Informatica

Informatica® Dynamic Data Masking
9.8.2

User Guide

Informatica Dynamic Data Masking User Guide
9.8.2
October 2016

© Copyright Informatica LLC 1993, 2018

This software and documentation contain proprietary information of Informatica LLC and are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright law. Reverse engineering of the software is prohibited. No part of this document may be reproduced or transmitted in any form, by any means (electronic, photocopying, recording or otherwise) without prior consent of Informatica LLC. This Software may be protected by U.S. and/or international Patents and other Patents Pending.

Use, duplication, or disclosure of the Software by the U.S. Government is subject to the restrictions set forth in the applicable software license agreement and as provided in DFARS 227.7202-1(a) and 227.7702-3(a) (1995), DFARS 252.227-7013©(1)(ii) (OCT 1988), FAR 12.212(a) (1995), FAR 52.227-19, or FAR 52.227-14 (ALT III), as applicable.

The information in this product or documentation is subject to change without notice. If you find any problems in this product or documentation, please report them to us in writing.

Informatica, Informatica Platform, Informatica Data Services, PowerCenter, PowerCenterRT, PowerCenter Connect, PowerCenter Data Analyzer, PowerExchange, PowerMart, Metadata Manager, Informatica Data Quality, Informatica Data Explorer, Informatica B2B Data Transformation, Informatica B2B Data Exchange Informatica On Demand, Informatica Identity Resolution, Informatica Application Information Lifecycle Management, Informatica Complex Event Processing, Ultra Messaging, Informatica Master Data Management, and Live Data Map are trademarks or registered trademarks of Informatica LLC in the United States and in jurisdictions throughout the world. All other company and product names may be trade names or trademarks of their respective owners.

Portions of this software and/or documentation are subject to copyright held by third parties, including without limitation: Copyright DataDirect Technologies. All rights reserved. Copyright © Sun Microsystems. All rights reserved. Copyright © RSA Security Inc. All Rights Reserved. Copyright © Ordinal Technology Corp. All rights reserved. Copyright © Aandacht c.v. All rights reserved. Copyright Genivia, Inc. All rights reserved. Copyright Isomorphic Software. All rights reserved. Copyright © Meta Integration Technology, Inc. All rights reserved. Copyright © Intalio. All rights reserved. Copyright © Oracle. All rights reserved. Copyright © Adobe Systems Incorporated. All rights reserved. Copyright © DataArt, Inc. All rights reserved. Copyright © ComponentSource. All rights reserved. Copyright © Microsoft Corporation. All rights reserved. Copyright © Rogue Wave Software, Inc. All rights reserved. Copyright © Teradata Corporation. All rights reserved. Copyright © Yahoo! Inc. All rights reserved. Copyright © Glyph & Cog, LLC. All rights reserved. Copyright © Thinkmap, Inc. All rights reserved. Copyright © Clearpace Software Limited. All rights reserved. Copyright © Information Builders, Inc. All rights reserved. Copyright © OSS Nokalva, Inc. All rights reserved. Copyright Edifecs, Inc. All rights reserved. Copyright Cleo Communications, Inc. All rights reserved. Copyright © International Organization for Standardization 1986. All rights reserved. Copyright © ej-technologies GmbH. All rights reserved. Copyright © Jaspersoft Corporation. All rights reserved. Copyright © International Business Machines Corporation. All rights reserved. Copyright © yWorks GmbH. All rights reserved. Copyright © Lucent Technologies. All rights reserved. Copyright © University of Toronto. All rights reserved. Copyright © Daniel Veillard. All rights reserved. Copyright © Unicode, Inc. Copyright IBM Corp. All rights reserved. Copyright © MicroQuill Software Publishing, Inc. All rights reserved. Copyright © PassMark Software Pty Ltd. All rights reserved. Copyright © LogiXML, Inc. All rights reserved. Copyright © 2003-2010 Lorenzi Davide, All rights reserved. Copyright © Red Hat, Inc. All rights reserved. Copyright © The Board of Trustees of the Leland Stanford Junior University. All rights reserved. Copyright © EMC Corporation. All rights reserved. Copyright © Flexera Software. All rights reserved. Copyright © Jinfonet Software. All rights reserved. Copyright © Apple Inc. All rights reserved. Copyright © Telerik Inc. All rights reserved. Copyright © BEA Systems. All rights reserved. Copyright © PDFlib GmbH. All rights reserved. Copyright © Orientation in Objects GmbH. All rights reserved. Copyright © Tanuki Software, Ltd. All rights reserved. Copyright © Ricebridge. All rights reserved. Copyright © Sencha, Inc. All rights reserved. Copyright © Scalable Systems, Inc. All rights reserved. Copyright © jQWidgets. All rights reserved. Copyright © Tableau Software, Inc. All rights reserved. Copyright© MaxMind, Inc. All Rights Reserved. Copyright © TMate Software s.r.o. All rights reserved. Copyright © MapR Technologies Inc. All rights reserved. Copyright © Amazon Corporate LLC. All rights reserved. Copyright © Highsoft. All rights reserved. Copyright © Python Software Foundation. All rights reserved. Copyright © BeOpen.com. All rights reserved. Copyright © CNRI. All rights reserved.

This product includes software developed by the Apache Software Foundation (http://www.apache.org/), and/or other software which is licensed under various versions of the Apache License (the "License"). You may obtain a copy of these Licenses at http://www.apache.org/licenses/. Unless required by applicable law or agreed to in writing, software distributed under these Licenses is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the Licenses for the specific language governing permissions and limitations under the Licenses.

This product includes software which was developed by Mozilla (http://www.mozilla.org/), software copyright The JBoss Group, LLC, all rights reserved; software copyright © 1999-2006 by Bruno Lowagie and Paulo Soares and other software which is licensed under various versions of the GNU Lesser General Public License Agreement, which may be found at http:// www.gnu.org/licenses/lgpl.html. The materials are provided free of charge by Informatica, "as-is", without warranty of any kind, either express or implied, including but not limited to the implied warranties of merchantability and fitness for a particular purpose.

The product includes ACE(TM) and TAO(TM) software copyrighted by Douglas C. Schmidt and his research group at Washington University, University of California, Irvine, and Vanderbilt University, Copyright (©) 1993-2006, all rights reserved.

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (copyright The OpenSSL Project. All Rights Reserved) and redistribution of this software is subject to terms available at http://www.openssl.org and http://www.openssl.org/source/license.html.

This product includes Curl software which is Copyright 1996-2013, Daniel Stenberg, <daniel@haxx.se>. All Rights Reserved. Permissions and limitations regarding this software are subject to terms available at http://curl.haxx.se/docs/copyright.html. Permission to use, copy, modify, and distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies.

The product includes software copyright 2001-2005 (©) MetaStuff, Ltd. All Rights Reserved. Permissions and limitations regarding this software are subject to terms available at http://www.dom4j.org/ license.html.

The product includes software copyright © 2004-2007, The Dojo Foundation. All Rights Reserved. Permissions and limitations regarding this software are subject to terms available at http://dojotoolkit.org/license.

This product includes ICU software which is copyright International Business Machines Corporation and others. All rights reserved. Permissions and limitations regarding this software are subject to terms available at http://source.icu-project.org/repos/icu/icu/trunk/license.html.

This product includes software copyright © 1996-2006 Per Bothner. All rights reserved. Your right to use such materials is set forth in the license which may be found at http:// www.gnu.org/software/ kawa/Software-License.html.

This product includes OSSP UUID software which is Copyright © 2002 Ralf S. Engelschall, Copyright © 2002 The OSSP Project Copyright © 2002 Cable & Wireless Deutschland. Permissions and limitations regarding this software are subject to terms available at http://www.opensource.org/licenses/mit-license.php.

This product includes software developed by Boost (http://www.boost.org/) or under the Boost software license. Permissions and limitations regarding this software are subject to terms available at http:/ /www.boost.org/LICENSE_1_0.txt.

This product includes software copyright © 1997-2007 University of Cambridge. Permissions and limitations regarding this software are subject to terms available at http:// www.pcre.org/license.txt.

This product includes software copyright © 2007 The Eclipse Foundation. All Rights Reserved. Permissions and limitations regarding this software are subject to terms available at http:// www.eclipse.org/org/documents/epl-v10.php and at http://www.eclipse.org/org/documents/edl-v10.php.

