also set another alias in a database object. The alias must already exist in the designated keystore. You can set the alias, the keystore name, or both.

The setKeyStore command uses the following parameters:

path

Path to the database configured for use with the keystore that you want to change.

storeName

Name of the keystore that you want to change.

alias

Alias name of the keystore.

user

Username to access the database configured for use with the keystore.

password

Password for the database configured for use with the keystore.

Server Service Commands

Use Server Control `server service` commands to manage Dynamic Data Masking services.

Server Control has the following `server service` commands:

- export
- import

Export

Exports a Dynamic Data Masking service. The Dynamic Data Masking Server must be running.

Use the `export` command to export a Dynamic Data Masking service and the connection rules associated with the service into a file. You can then import the file into one or more Dynamic Data Masking Servers.

If you do not specify a source Dynamic Data Masking Server, the `export` command exports the service from the local Dynamic Data Masking Server.

The command uses the following syntax:

```
server service
  export [/y] <DDM Service> <file>
  [[-source] user/pwd@host:port]
```

For example, if you want to export the DDM for Oracle service from the local Dynamic Data Masking Server and name the file DDM_for_Oracle, you might enter the following command:

```
server service export "DDM for Oracle" "DDM_for_Oracle"
```

If you want to export the DDM_for_Oracle service from a remote Dynamic Data Masking Server, you might enter the following command:

```
server service export /y "DDM for Oracle" "DDM_for_Oracle" -source admin/
admin@ABC12345:1111
```

The `export` command uses the following parameters:

/y

Forces the command to continue without user confirmation. If you use the `/y` parameter, Server Control does not require confirmation to overwrite an existing file. For example, you might use the `/y` parameter in a script.

DDM Service

The Dynamic Data Masking service that you want to export.

file

The name of the file that the Dynamic Data Masking service exports to.

-source

Indicates that the following argument is a source.

user

The Dynamic Data Masking user name that you use to log in to the Dynamic Data Masking Server.

pwd

The password for the Dynamic Data Masking user.

host

The Dynamic Data Masking Server host name or IP address.

port

The Dynamic Data Masking Server listener port.

Note: You cannot use the `sync` command to synchronize Dynamic Data Masking services. To copy a service, you must export the service from the source location and import the service into the target location.

Import

Imports a Dynamic Data Masking service. The Dynamic Data Masking Server must be running.

Use the `import` command to import a Dynamic Data Masking service and the connection rules associated with the service into a Dynamic Data Masking Server. The type of service that you import must be the same type of service that you specify as the location to import the service. For example, you can import a DDM for Oracle service into a DDM for Oracle service, but you cannot import a DDM for Oracle service into a DDM for Sybase service.

If you do not specify a target Dynamic Data Masking Server, the `import` command imports the service into the local Dynamic Data Masking Server. Use the `-targets` parameter to specify multiple target Dynamic Data Masking Servers.

The command uses the following syntax:

```
server service
     import <DDM Service> <file>
  [[-targets] user/pwd@host:port[user/pwd@host:port user/pwd@host:port]]
```

For example, if you exported a DDM for Oracle service and named the file DDM_for_Oracle, and you want to import the service into the local Dynamic Data Masking Server, you might enter the following command:

```
server service import "DDM for Oracle" "DDM_for_Oracle"
```

If you want to import the DDM_for_Oracle file into multiple Dynamic Data Masking Servers, you might enter the following command:

```
server service import "DDM for Oracle" "DDM_for_Oracle" -targets admin/
admin@ABC12345:1111 admin/admin@XYZ9876:2222
```

The `import` command uses the following parameters:

DDM Service

The name of the Dynamic Data Masking service that you want to import. You must enter a valid name for the service, such as "DDM for Oracle." If the service that you want to import does not exist, Dynamic Data Masking creates the service in the Management Console tree. If you specify the name of an existing service, the type of service that you import must be the same type as the service that you import into.

file

The file name of the Dynamic Data Masking service that you want to import.