This product includes software licensed under the terms at http://www.tcl.tk/software/tcltk/license.html, http://www.bosrup.com/web/overlib/?License, http://www.stlport.org/doc/ license.html, http://asm.ow2.org/license.html, http://www.cryptix.org/LICENSE.TXT, http://hsqldb.org/web/hsqlLicense.html, http://httpunit.sourceforge.net/doc/ license.html, http://jung.sourceforge.net/license.txt , http://www.gzip.org/zlib/zlib_license.html, http://www.openldap.org/software/release/license.html, http://www.libssh2.org, http://slf4j.org/license.html, http://www.sente.ch/software/OpenSourceLicense.html, http://fusesource.com/downloads/license-agreements/fuse-message-broker-v-5-3- license-agreement; http://antlr.org/license.html; http://aopalliance.sourceforge.net/; http://www.bouncycastle.org/licence.html; http://www.jgraph.com/jgraphdownload.html; http://www.jcraft.com/jsch/LICENSE.txt; http://jotm.objectweb.org/bsd_license.html; . http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231; http://www.slf4j.org/license.html; http://nanoxml.sourceforge.net/orig/copyright.html; http://www.json.org/license.html; http://forge.ow2.org/projects/javaservice/, http://www.postgresql.org/about/licence.html, http://www.sqlite.org/copyright.html, http://www.tcl.tk/software/tcltk/license.html, http://www.jaxen.org/faq.html, http://www.jdom.org/docs/faq.html, http://www.slf4j.org/license.html; http://www.iodbc.org/dataspace/iodbc/wiki/iODBC/License; http://www.keplerproject.org/md5/license.html; http://www.toedter.com/en/jcalendar/license.html; http://www.edankert.com/bounce/index.html; http://www.net-snmp.org/about/license.html; http://www.openmdx.org/#FAQ; http://www.php.net/license/3_01.txt; http://srp.stanford.edu/license.txt; http://www.schneier.com/blowfish.html; http://www.jmock.org/license.html; http://xsom.java.net; http://benalman.com/about/license/; https://github.com/CreateJS/EaselJS/blob/master/src/easeljs/display/Bitmap.js; http://www.h2database.com/html/license.html#summary; http://jsoncpp.sourceforge.net/LICENSE; http://jdbc.postgresql.org/license.html; http://protobuf.googlecode.com/svn/trunk/src/google/protobuf/descriptor.proto; https://github.com/rantav/hector/blob/master/LICENSE; http://web.mit.edu/Kerberos/krb5-current/doc/mitK5license.html; http://jibx.sourceforge.net/jibx-license.html; https://github.com/lyokato/libgeohash/blob/master/LICENSE; https://github.com/hjiang/jsonxx/blob/master/LICENSE; https://code.google.com/p/lz4/; https://github.com/jedisct1/libsodium/blob/master/LICENSE; http://one-jar.sourceforge.net/index.php?page=documents&file=license; https://github.com/EsotericSoftware/kryo/blob/master/license.txt; http://www.scala-lang.org/license.html; https://github.com/tinkerpop/blueprints/blob/master/LICENSE.txt; http://gee.cs.oswego.edu/dl/classes/EDU/oswego/cs/dl/util/concurrent/intro.html; https://aws.amazon.com/asl/; https://github.com/twbs/bootstrap/blob/master/LICENSE; https://sourceforge.net/p/xmlunit/code/HEAD/tree/trunk/LICENSE.txt; https://github.com/documentcloud/underscore-contrib/blob/master/LICENSE, and https://github.com/apache/hbase/blob/master/LICENSE.txt.

This product includes software licensed under the Academic Free License (http://www.opensource.org/licenses/afl-3.0.php), the Common Development and Distribution License (http://www.opensource.org/licenses/cddl1.php) the Common Public License (http://www.opensource.org/licenses/cpl1.0.php), the Sun Binary Code License Agreement Supplemental License Terms, the BSD License (http:// www.opensource.org/licenses/bsd-license.php), the new BSD License (http://opensource.org/licenses/BSD-3-Clause), the MIT License (http://www.opensource.org/licenses/mit-license.php), the Artistic License (http://www.opensource.org/licenses/artistic-license-1.0) and the Initial Developer's Public License Version 1.0 (http://www.firebirdsql.org/en/initial-developer-s-public-license-version-1-0/).

This product includes software copyright © 2003-2006 Joe Walnes, 2006-2007 XStream Committers. All rights reserved. Permissions and limitations regarding this software are subject to terms available at http://xstream.codehaus.org/license.html. This product includes software developed by the Indiana University Extreme! Lab. For further information please visit http://www.extreme.indiana.edu/.

This product includes software Copyright (c) 2013 Frank Balluffi and Markus Moeller. All rights reserved. Permissions and limitations regarding this software are subject to terms of the MIT license.

See patents at https://www.informatica.com/legal/patents.html.

DISCLAIMER: Informatica LLC provides this documentation "as is" without warranty of any kind, either express or implied, including, but not limited to, the implied warranties of noninfringement, merchantability, or use for a particular purpose. Informatica LLC does not warrant that this software or documentation is error free. The information provided in this software or documentation may include technical inaccuracies or typographical errors. The information in this software and documentation is subject to change at any time without notice.

NOTICES

This Informatica product (the "Software") includes certain drivers (the "DataDirect Drivers") from DataDirect Technologies, an operating company of Progress Software Corporation ("DataDirect") which are subject to the following terms and conditions:

1. THE DATADIRECT DRIVERS ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT.
2. IN NO EVENT WILL DATADIRECT OR ITS THIRD PARTY SUPPLIERS BE LIABLE TO THE END-USER CUSTOMER FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL OR OTHER DAMAGES ARISING OUT OF THE USE OF THE ODBC DRIVERS, WHETHER OR NOT INFORMED OF THE POSSIBILITIES OF DAMAGES IN ADVANCE. THESE LIMITATIONS APPLY TO ALL CAUSES OF ACTION, INCLUDING, WITHOUT LIMITATION, BREACH OF CONTRACT, BREACH OF WARRANTY, NEGLIGENCE, STRICT LIABILITY, MISREPRESENTATION AND OTHER TORTS.

Publication Date: 2018-06-25

Table of Contents

Preface 8

Informatica Resources. 8

Informatica Network. 8

Informatica Knowledge Base. 8

Informatica Documentation. 8

Informatica Product Availability Matrixes. 9

Informatica Velocity. 9

Informatica Marketplace. 9

Informatica Global Customer Support. 9

Chapter 1: Introduction to Dynamic Data Masking. 10

Dynamic Data Masking Overview. 10

Dynamic Data Masking Architecture. 11

Dynamic Data Masking Components. 11

Dynamic Data Masking Process. 13

Management Console. 13

Navigation. 14

Menu. 14

Management Console Tree. 15

Logging In to the Management Console. 16

Domain Management. 17

Security Rule Set. 17

Chapter 2: Rules. 18

Rules Overview. 18

Rule Components. 18

Rule Trees. 19

Rule Tree Components. 19

Rule Tree Efficiency. 20

Security Rule Sets. 20

Creating a Security Rule Set. 20

Rule Folders. 21

Creating a Connection Rule Folder. 21

Creating a Security Rule Folder. 22

Adding a Rule to a Rule Folder. 22

Rule Management. 23

Editing a Rule 23

Enabling and Disabling a Rule. 23

Deleting a Rule or Rule Folder. 24

Rule Export and Import. 24

Rule Example. . . . 24
Masking Rule. . . . 24
Blocking Rule. . . . 25

Chapter 3: Connection Rules. . . . 26

Connection Rules Overview. . . . 26
Connection Rule Parameters. . . . 26
Connection Rule Matchers. . . . 27
All Incoming Connections Matcher. . . . 28
Check Database Connection Matcher. . . . 28
Check Database DSN Matcher. . . . 28
Check Database URL Matcher. . . . 29
Check Property Matcher. . . . 29
Check Server Name Matcher. . . . 29
Client Information Matcher. . . . 30
Client/Application Information Matcher. . . . 30
Current Target Database Matcher. . . . 31
Incoming DDM Listener Port Matcher. . . . 31
Connection Rule Actions. . . . 31
Load Control Action. . . . 32
Processing Actions. . . . 34
Creating a Connection Rule. . . . 34
Connection Rule Export and Import. . . . 34
Connection Rule Export. . . . 35
Connection Rule Import. . . . 35

Chapter 4: Security Rules. . . . 36

Security Rules Overview . . . 36
Security Rule Parameters. . . . 37
Security Rule Matchers . . . 37
Any Matcher. . . . 37
From Clause Object Matcher. . . . 38
Java Matcher. . . . 38
Parser Matcher. . . . 40
PL/SQL Function Matcher. . . . 41
SQL Syntax Matcher. . . . 49
Symbol Matcher. . . . 49
Text Matcher. . . . 51
Time of Day Matcher. . . . 56
Security Rule Actions. . . . 56
Block Statement Action. . . . 56
Define Symbol Action. . . . 57
Folder Action. . . . 58

Java Action. 58
Log Message Action. 59
Mask Action. 60
Nothing Action. 63
PL/SQL Function Action. 64
Replace Table Action. 65
Rewrite Action. 67
Search and Replace Action. 67
Processing Actions. 68
Creating a Security Rule. 68
Security Rule Export and Import. 69
Security Rule Export. 69
Security Rule Import. 69
Security Rule Set Export and Import. 70
Security Rule Set Export. 70
Security Rule Set Import. 71

Chapter 5: Security Rule Set Simulator 72

Security Rule Set Simulator Overview. 72
Running the Security Rule Set Simulator. 72
Working with the Security Rule Set Simulator. 73
Security Rule Set Simulator Parameters 73
Simulating a Rule. 75
Security Rule Set Simulator Example. 76

Chapter 6: Masking Functions. 83

Masking Functions Overview. 83
Masking Function Examples. 83
Column and Alias Regular Expression Symbols. 84
Column Alias \(alias). 84
Column Alias \(col). 84
Scrambling Functions. 85
Reverse Masking Function. 85
Column Level Masking. 86

Appendix A: XML Functions Reference. 87

XML Functions Overview. 87
IBM DB2. 87
Microsoft SQL Server. 88
Oracle. 89
Sybase. 90

Appendix B: Glossary 91

Index 96

Preface

The *Informatica Dynamic Data Masking User Guide* describes how to implement Dynamic Data Masking to protect sensitive information in a production environment. This guide assumes that you have knowledge of your operating systems and relational database systems, which includes the database engines, flat files, and mainframe systems in your environment.

Informatica Resources

Informatica Network

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

As a member, you can:

- Access all of your Informatica resources in one place.
- Search the Knowledge Base for product resources, including documentation, FAQs, and best practices.
- View product availability information.
- Review your support cases.
- Find your local Informatica User Group Network and collaborate with your peers.

Informatica Knowledge Base

Use the Informatica Knowledge Base to search Informatica Network for product resources such as documentation, how-to articles, best practices, and PAMs.

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

Informatica Documentation

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

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

Informatica Product Availability Matrixes

Product Availability Matrixes (PAMs) indicate the versions of operating systems, databases, and other types of data sources and targets that a product release supports. If you are an Informatica Network member, you can access PAMs at https://network.informatica.com/community/informatica-network/product-availability-matrices.

Informatica Velocity

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

If you are an Informatica Network member, you can access Informatica Velocity resources at http://velocity.informatica.com.

If you have questions, comments, or ideas about Informatica Velocity, contact Informatica Professional Services at ips@informatica.com.

Informatica Marketplace

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

Informatica Global Customer Support

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

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

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

CHAPTER 1

Introduction to Dynamic Data Masking

This chapter includes the following topics:

- Dynamic Data Masking Overview, 10
- Dynamic Data Masking Architecture, 11
- Dynamic Data Masking Process, 13
- Management Console, 13
- Domain Management, 17

Dynamic Data Masking 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.

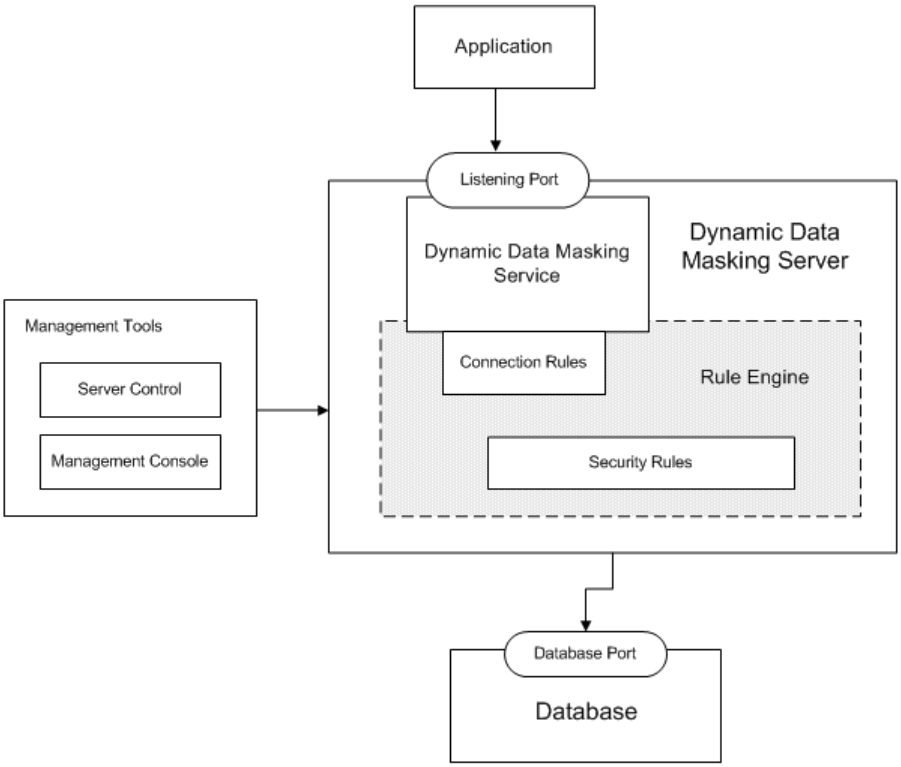
You can use Dynamic Data Masking to mask or prevent access to sensitive data stored in production and non-production databases. You set up the rules to specify the database requests to intercept and the masking actions to apply. Dynamic Data Masking monitors incoming database requests from the application. Dynamic Data Masking applies the data masking rules to the database request before it sends it to the database. The database processes the modified request as normal and returns masked results to Dynamic Data Masking. Dynamic Data Masking then sends the results to the application.

You can use Dynamic Data Masking to mask data for specific types of database requests or you can restrict access to data from certain groups within an organization. For example, you can create a rule to apply a masking function to credit card numbers when the database request comes from a support team member. When the database sends the data back to the application, the support team member sees the masked numbers instead of the real credit card numbers.

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 routes incoming database requests.

You can run the following Dynamic Data Masking services:

- DDM for DB2. Listens for and routes database requests for an IBM DB2 database.
- DDM for Data Vault. Listens for and routes database requests for Data Vault.
- DDM for Hive. Listens for and routes database requests for a Hive database.
- 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.
- DDM for Microsoft SQL Server. Listens for and routes database requests for a Microsoft SQL Server database.
- 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.

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.

Navigation

The Management Console is a user interface that you use to create and configure the Dynamic Data Masking domains, databases, services, and rule sets.

The left side of the Management Console contains a hierarchical tree that represents the components of Dynamic Data Masking. The tree contains domain, database, server, service, and rule set nodes. Use the menu and toolbar above the tree to add, edit, remove, cut, copy, paste, and sort items within the tree. You can drag and drop nodes to change the placement of the node within the tree.

The right side of the Management Console contains a view pane with parameters of the node you select in the tree.

Menu

The Management Console Tree menu contains options that you use to edit the nodes within the Management Console tree. The toolbar contains shortcuts to options in the menu. Available menu items change based on the type of node you select in the tree. Items that are not available for the tree node you select are grayed out.

The Management Console Tree menu contains the following options:

Menu Item	Action
Login	Shows the server host, port, and username for the last login.
Exit	Exits the Dynamic Data Masking session.
Add Domain	Creates a domain in the Management Console tree. Add Domain is available when a domain node is selected.
Add Database	Defines the connection properties for an additional database in the Management Console tree. Add Database is available when a domain node is selected.
Add DDM Services	Adds a Dynamic Data Masking service to the Management Console tree. Add DDM Services is available when the server node is selected.
Add Rule Set	Adds a security rule set to the Management Console tree. Add Rule Set is available when a domain node is selected.
Edit	Opens a window to edit domain and rule set names, define service listener ports, and edit connection information for databases and the Dynamic Data Masking Server.
Security Rule Set	Opens a security rule set. Security Rule Set is available when a security rule set is selected.
Connection Rules	Opens a connection rule set. Connection Rules is available when a Dynamic Data Masking service is selected.
Authorization	Opens a window to set and edit permissions for the selected node. Authorization is available when a database, domain, or security rule set node is selected.
Cut	Copies and deletes the selected Management Console tree node. You can cut server, database, domain, and security rule set nodes. You cannot cut service nodes or the root domain node.
Copy	Copies the selected Management Console tree node. You can copy database, domain, and security rule set nodes. You cannot copy service nodes, server nodes, or the root domain node.