-targets

Indicates that the following arguments are targets.

user

The Dynamic Data Masking user name that you use to log in to the Dynamic Data Masking Server.

pwd

The password for the Dynamic Data Masking user.

host

The Dynamic Data Masking Server host name or IP address.

port

The Dynamic Data Masking Server listener port.

Note: You cannot use the `sync` command to synchronize Dynamic Data Masking services. To copy a service, you must export the service from the source location and import the service into the target location.

CHAPTER 12

Performance Tuning

This chapter includes the following topics:

Performance Tuning Overview

Performance tuning efficiently allocates resources to optimize the performance of Dynamic Data Masking in production and non-production environments. When you implement performance tuning techniques, you help to ensure that the Dynamic Data Masking operations do not affect database and application performance. Use the guidelines in this chapter to reduce overhead and increase Dynamic Data Masking efficiency.

Performance Factors

The primary factors that influence Dynamic Data Masking performance are network traffic and rule processing. You can minimize Dynamic Data Masking overhead by using network traffic statistics to ensure that the Dynamic Data Masking Server has sufficient available resources and by creating efficient rules that do not cause unnecessary rule processing time.

Network Traffic

Network traffic is measured in the number of packets per second. To determine the number of cores that you need to dedicate to the Dynamic Data Masking Server, you must measure the volume of the network traffic between the client and the database.

The Dynamic Data Masking Server can handle 10,000 packets per second per core, or one packet in 100 microseconds. Therefore, if your network traffic is 40,000 packets per second, the Dynamic Data Masking Server requires four cores. You can use a network analyzing tool such as Wireshark to measure your network traffic.

If you install the Dynamic Data Masking Server on the database server, you must verify that the required cores are available to the Dynamic Data Masking Server, in addition to the cores required by the database. If the database server does not have the necessary cores, you can install the Dynamic Data Masking Server on a dedicated machine. However, if the Dynamic Data Masking Server is installed on a separate machine, network performance decreases. Informatica recommends that you install the Dynamic Data Masking Server on the database server if possible.

Resource Consumption Example

The following example illustrates how you can determine the Dynamic Data Masking resource consumption for an Oracle database.

In an Oracle database, the CPU consumption is linearly proportional to the SQL *Net traffic that the Dynamic Data Masking service routes. You can use the SQL *Net and DBlinks traffic values to estimate the amount of CPU that the Dynamic Data Masking service consumes.

The Dynamic Data Masking resource consumption is approximately 1% of the server CPU and has no I/O overhead. The Dynamic Data Masking service requires approximately 1 GB for memory and some disk space for logs.

To calculate the CPU consumption, you must determine the round-trip value, which is the total packet traffic that a client and server sends and receives each second. The total round-trip value includes SQL *Net traffic and DBlinks.

Use the following variables and equations to calculate the CPU consumption:

Variables

X1 = SQL *Net round-trip value

X2 = DBlinks round-trip value

PR = Total round-trip packets for each second

Equations

PR = (X1+X2)*2

CPU Consumption = PR/10,000

Rule Processing

Rule processing causes Dynamic Data Masking overhead. Understanding the factors that contribute to rule processing time is necessary in order to increase efficiency.

Dynamic Data Masking processing time varies based on the matcher or action type that you use. The following list describes the performance impact for each rule type:

Rules that access the database

Dynamic Data Masking takes between 10 and 100 milliseconds to access the database and about 100 milliseconds to handle a packet.

The following matchers and actions require Dynamic Data Masking to access the database:

- PL/SQL Function matcher
- PL/SQL Function action
- The Mask action if the statement contains an asterisk in the column list

Rules that require SQL parsing

Dynamic Data takes between one and ten milliseconds to perform SQL parsing, depending on the complexity of the statement.

The following matchers and actions require SQL parsing:

- From Clause Object matcher
- SQL Syntax matcher
- Mask action
- Replace Table action

Other types of rules

Rules that do not access the database or require SQL parsing take Dynamic Data Masking between 10 and 50 microseconds (0.05 milliseconds) to complete.