Menu Item	Action
Paste	Pastes the cut or copied Management Console tree node. You can paste on domain nodes.
Remove	Removes a node from the Management Console tree. **Note:** You cannot remove the Dynamic Data Masking Server or the Management Console root domain.
Start Service	Starts a Dynamic Data Masking service. The Start Service option is available when a Dynamic Data Masking service is selected.
Stop Service	Stops a Dynamic Data Masking service. The Stop Service option is available when a Dynamic Data Masking service is selected.
Add Logger	Adds a custom logger to the Management Console tree. Add Logger is available when the Loggers node is selected.
Add Appender	Adds an appender to the Management Console tree. Add Appender is available when a logger node is selected.
Support	Creates an encrypted .zip archive of Dynamic Data Masking logs. You can send the log archive to Informatica Global Customer Support to troubleshoot issues with Dynamic Data Masking.
Manage Licenses	Allows you to select a new license file. Use the Manage Licenses option if the Dynamic Data Masking license file has expired. The Manage Licenses option is available when the Dynamic Data Masking Server node is selected.
Sort by Name	Sorts child nodes and nested child nodes in alphabetical order by the name of the node. Sort by Name is available when a node with child nodes is selected.
Sort by Owner	Sorts child nodes and nested child nodes in alphabetical order by the login name of the user that created the nodes. Sort by Owner is available when a node with child nodes is selected.
Sort by Type	Sorts child nodes and nested child nodes in alphabetical order by the type of node. Sort by Type is available when a node with child nodes is selected.

Management Console Tree

The Management Console tree is a navigation tree organized by nodes. When the Management Console is not connected to a Dynamic Data Masking Server, it shows a default domain node. All actions are disabled, except Login, Exit, and About. After successful login to a Dynamic Data Masking Server, the Management Console shows a tree with the Dynamic Data Masking Server node that it is connected to.

The Management Console tree can contain domain, database, server, service, logger, appender, and rule set nodes. Tree nodes are arranged hierarchically.

On the Management Console, the relationship between the Dynamic Data Masking Server and a database is based on the domain organization. The Dynamic Data Masking Server will connect to databases that are in the same domain or a sub domain of the Dynamic Data Masking Server. The Dynamic Data Masking Server will not connect to any database that is outside the domain that contains the Dynamic Data Masking Server.

Logging In to the Management Console

You can access to 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.

Domain Management

A domain is a virtual node in the Management Console tree that you use to group other nodes. The Management Console contains a default root domain. You can use domains to create a visual representation of the structure of the databases within an organization.

You can create an unlimited number of domains in the Management Console tree. A domain can contain other domains, databases, and server child nodes. You can set user permissions on domain nodes.

You can add, edit, cut, copy, paste, and remove a domain. You cannot remove the root domain. Drag a domain up or down in the Management Console tree to change the position of the domain within the tree.

Security Rule Set

A security rule set is a tree node that contains references to one or more security rules. You must create security rule sets with masking, rewrite, and blocking actions to secure data.

You can add a security rule set to a domain node in the Management Console tree.

The Management Console tree can contain an unlimited number of rule set nodes. Rule sets do not have child nodes. You can add, edit, move, and remove rule set nodes. You can set user permissions on security rule set nodes.

CHAPTER 2

Rules

This chapter includes the following topics:

- Rules Overview, 18
- Rule Components, 18
- Rule Trees, 19
- Security Rule Sets, 20
- Rule Folders, 21
- Rule Management, 23
- Rule Export and Import, 24
- Rule Example, 24

Rules Overview

A rule contains the conditions and actions that the Rule Engine uses to process a request. Connection rules process application connection requests. Security rules process SQL statements. You create and define rules to manage the SQL requests that the client or application sends to the target database.

A rule defines connection criteria and masking techniques. The Rule Engine uses connection criteria to forward requests and masking techniques to mask data. Rules can be connection rules or security rules, placed in a rule tree. The organization of the rule tree determines the order in which the Rule Engine applies the rules. You can create and edit rules within the Management Console.

Rule Components

A rule consists of a matcher, an action, and a processing action. Each rule component defines how the Rule Engine identifies and processes a request to the database.

The Rule Engine applies a connection rule to incoming connection requests and applies security rules to SQL statements. A connection rule defines how the Dynamic Data Masking service establishes a connection with the application. A security rule defines the conditions and masking rules that the Rule Engine applies to the SQL statement request. The Rule Engine applies a security rule if you configure a connection rule to apply a rule set.

A rule consists of the following components:

Matcher

Defines the criteria that the Rule Engine uses to identify a match.

Action

Defines the action that the Rule Engine applies to the request.

Processing action

Defines the action that the Rule Engine applies to the request after the Rule Engine applies the rule. The processing action manages how the Rule Engine processes the request through the rule tree. A processing action can specify that the Rule Engine does not process further rules in the rule tree or that the Rule Engine continues to evaluate other rules.

Rule Trees

A rule tree represents the organizational structure of rules and rule folders. The position that you assign a rule or rule folder within the rule tree determines the order in which the Rule Engine processes the rule. You build conditional relationships between rules through the rule tree.

A rule tree can be a connection rule tree or a security rule tree. Each rule tree uses a system of folders and rules to determine the hierarchical order for rule processing.

When the Dynamic Data Masking Server receives a connection request, the Rule Engine parses the request through the connection rule tree. If the connection rule assigns a security rule set, then the Rule Engine parses the SQL request through the security rule tree. The security rule set defines the security techniques that the Rule Engine applies to rewrite the SQL statement.

Rule Tree Components

Connection rules and security rules work together to define when and how data is masked.

Use the following components to identify and manage application requests:

Rule

The conditions and actions that you want to apply to a request. A rule can be a connection rule or a security rule. You can create an individual rule or create a rule as part of a rule folder.

A rule consists of a matcher, action, and processing action.

Rule folder

A rule that uses the Folder rule action. You can use a rule folder to group conditional rules. The Rule Engine processes the contents of a rule folder hierarchically.

A connection rule folder contains connection rules. A security rule folder contains security rules.

Connection rule

A rule that defines the criteria that the Rule Engine uses to identify the target database for the request. A connection rule consists of a matcher and an action that you define to identify and route a connection request from an application.

Connection rule tree

The connection rule tree defines the order in which the Rule Engine processes connection rules. The connection rule tree contains all the connection rules that you define for the target databases. The Rule

Engine processes the first rule or rule folder in the connection rule tree and stops at the end of the rule tree or when there is a stop processing action.

Security rule

A rule that defines the criteria that the Rule Engine uses to parse and alter the SQL statement request. A security rule consists of a matcher and action that you define to identify and mask a SQL request.

Security rule set

A security rule set is a container for security rules. You use rule folders to organize and nest rules within the rule set. A security rule set can contain multiple rule folders. The Rule Engine processes the SQL statement through the rule set until the Rule Engine encounters a stop processing action.

Security rule tree

Each security rule set has an individual security rule tree. The security rule tree defines the order in which the Rule Engine processes security rules. The security rule tree contains the security rules for a particular security rule set. The Rule Engine processes the first rule or rule folder in the security rule tree and stops when there is a stop processing action.

Rule Tree Efficiency

Organize a rule tree to minimize the impact of Dynamic Data Masking on the target database. An efficient rule tree reduces the number of rules that the Rule Engine must apply and defines an organizational structure that leverages the best use of the rule components.

Use the following guidelines to create an efficient rule tree:

- Use text matchers for all incoming connections to ensure a faster response time. Text matchers take approximately one microsecond per request.
- Minimize the use of syntax matchers. Syntax matchers parse SQL request that Dynamic Data Masking listener port receives. A syntax matcher can take over 100 microseconds to process.
- Configure the matcher to keep the matcher result valid for all concurrent SQL requests that are open in a session. The default time that the Rule Engine stores matcher results is 3,600 seconds.

Security Rule Sets

A rule set is a collection of security rules. Create a rule set to group security rules that you want the Rule Engine to apply to SQL request statements. Assign the rule set to a connection rule. The Rule Engine applies the rule set if the connection request matches the conditions that the connection rule defines.

A security rule set can contain a combination of rules that perform different tasks. You can add security rules that mask data, block requests, rewrite the SELECT statement, or replace a part of the SELECT statement. To organize rules within a rule set, you use rule folders to group security rules that apply the same action. For example, group rules that use the mask action in one rule folder, and group rules that use the block action in a different rule folder.

Creating a Security Rule Set

Create a security rule set to group security rules that share a relational link.

1. In the Management Console, click on a domain node in the rule tree.

2. Click **Tree** > **Add Rule Set**.

   The **Add Rule Set** window appears.

3. Enter a name for the security rule set and click **OK**.

   The security rule set appears inside the rule tree in the Management Console.

You can add security rules to the rule set.

Rule Folders

A rule folder is a container for rules that share conditional relationships. You can rename, reconfigure, move, cut, copy, paste, delete, disable, and enable a rule folder. Create rule folders to organize related rules and define the order in which the Rule Engine applies them.

A rule folder can contain connection rules or security rules. A connection rule folder groups rules that you apply to connection requests. A security rule folder contains rules that you apply to SQL request statements.

To create a rule folder, specify the folder action for the rule. Within the rule folder, create relational rules that help the Rule Engine identify, match, and rewrite requests. Before you can add rules to a folder, you must outline the relationships between the rules that you want the folder to contain.

You can group rules that share the same purpose. For example, group masking rules together in a rule folder and group access control rules in another rule folder.

Note: When you create a rule folder, you must set the processing action to **Continue**. When the Rule engine encounters a continue processing action, the Rule Engine processes the request through the rules that the folder contains.

Creating a Connection Rule Folder

You can create a connection rule folder to group connection rules that share a conditional relationship. The preceding connection rule defines whether the Rule Engine applies the subsequent connection rule.

1. In the Management Console, click on the Dynamic Data Masking service in the rule tree.
2. Select **Tree** > **Connection Rules**.

   The **Rule Editor** opens.

3. Select the location within the rule tree where you want to add the folder. If you want to add a top-level rule folder, select **Connection Rules** at the top of the rule tree.
4. Select **Action** > **Append Rule**.

   The **Append Rule** window appears.

5. Enter a rule name in the **Rule Name** box.
6. Select and define a matcher.
7. Select the folder action.
8. Select the continue processing action.
9. Click **OK**.

   The rule appears as a folder in the rule tree.

10. Select **File** > **Update Rules**.

    The rule tree is saved.

You can add connection rules after you create the rule folder.

Creating a Security Rule Folder

Create a security rule folder to group security rules. Add a security rule folder to a rule set.

1. In the Management Console, select a security rule set.
2. Select **Tree** > **Security Rule Set**.

   The **Rule Editor** opens.
3. Select the location in the rule tree where you want to add the rule folder. If you want to add a top-level rule folder, select the name of the rule set at the top of the rule tree.
4. Select **Action** > **Append Rule**.

   The **Append Rule** window appears.
5. Enter a rule name in the **Rule Name** box.
6. Select and define a matcher.
7. Select the folder action.
8. Select the continue processing action.
9. Click **OK**.

   The rule appears as a folder in the rule tree.
10. Select **File** > **Update Rules**.

    The rule tree is updated.

You can add security rules after you create the rule folder.

Adding a Rule to a Rule Folder

You can add rules to a rule folder.

1. In the Management Console, open the **Rule Editor** for connection rules or security rules.
2. In the **Rule Editor**, select the rule folder where you want to add the rule.
3. Select **Action** > **Append Rule**.

   The **Append Rule** window appears.
4. Enter a name for the rule.
5. Enter a description for the rule.
6. Select and define a matcher.
7. Select and define a rule action.
8. Select and define the processing action.
9. Click **OK**.

   The rule appears in the rule folder.
10. Select **File** > **Update Rules**.

    The rule tree is updated.

Rule Management

You can rename, reconfigure, move, cut, copy, paste, delete, disable, and enable a rule. When you make changes to a rule, you must update the rule tree for the change to take effect.

When you create, edit, or add a connection rule, you must restart the current session with the application before the Rule Engine can apply the connection rule. Updates to a security rule take effect immediately. You do not need to start a new session for the Rule Engine to apply a security rule.

You can move a rule to change the position of the rule in the rule tree. You can move a rule folder to change the position of the folder in the rule tree. You cannot move a rule into or out of a folder. Move a rule to change the order in which the Rule Engine applies the rule.

You can enable or disable a rule or rule set. By default, the rule or rule set is enabled. When you enable a rule or rule set, the Rule Engine applies the rule or rule set to the request. When you disable a rule, the Rule Engine skips the rule or rule set and applies the next rule or rule set in the rule tree.

You can delete a rule or rule folder from the rule tree. When you delete a rule folder, you delete the content that the rule folder contains. You cannot undo the changes after you delete a rule or rule folder from the rule tree.

Editing a Rule

You can edit the matcher, action, processing action, and name for any rule.

1. Open the **Rule Editor** for connection rules or security rules.
2. Click on a rule in the rule tree.
3. Click **Edit**.

   The **Edit** window appears.
4. Make the changes that you want to the rule components.
5. Click **OK**.
6. Select **File** > **Update Rules**.

   The rule tree is updated.

Enabling and Disabling a Rule

You can enable or disable a rule. By default, the rule is enabled. When you enable a rule, the Rule Engine applies the rule to the request. When you disable a rule, the Rule Engine skips the rule and applies the next rule in the rule tree.

1. Open the **Rule Editor** to view security rules or connection rules.
2. Click the rule that you want to disable or enable.
3. Click the **Disable** check box from the **Rule Editor** tool bar.
4. Update the rules.

Deleting a Rule or Rule Folder

You can delete a rule or rule folder from the rule tree. When you delete a rule folder, you delete the rules that the rule folder contains. You cannot undo the changes after you delete a rule or rule folder from the rule tree.

1. Open the Rule Editor for security rules or connection rules.
2. Click a rule or rule folder from the rule tree.
3. Click **Delete**.

   The rule or rule folder is removed from the rule tree.
4. Update the rule tree.

Rule Export and Import

You can import and export rule folders and rule sets to back up and reuse data. If you have complex rules, you might want to export the rules so that you have a copy of the data. Additionally, if you have rules that you want to reuse, you can use the import and export utilities to create a copy of the rule, rule folder, or security rule set in the rule tree.

You can export rule folders and security rule sets. You cannot export a single connection rule or security rule. If you want to export a single rule, you must add the rule to a rule folder.

The exported XML file defines the configurations for rules that the rule folders and rule sets contain. When you import the XML file, you import the rules that the XML file defines.

Rule Example

Employees are permitted to access masked Social Security numbers of customers during business hours. You want to create a rule that blocks all access to Social Security numbers between 5:00 p.m. and 8:00 a.m.

You can create a rule set that contains rules that block or mask data based on the time that the applications send the request. Requests that do not fall within that time frame receive masked data.

Masking Rule

In the masking rule folder, create the masking rule.

To create the masking rule, assign the text matcher to identify SELECT lists that contain the Social Security column. If the SELECT list matches the security rule criteria, the Rule Engine applies the mask action to the SELECT list. The mask action hides all the digits of the Social Security number. Enable the keep matcher result option to store the matched result. Then, apply the Stop if Applied processing action. If the Rule Engine applies the action, the Rule Engine does not continue to process the SQL statement through the rule tree and forwards the rewritten request to the database. Otherwise, the Rule Engine processes the SQL statement through the next rule.

Blocking Rule

In the blocking rule folder, create the blocking rule.

The blocking rule identifies requests that applications send between 5:00 p.m. and 8:00 a.m. If the SELECT list matches the security rule criteria, the Rule Engine applies a block security action to the request. The user receives a message indicating that access is denied. Apply the Stop if Matched processing action so that the Rule Engine does not continue to process the request. The Dynamic Data Masking service sends the response to the application.

As a result of the security rule set, users can view masked Social Security numbers during business hours, but cannot see the data after business hours.

CHAPTER 3

Connection Rules

This chapter includes the following topics:

- Connection Rules Overview, 26
- Connection Rule Parameters, 26
- Connection Rule Matchers, 27
- Connection Rule Actions, 31
- Processing Actions, 34
- Creating a Connection Rule, 34
- Connection Rule Export and Import, 34

Connection Rules Overview

A connection rule defines the connection criteria that the Rule Engine uses to identify a connection and the target database.

A connection rule uses a matcher and a rule action to identify and route a connection. The matcher defines the criteria that the Rule Engine uses to identify a match. The rule action determines how the Rule Engine processes the matched connection.