Rule Performance Factors

When you configure Dynamic Data Masking rules, consider the types of rules that you create and the types of queries that you send to the database in order to increase efficiency and accurately measure Dynamic Data Masking overhead.

The following factors affect Dynamic Data Masking processing time:

Rule efficiency

Define a precise rule matcher to ensure that Dynamic Data Masking only applies the rule to the necessary queries. You can use the Regular Expression Text matcher to determine whether the query contains the relevant table. The Text matcher requires minimal processing time compared to other matchers, such as the PL/SQL Function matcher and the SQL Syntax matcher.

Size of the query

The size of the database query affects how long it takes Dynamic Data Masking to parse the query. Longer queries take longer to parse, and therefore increase processing time.

The following table lists the Dynamic Data Masking overhead for different query sizes. The overhead percentages are for query execution, not end to end transactions.

Query Size	Overhead
Small query Example: `SELECT c_data FROM ddmuser1.sybcustomer`	10% to 30%
Medium query Example: `SELECT c_data FROM ddmuser1.sybcustomer order by c_id`	20% to 40%
Large query Example: `select c_data from ddmuser1.sybcustomer where c_id in (select c_id from ddmuser1.sybcustomer)`	30% to 60%

Result set size

A larger result set size does not increase Dynamic Data Masking overhead because Dynamic Data Masking does not create additional work for the database. If you measure Dynamic Data Masking overhead as a percentage of the total time it takes to return the result set, the overhead percentage decreases as the result set size increases. Therefore, if you execute a query without fetching a result set, the overhead percentage is higher than it would be for an end to end transaction.

Note: In stored procedure masking, the result set size does increase overhead because Dynamic Data Masking must create temporary tables and populate the tables with masked data.

Complexity of the rules

Complex rules increase Dynamic Data Masking processing time. For example, if you want to mask all the columns in a table named table1, which has five columns, you send the following query to the database:

```
SELECT col1, col2, col3, col4, col5 from table1;
```

You can configure Dynamic Data Masking rules to mask the output in multiple ways, including the following options:

- Option 1. Create one rule that masks every column if the table name is table1.
- Option 2. Create a rule folder that has five rules. Each rule matches the table name and a column name, and masks the column.

Option 2 increases processing time because there are more masking rules. Simple, efficient rules, such as the rule in Option 1, reduce processing time.

Performance of the rewritten query

When Dynamic Data Masking rewrites the original database query, the rewritten query is not optimized. For example, the rewritten query may have a GROUP BY or ORDER BY clause that increases processing time.

Stored Procedure Masking

Unlike query masking, stored procedure masking overhead increases as the result set size increases. This is because Dynamic Data Masking must create temporary tables and populate the tables with data before running the stored procedure. The following table lists the overhead percentages for various result set sizes:

Number of Rows in the Result Set	Overhead
200	10%
500	20%
2,000	30%
8,000	40%
16,000	60%
50,000	100%

Log Performance

Log files use server resources to record service information and events. The tracing level determines the amount of information that the log stores.

Logs consume system resources and can slow down performance. To improve log performance, you can change the tracing level to reduce the amount of information that the log stores. By default, each log uses the information tracing level. The information tracing level is a high impact tracing level and uses the most server resources.

Note: If you change the tracing level, it does not affect the tracing level for SQL Server.

The following table describes the different tracing levels:

Tracing Level	Description
Information	The default log level. Logs all information messages and provides comprehensive information about the Dynamic Data Masking service. The information that the log provides is useful if you encounter an issue with Dynamic Data Masking operations. You can refer to the logs to troubleshoot the problem. Performance impact is high.
Warn	Logs warning messages from the Dynamic Data Masking service. Performance impact is moderate.
Error	Logs error messages and instances when a user connection is force closed. Performance impact is low.

Configuring the Tracing Level

Configure the tracing level in the Management Console with a low impact tracing level to reduce resource consumption.

1. In the Management Console, click the Dynamic Data Masking Server.
2. Select **Tree** > **Edit**.

   The Dynamic Data Masking Server configuration window appears.
3. Select the log level from the Log Level menu.
4. Click **OK**.

User Stack Limit

High user stack limits cause high CPU consumption and connection refusal.

Set the user stack limit, or ulimit, to 1,024 kilobytes to ensure that system can create threads. To view the user stack limit, enter `#ulimit -s` in the command shell. The system returns the value in kilobytes.

Calculating the User Stack Limit

An insufficient number of available threads cause high CPU load and client request delays. Before you change the user stack limit, verify the minimum number of required threads necessary for the system.

Use the following formula to calculate the minimum number of threads:

```
<number of concurrent sessions> x 10
```

Insufficient Thread Error

The following error appears in server.log if the number of open files is insufficient:

```
12/16 13:24:51,597 [dnr-1] WARN - Service dnr.createClientPeer: received IOException
while listening:
java.io.IOException: Too many open files
```

CHAPTER 13

Troubleshooting

This chapter includes the following topics:

Troubleshooting Overview

The troubleshooting procedures described in this chapter aim to help you resolve commonly known issues. Contact Informatica Global Customer Support if the troubleshooting procedures do not help you resolve a problem.

Log Archive

You can create an encrypted log archive file that you can send to Informatica Global Customer Support to troubleshoot issues with Dynamic Data Masking.

To create the log archive in Server Control, use the `server support` command. You can use the `-noencrypt` parameter to create an unencrypted log archive. To encrypt an unencrypted log archive, use the `server encrypt` command.

To create an encrypted log archive in the Management Console, select the Dynamic Data Masking Server in the Management Console tree and click **Tree** > **Support**. The default file name is support_encrypted_<date>.zip. You cannot create an unencrypted log archive in the Management Console.

If you have a case open with Informatica Global Customer Support, Informatica recommends that you include the case number in the name of the file.

The log archive includes the following files:

- All files in the `<Dynamic Data Masking installation>/cfg` directory with the exception of the following files:
  - `config.cfg`. The archive contains `config.cfg.xml`, which is `config.cfg` in XML format. The XML file does not contain user passwords.
  - `config.pbk`

- All files in the `<Dynamic Data Masking installation>/lib` directory.
- All files in the `<Dynamic Data Masking installation>/log` directory.
- jvm.params

The following files are generated at runtime and included in the archive:

- DDM for <database>.txt. Contains statistics for running the Dynamic Data Masking services.
- `environment.txt`. Contains operating system environment parameters.
- `java_properties.txt`. Contains Java system properties.

Note: The log archive does not contain user passwords. Dynamic Data Masking replaces encrypted user passwords with new encrypted passwords.

Database Connections

This section provides solutions for problems that you might encounter when you connect to a database with Dynamic Data Masking.

No Listener Defined

The `TNS: No Listener` error indicates that clients cannot reach the Dynamic Data Masking listener port.

If you receive this error, verify that the firewall configuration has the Dynamic Data Masking listener port and administrator port open. For example, the default Dynamic Data Masking listener port for Oracle is 1525 and the default administrator port is 8195.

Note: The `TNS: No Listener` error is an error for connections to Oracle databases. A similar error might appear for a different database.

Database Refuses Connection

If you do not define the service name for the database, the target database refuses a connection. If the database refuses the connection, you receive the `TNS connection refused` error.

To resolve the TNS connection error, define the database service name.

1. Right-click on the database name and select **Edit**.
2. Click **Add** and define a service name.
3. Click **OK.**

Dynamic Data Masking Service Refuses Connection Request

If the Dynamic Data Masking service refuses a connection from a client, you receive an `IO Exception, Connection Refused` error.