When the Dynamic Data Masking service receives a connection request, the Rule Engine applies connection rules to the connection. If the connection request matches the criteria defined by the matcher, the Rule Engine applies the rule action to the connection. After the Rule Engine applies the rule action, the Rule Engine applies processing action. The processing action defines the next action that the Rule Engine applies to the request.

Connection Rule Parameters

Connection rule parameters are options that you use to configure a connection rule. Use connection rule parameters to define the matcher, rule action, and processing action that the Rule Engine uses to identify, match, and forward connections.

The following table describes the connection rule parameters:

Parameter	Description
Rule Name	The logical name of the rule that appears in the rule tree.
Matcher	Defines the criteria that the Rule Engine uses to identify a match. The matcher defines the matching method used for incoming connections.
Rule Action	Defines the action that the Rule Engine applies to incoming connections. The action determines where the Rule Engine routes the connection.
Processing Action	Determines how the Rule Engine processes the connection after the Rule Engine applies the connection rule.

Connection Rule Matchers

A connection rule matcher defines the criteria that the Rule Engine uses to identify incoming SQL requests that receive the connection rule action.

For example, to identify requests from a specific application, such as Toad, create a connection rule that uses the Client/Application Information matcher. Add Toad to the matcher Include List, and the Rule Engine returns a match. When a match occurs, the Rule Engine applies the rule action.

You can use the following connection rule matchers:

All Incoming Connections

The Rule Engine applies the rule action to all incoming connections.

Check Database Connection

The Rule Engine returns a match based on the availability of the database connection.

Check Database DSN

The Rule Engine returns a match based on the data source name of the database.

Check Database URL

The Rule Engine returns a match based on the URL that the client application gives for the driver.

Check Property

The Rule Engine returns a match based on a DSN property of the driver.

Check Server Name

The Rule Engine returns a match based on the name of the Dynamic Data Masking Server that sends the SQL request.

Client Information

The Rule Engine returns a match based on the client that sends the SQL request. The Client Information is available for the DDM for JDBC and DDM for ODBC services.

Client/Application Information

The Rule Engine returns a match based on the client or application that sends the SQL request. The Client/Application Information matcher is not available for the DDM for JDBC and DDM for ODBC services.

Current Target Database

The Rule Engine returns a match based on the target database of the SQL request.

Incoming DDM Listener Port

The Rule Engine returns a match based on the listener port where Dynamic Data Masking receives the request.

All Incoming Connections Matcher

The All Incoming Connections matcher applies the matching action to all incoming connections, regardless of target database, listener port, or application information. The All Incoming Connections matcher does not specify connection criteria and does not return a match. You can use the All Incoming Connections matcher to enforce the connection rule on all incoming connection requests to the database.

Check Database Connection Matcher

The Check Database Connection matcher checks whether Dynamic Data Masking has a connection to the database. Use the Check Database Connection matcher to send requests to the database based on whether a connection exists.

The Check Database Connection matcher uses the following parameter:

Database

The name of the database in the Management Console tree.

Database High Availability Example

You have a primary database and a secondary database. You want Dynamic Data Masking to send requests to the primary database if the database is available. If the primary database is unavailable, you want Dynamic Data Masking to send requests to the secondary database.

You create two connection rules. The first connection rule uses the Check Database Connection matcher to determine whether a connection to the primary database exists. The rule uses the Switch to Database rule action and Stop if Matched processing action to direct the request to the primary database if the connection is available.

If the first rule does not return a match, the Dynamic Data Masking sends the request to the second connection rule. The second rule uses the Check Database Connection matcher to determine whether a connection to the secondary database exists. If the second rule returns a match, the Switch to Database rule action instructs Dynamic Data Masking to the request to the secondary database.

Check Database DSN Matcher

The Check Database DSN matcher identifies a request based on the data source name of the database.

The Check Database DSN matcher is available for the DDM for ODBC service.

You can use the Check Database DSN matcher to identify and direct requests to the database. For example, if the DDM for DSN service serves multiple databases, you can create a connection rule that uses the Check Database DSN matcher to identify the database DSN and the Switch to Database rule action to direct the request to the correct database.

Configure the following parameter for the Check Database DSN matcher:

Value

The logical data source name that the client uses to connect to the database.

Check Database URL Matcher

The Check Database URL matcher identifies a request based on the URL that the client application gives for the driver.

The Check Database URL matcher is available for the DDM for JDBC service.

You can use the Check Database URL matcher to identify and direct requests to the database. For example, if the DDM for JDBC service serves multiple databases, you can create a connection rule that uses the Check Database URL matcher to identify the connection URL and the Switch to Database rule action to direct the request to the correct database.

Configure the following parameter for the Check Database URL matcher:

Value

The complete URL for the client application.

Check Property Matcher

The Check Property Matcher identifies a request based on a DSN property of the driver.

The Check Property matcher is available for the DDM for JDBC and DDM for ODBC services.

You can use the Check property matcher to identify and direct requests to the database. For example, if you have two versions of Oracle, you can create a connection rule that uses the Check Property matcher to identify the request based on the ODBC DSN version property and the Switch to Database rule action to direct the request to the correct database.

Note: For security, the database password is not sent to the Dynamic Data Masking Server. If the password is available, Dynamic Data Masking filters the password. Therefore, you cannot use the password property with the Check Property matcher.

Configure the following parameters for the Check Property matcher:

Property Name

The name of the DSN property.

Value

The value of the DSN property.

Check Server Name Matcher

The Check Server Name matcher creates a match based on the name of the Dynamic Data Masking Server that receives the request.

You can create connection rules and security rules for multiple Dynamic Data Masking Servers in a single Management Console. Use the Check Server Name matcher to apply the connection rule action to the request if the request is to the Dynamic Data Masking Server that you specify in the Server Name parameter.

The Check Server Name matcher uses the following parameter:

Server Name

The name of the Dynamic Data Masking Server that receives the request.

Client Information Matcher

The Client Information matcher determines the clients that can access the database.

The Client Information matcher is available for the DDM for JDBC and DDM for ODBC services.

The Client Information matcher provides an include and exclude list that you use to manage the clients that have access to the database. The include list specifies the clients that you want to restrict access to information within the database. The exclude list specifies the clients that you want to allow access to the database.

Enter the host name of the client where the JDBC or ODBC agent runs. The agent runs on the same machine as the application. You can find the host name in the `Host` property value of the Dynamic Data Masking rule.log file.

When the Rule Engine applies the Client Information matcher, the Rule Engine parses the exclude list first. Clients that are on the exclude list bypass Dynamic Data Masking and access the database directly. Next, the Rule Engine parses the include list. Clients that are on the include list can access obfuscated data.

Note: If you do not specify any information in the include list, all hosts can access unmasked data.

Configure the following parameters for the Client Information matcher:

Include List

Defines the client host name or IP address that you want to add to the include list. Add clients that you want to deny database access to.

Exclude List

Defines the client host name or IP address that you want to add to the exclude list. Add clients that you want to allow to access the database.

Client/Application Information Matcher

The Client/Application Information matcher determines the users and applications that can access the database. Use the application information, such as the user name, host information, and program name, to define the connection criteria for the Client/Application Information matcher.

The Client/Application Information matcher provides include and exclude lists that you use to manage the users and applications that have access to the database. The include list specifies the applications and users that you want to restrict access to information within the database. The exclude list specifies the applications and users that you want to allow access to the database. For example, you can use the Client/Application Information matcher to mask requests from development applications, but allow database administrators to access the application data.

When the Rule Engine applies the Client/Application Information matcher, the Rule Engine parses the exclude list first. Users and applications that are on the exclude list bypass Dynamic Data Masking and access the database directly. Next, the Rule Engine parses the include list. Users and applications that are on the include list can access obfuscated data.

Note: If you do not specify any information in the include list, the Rule Engine includes all application programs.

The Rule Engine uses regular expressions to perform application identification on the full name. For programs, you must use `.*` before the program name to match the full program path. For example, use `.*sqlplus.exe` for `application=C:\oracle\app\product\11.1.0\db_1\bin\sqlplus.exe`.

The following table describes the parameters for Client/Application Information matcher:

Parameter	Description
OS user	Operating system user name.
Host name	Client/application host name.
Program name	Name of the file to run the application.
Include list	Defines the program file names, hosts, and application users that you want to add to the include list.
Exclude list	Defines the program file names, hosts, and application users that you want to add to the exclude list.

The Client/Application Information matcher is not available for the DDM for JDBC or DDM for ODBC services.

Current Target Database Matcher

The Current Target Database matcher searches incoming requests for the target database value. Use the Current Target Database matcher to identify requests to the target database that you define in the Database Editor.