The following table describes the possible reasons for the connection refusal:

Reason	Solution
Error in the listener address	Run a ping command from the system running the Management Console to the Dynamic Data Masking service. Use the Dynamic Data Masking host name. For example: `ping 10.65.48.73`.
Error in the listener port	Verify that the firewall has the Dynamic Data Masking listener port and administrator port open.
Undefined service name	Verify the database service name. Add a service name if one does not exist.
Database server is down	Restart the database.

Cleanup Commands

Dynamic Data Masking runs cleanup commands before closing a connection pool to remove privileges from the Dynamic Data Masking administrator.

Dynamic Data Masking runs the following cleanup commands:

Oracle

```
ALTER SESSION SET CURRENT_SCHEMA = <DDM admin user>
```

Sybase

```
SET PROXY <DDM admin user>
```

```
USE <DDM catalog>
```

Microsoft SQL Server

```
REVERT
```

```
USE "<DDM catalog>"
```

DB2

```
SET CURRENT SCHEMA <DDM admin user>
```

```
SET CURRENT PATH <DDM admin current path>
```

APPENDIX A

Database Keywords

This appendix includes the following topic:

Database Keywords

Dynamic Data Masking reserves certain keywords that the database parsers cannot parse. If you use these keywords in an SQL query when you form a rule, the query might fail. For example, if a column name in the SQL query contains one of these keywords, you might receive an invalid character error.

The following table lists the database keywords for Microsoft SQL Server and Oracle databases:

Keywords for Microsoft SQL Server Databases	Keywords for Oracle Databases
ABSOLUTE	ABSENT
APPLY	ANY_VALUE
AT	APPROX_COUNT
ATTRIBUTES	APPROX_COUNT_DISTINCT
AUTO	APPROX_COUNT_DISTINCT_AGG
BASE64	APPROX_COUNT_DISTINCT_DETAIL
BIGINT	APPROX_MEDIAN, DETERMINISTIC
BINARY	APPROX_PERCENTILE, DETERMINISTIC
BIT	APPROX_PERCENTILE_AGG
BLOB	APPROX_PERCENTILE_DETAIL
CALL	APPROX_SUM,APPROX_RANK
CHARACTER	ATTRIBUTES
CLOB	AUTO

Keywords for Microsoft SQL Server Databases	Keywords for Oracle Databases
COLUMNS	BASE64
CONTAINED	BEGIN
CONTENT	BIGINT
DATE	BINARY
DATETIME	BIT
DATETIME2	BLOB
DATETIMEOFFSET	CHARACTER
DEC	CLOB
DECIMAL	COLUMNS
DELAY	CONTENT
DISABLE	CONVERT
DISABLE_OPTIMIZED_NESTED_LOOP	COLLATE
DYNAMIC	CONNECT BY PRIOR
ELEMENTS	CONNECT_BY_ISCYCLE
EMPTY	CONNECT_BY_ISLEAF
EVALNAME	CONNECT_BY_ROOT
EXPAND	CONTAINERS
EXTERNALPUSHDOWN	CONTAINS
FAST	CROSS
FAST_FORWARD	CURSOR
FIRST	DATETIME
FLOAT	DATETIME2
FN	DATETIMEOFFSET
FORCE	DAYS
FORCED	DBCLOB
FORCESCAN	DEC
FORCESEEK	DECFLOAT

Keywords for Microsoft SQL Server Databases	Keywords for Oracle Databases
FORWARD_ONLY	DOCUMENT
GLOBAL	DOUBLE
HASH	ELEMENTS
HINT	EMPTY
IIF	END
IMAGE	ENTITYESCAPING
INCLUDING	EVALNAME
INDICATOR	EXCEPT
INSENSITIVE	EXCLUDING
INT	EXIT
INTEGER	EXPLICIT
KEEP	FETCH
KEEPFIXED	FIRST
KEYSET	FULL
LABEL	HOLDLOCK
LARGE	HOUR
LAST	IF
LEVEL	ILIKE
LOCAL	IMAGE
LOGIN	INCLUDING
LOOP	INDICATOR
MAXDOP	INNER
MAXRECURSION	INT
MAX_GRANT_PERCENT	IREGEXP
MIN_GRANT_PERCENT	JOIN
MONEY	JOINS
NATURAL	LARGE