The following table describes the Current Target Database parameter:

Parameter	Description
Database	Defines the target database name.

Incoming DDM Listener Port Matcher

The Incoming DDM Listener Port matcher searches for the Dynamic Data Masking listener port in the connection request.

The following table describes the parameters for the Incoming DDM Listener Port matcher:

Parameter	Description
Incoming port	Defines the listener port.

Connection Rule Actions

The connection rule action defines the process that the Dynamic Data Masking service applies to the connection after the Rule Engine identifies a match.

Apply connection rule actions to manage the connection routing. An action can determine whether the Rule Engine blocks the connection request, applies a rule set to the connection, or forwards the connection to the database.

The following table describes the connection rule actions:

Connection Rule Action	Description
Nothing	Applies the connection processing action, but does not apply an action.
Folder	Creates a folder and processes the connection through the contents of the folder until a processing action stops the Rule Engine. Use the folder action to nest rules that are conditional upon each other. **Note:** Use the continue processing action with the folder connection rule action to apply the rules defined within the folder.
Switch to database	Forwards connections to the specified database.
Direct	Connects applications and clients directly to the target database. When the Rule Engine applies the direct action, the connection bypasses the Dynamic Data Masking service. You can only use the Direct action with the DDM for Oracle service.
Use rule set	Applies the specified security rule set to the SQL statement request.
Refuse	Defines the access that an application has to the database. The application receives an error that indicates that the database refuses the connection.
Transparent Archive	Routes the request to the Data Vault. Define the following parameters: - DDM Database Name. The Dynamic Data Masking database name for the Data Vault that you define in the Management Console. - DBA User. The Data Archive user name. - DBA Password. The password for the Data Archive user.
Load Control	Sets the priority levels for primary and secondary Dynamic Data Masking Servers when you configure Dynamic Data Masking Server high availability for DB2. Define the following parameters: - Host. The host names of the Dynamic Data Masking Servers. - Port. The listener port numbers of the Dynamic Data Masking Servers. - Priority. The priority of the Dynamic Data Masking Servers.

Load Control Action

The Load Control action sets the priority levels for primary and secondary Dynamic Data Masking Servers when you configure high availability for DB2.

You can configure the priority levels of two or more Dynamic Data Masking Servers in the Load Control action. The value of the Priority property corresponds to the frequency that the client sends the request through the Dynamic Data Masking Server. For example, if you have the priority of DDMServer1 set to one and the priority of DDMServer2 set to 19, DDMServer1 receives 5 percent of the requests from the client and DDMServer2 receives 95 percent of the requests.

Enter zero (0) as the priority level of a server to indicate that requests must only go through the server if no other server available. For example, if you have the priority of DDMServer1 set to zero and the priority of DDMServer2 set to one, DDMServer2 receives 100 percent of the requests from the client unless DDMServer2 is unavailable.

Configure the following properties for the Load Control action:

Host

The host names of the Dynamic Data Masking Servers.

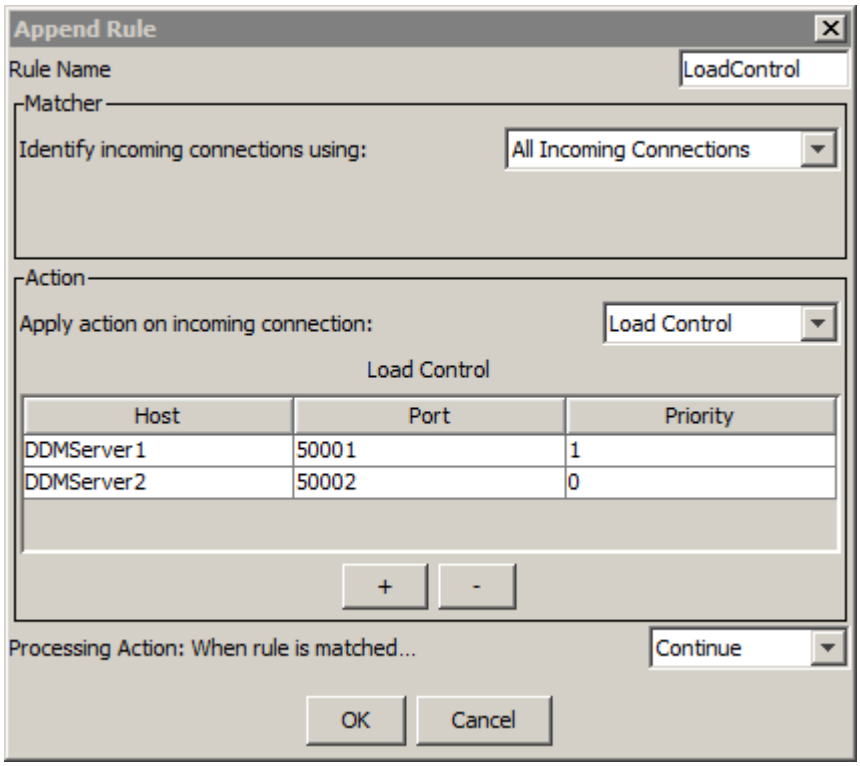
Port

The listener port numbers of the Dynamic Data Masking Servers.

Priority

The priority level of the Dynamic Data Masking Servers.

The following image shows the Load Control action configured for a primary Dynamic Data Masking Server and a failover Dynamic Data Masking Server:



Processing Actions

The processing action defines what the Dynamic Data Masking service does after the rule action has been performed. The processing action determines whether the Rule Engine stops applying rules or continues to the next rule in the rule tree.

The following table describes the processing actions:

Connection Rule Processing Action	Description
Continue	The Rule Engine continues to the next rule in the rule tree. **Note:** Assign the Continue processing action to rule folders so that the Rule Engine applies the rules within the rule folder.
Stop if Applied	If the rule is applied, the Rule Engine does not continue to the next rule in the rule tree.
Stop if Matched	If the rule is matched, the Rule Engine does not process the connection rule action.

Creating a Connection Rule

Create a connection rule to manage how the Dynamic Data Masking service forwards incoming connections. Define the connection criteria and processing action for a connection rule. Add a single connection rule to the rule tree or nest a connection rule within a rule folder.

1. In the Management Console, select the Dynamic Data Masking service that you want to add the connection rule to, and click **Tree** > **Connection Rules**.

   The **Rule Editor** window appears.
2. Select **Action** > **Append Rule**.

   The **Append Rule** window appears.
3. Enter a rule name in the **Rule Name** box.
4. Select and define a matcher.
5. Select and define an action.
6. Select a processing action.
7. Click **OK**.

   The connection rule appears in the rule tree.
8. Select **File** > **Update Rules**.

   The connection rule is saved.

Connection Rule Export and Import

Export connection rules within security rule folders. You can save connection rule configurations in an XML file. You can import XML files to use saved connection rule configurations.

Connection Rule Export

Export a connection rule folder to an XML file to back up connection rule definitions. You can make a copy of a rule folder for complex rule configurations that you want to reuse. You can use the export utility to make a copy of a rule folder and import the copy to the connection rule tree.

An exported file contains the connection rules and the connection rule components, such as the connection rule matchers, connection rule actions, and connection rule processing actions. You must export connection rules inside a connection rule folder. To export a single connection rule, create a folder that contains only the connection rule.

Exporting Connection Rules

Export connection rules inside a connection rule folder through the Management Console.

1. In the Management Console, click **Tree** > **Connection Rules**.

   The **Rule Editor** appears.
2. Click the connection rule folder that you want to export.
3. Click **Action** > **Export**.

   The **Export** window appears.
4. Browse to a location to save the export file.
5. Enter a name for the export file.
6. Click **Export.**

Connection Rule Import

When you import a connection rule file, you import the connection rules and the connection rule components that the file defines. The connection rule components include matchers, actions, and processing actions. You can import connection rules into an empty rule folder or to the top of the connection rule tree. Import the connection rules, then update the rule tree.

Before you import rules, create an empty rule folder in the connection rule tree. If you import into a rule folder that is not empty, the imported rules override the rules in the rule folder.

If you choose to import the connection rules to the top of the connection rule tree, the contents of the imported file overrides the existing rules in the rule tree. The existing rules are deleted.

Importing Connection Rules

Import connection rules through the Management Console.

1. In the Management Console, click **Tree** > **Connection Rules**.

   The **Rule Editor** appears.
2. Click an empty rule folder or the top of the rule tree. Rules that you import appear in the location that you choose.
3. Click **Action** > **Import**.

   The **Import** window appears.
4. Browse to the location of the XML file that defines the connection rules that you want to import and double-click the file.