Keywords for Microsoft SQL Server Databases	Keywords for Oracle Databases
NEXT	LAST
NO	LIKE2
NOCOUNT	LIKE4
NOEXPAND	LIKEC
NOLOCK	LOCKED
NONE	MONEY
NOWAIT	MONTHS
NTEXT	MULTISET
NUMERIC	NATURAL
NVARCHAR	NCHAR
OBJECT	NO
OFFSET	NOENTITYESCAPING
OPTIMISTIC	NOHOLDLOCK
OPTIMIZE	NOSCHEMACHECK
ORDINALITY	NTEXT
OUTPUT	NULLS
PAGLOCK	NUMERIC
PARAMETERIZATION	NVARCHAR
PARSE	OBJECT
PATH	OFF
PRECEDING	OFFSET
PRECISION	OPENXML
PRESERVE	ORA_SHARD_ID
QUOTED_IDENTIFIER	ORDINALITY
RAW	OUTER
READCOMMITTED	OVER
READCOMMITTEDLOCK	PARTITION

Keywords for Microsoft SQL Server Databases	Keywords for Oracle Databases
READPAST	PASSING
READUNCOMMITTED	PATH
READ_ONLY	PRESERVE
REAL	QUALIFY
RECOMPILE	QUOTED_IDENTIFIER
REDISTRIBUTE	REAL
REDUCE	REF
REF	REGEXP
RELATIVE	RETURNING
REPEATABLE	RLIKE
REPEATABLEREAD	ROOT
RESULT	ROUND_TIES_TO_EVEN
ROBUST	ROWCOUNT
ROOT	SAMPLE
ROW	SCHEMACHECK
ROWLOCK	SEMI
ROWS	SEQUENCE
SCROLL	SHARDS
SCROLL_LOCKS	SHARED
SERIALIZABLE	SIBLINGS
SETS	SMALLDATETIME
SIMPLE	SMALLMONEY
SMALLDATETIME	SOME
SMALLINT	STRIP
SMALLMONEY	SUBPARTITION
SNAPSHOT	SYS_CONNECT_BY_PATH
SPATIAL_WINDOW_MAX_CELLS	SYSTIMESTAMP

Keywords for Microsoft SQL Server Databases	Keywords for Oracle Databases
START	TEXT
STATIC	TINYINT
STRING_SPLIT	TO_UTC_TIMESTAMP_TZ
SYSTEM	UPDATE
SYSTEM_TIME	USE
TABLOCK	USING
TABLOCKX	VARBINARY
TEXT	WAIT
TINYINT	WHEN
TRY_CAST	WITH
TRY_CONVERT	XMLAGG
TRY_PARSE	XMLATTRIBUTES
TYPE_WARNING	XMLCAST
UNBOUNDED	XMLCOLATTVAL
UNDEFINED	XMLCONCAT
UNKNOWN	XMLDOCUMENT
UPDLOCK	XMLELEMENT
USING	XMLEXISTS
VARBINARY	XMLFOREST
VARCHAR	XMLNAMESPACES
VIEWS	XMLPARSE
WRITE	XMLPI
XLOCK	XMLQUERY
XML	XMLROW
XMLAGG	XMLSERIALIZE
XMLATTRIBUTES	XMLTABLE
XMLCAST	XMLXSROBJECTID

Keywords for Microsoft SQL Server Databases	Keywords for Oracle Databases
XMLCOLATTVAL	
XMLCONCAT	
XMLDOCUMENT	
XMLELEMENT	
XMLEXISTS	
XMLFOREST	
XMLNAMESPACES	
XMLPARSE	
XMLPI	
XMLQUERY	
XMLROW	
XMLSERIALIZE	
XMLTABLE	
ZONE	

INDEX