   The rule tree displays the connection rules.
5. Update the rule tree to save the changes.

CHAPTER 4

Security Rules

This chapter includes the following topics:

- Security Rules Overview , 36
- Security Rule Parameters, 37
- Security Rule Matchers , 37
- Security Rule Actions, 56
- Processing Actions, 68
- Creating a Security Rule, 68
- Security Rule Export and Import, 69
- Security Rule Set Export and Import, 70

Security Rules Overview

A security rule defines the criteria that the Rule Engine uses to parse and rewrite an SQL request. A security rule is part of a security rule set. You can create a security rule in a rule folder or as a rule in a rule tree.

Security rules specify the technique that the Rule Engine uses to mask data. Security rules consist of a matcher, a rule action, and a processing action. Use security rules to mask data in a specific row or to mask an entire column. For example, you can create a security rule that rewrites SQL requests that reference the Social Security column from the Employee table.

Connection rules define the security rules that the Rule Engine applies. Use the connection properties to define security rules for Microsoft SQL Server.

Security Rule Parameters

The Rule Engine applies security rules to incoming SQL statements to identify a match to the criteria that you define. If a match occurs, the Rule Engine applies the security action to the SQL statement.

The following table describes the security rule parameters:

Parameter	Description
Rule Name	Name of the rule that appears in the rule tree.
Description	Optional rule description.
Matcher	Defines the portion of the SQL request that the Rule Engine must match to apply an action. Returns a match when the Rule Engine identifies a match.
Keep Matcher Result	Specifies that the Rule Engine must use the current matcher result for subsequent incoming requests for the amount of time that you specify in the **Try to match every ___ seconds per session** parameter.
Try to match every ___ seconds per session.	Specifies the amount of time that you want the Rule Engine to use the current matcher result for incoming requests. You must enable the Keep Matcher Result parameter to use the **Try to match every ___ seconds per session** parameter. Default is 3600 seconds.
Rule Action	Defines the action the Rule Engine applies to an SQL statement.
Processing Action	Defines how the Rule Engine processes the connection after the Rule Engine applies the security rule action.
Log When Rule is Applied	Creates an entry in the rule.log file if the Rule Engine applies the rule action to the request.

Security Rule Matchers

Security rule matchers define the criteria that the Rule Engine uses to identify incoming SQL requests that receive the rule action.

The security rule matcher that you choose depends on the method you want the Rule Engine to use to identify SQL statements. For example, if you want to restrict access to a customer table, use the From Clause Object rule matcher with the Block Statement processing action. You can use the Text rule matcher to identify a specific text string within an SQL request or the symbol matcher to match a variable value, such as the session or client information. If you want to apply the security rule to all incoming requests, use the any matcher.

Any Matcher

The Any matcher applies the rule action to all incoming SQL statements.

Use the Any matcher for rules that apply the folder action, so that the Rule Engine processes the request through the contents of the folder.

From Clause Object Matcher

The From Clause Object matcher identifies SELECT statements that query an object or an alias in the FROM clause.

Use the From Clause Object matcher to specify a view and not the underlined table. Dynamic Data Masking matches based on the SQL parsing of the SELECT request and does not parse the execution plan of the SELECT request.

Java Matcher

The Java matcher runs a Java class that returns a boolean value. If the returned value is true, the matcher returns a match. If the returned value is false, the matcher does not return a match.

The Java class must contain a public static method with the following signature:

```
public static boolean match(RuleContext ctx)
```

You must enter the correct class in the Class Path field of the Java matcher. The Rule Engine reads every rule in the rule set and returns an error if the class path is incorrect, even if the rule is disabled.

When an error occurs with the Java class, the Rule Engine does not apply the rule. Instead, the Rule Engine continues to the next rule. The Rule Engine logs a detailed error message in the following file:

```
<Dynamic Data Masking installation>\log\DDMError.txt
```

The Java matcher has the following parameters:

Class Path

Defines a list of jars that contain the Java matcher class and necessary libraries. Use semicolons to separate multiple jar files.

Class Name

Defines the fully qualified class name. The class may be built within a named package or an unnamed package. The class must include a method named `match` with the following signature: `public static boolean match(RuleContext ctx)`.

The `RuleContext` object contains data from the Rule Engine. The `RuleContext` object has one method, `public String getSymbol(String symbol);`, that returns the symbol value from the symbol name. Use this method to retrieve symbol values defined by a Symbol rule action. Specify the name of the symbol as an argument to the method.

You can also use the following predefined symbols:

RuleContext.USERNAME

Retrieves the Oracle user name.

RuleContext.OS_USER

Retrieves the operating system user on the client machine.

RuleContext.CLIENT_HOST

Retrieves the name of the client machine.

RuleContext.PROGRAM

Retrieves the program name on the client machine.

RuleContext.STATEMENT

Retrieves the statement being processed.

Note: The Rule Engine runs the Java class with the JVM that Dynamic Data Masking uses. You can compile the Java matcher class with a JDK of the same major version or lower version of the JVM that Dynamic Data Masking uses.

Java Class Example

The following text is an example of a Java class that you can use as a Java matcher:

```
import com.activebase.rule.*;

public class ExampleJM {

    public static boolean match(RuleContext ctx) {
        String user = ctx.getSymbol("APPLICATION_USERNAME");
        if (user==null || "".equals(user.trim())) {
            user = ctx.getSymbol(RuleContext.OS_USER);
        }
        if (!"FRED".equals(user.toUpperCase())) {
            return true;
        }
        return false;
    }
}
```

The Java matcher runs the match method. The method uses the `getSymbol` invocation of the `RuleContext` object to return a value for the user based on the value in the user-defined APPLICATION_USERNAME symbol. If the value is null, the matcher obtains the operating system user name from the appropriate Dynamic Data Masking symbol name from RuleContext.OS_USER.

The method compares the symbol value to `FRED` and returns a match if they are different. You can use the Mask rule action to apply a masking function to requests that do not come from FRED. Select the Stop if Applied processing action because the only user that has authorization to see unmasked data is FRED. If the user is not FRED, Dynamic Data Masking applies the masking action.

To retrieve the user name, create a security rule in the rule set with the Java matcher rule, higher in the rule sequence in the rule tree than the Java matcher rule. In the security rule, select the Text matcher and enter a regular expression to retrieve the user name from the request statement. Select the Define Symbol rule action and assign the user name to the APPLICATION_USERNAME symbol.

The incoming SELECT statement has a tag at the end, which is a two consecutive hyphens (--) followed by a string. The string is the user name that the Java matcher uses for comparison.

Use the following regular expression that treats the consecutive hyphens as a place holder and captures the text after the place holder:

```
.*--(.*)
```

The `(.*)` component of the regular expression captures the text and the Define Symbol rule action in the security rule references the text with the following regular expression markup:

```
\(1)
```

The Define Symbol action defines the symbol name as APPLICATION_USERNAME and the value as `\(1)` from the regular expression.

For example, when the user Scott views the customer information screen, the application submits the following query to the database:

```
select customer_name from customer --Scott
```

The Text matcher captures the text `Scott` after the consecutive hyphens. The Define Symbol action sets the APPLICATION_USERNAME symbol to the value `Scott`. The Java matcher compares the

APPLICATION_USERNAME symbol value to the value `FRED`. If you did not define the APPLICATION_USERNAME symbol in a previous security rule, the Java matcher instead compares FRED to a predefined Dynamic Data Masking symbol. Because the values in this example not equal, the Java matcher returns a value of true and the Rule Engine applies the masking rule action in the Java matcher rule.

However, if the application submits the following query:

```
select customer_name from customer --Fred
```

The Java matcher returns a value of true and the Rule Engine does not apply the rule action in the Java matcher rule and allows Fred to see masked data.

Error Handling

If the Java class fails, the Rule Engine does not apply the security rule. The Rule Engine continues to the next rule in the rule tree.

The Dynamic Data Masking service stores the Java printout messages in the `log\DDMError.txt` file.

Parser Matcher

The Parser matcher identifies requests that Dynamic Data Masking is unable to parse.

You can use the Parser matcher with security rule actions to change the request. For example, if you want to block requests that Dynamic Data Masking cannot parse, you can create a rule with the following properties:

Rule Name

Enter a name for the rule. In this example, the rule name is parsed_statement.

Rule Matcher

Select the Parser matcher.

Rule Action

Select the Block Statement rule action. Enter a message that Dynamic Data Masking sends to the client when the statement is blocked. In this example the message is UNABLE TO PARSE STATEMENT.

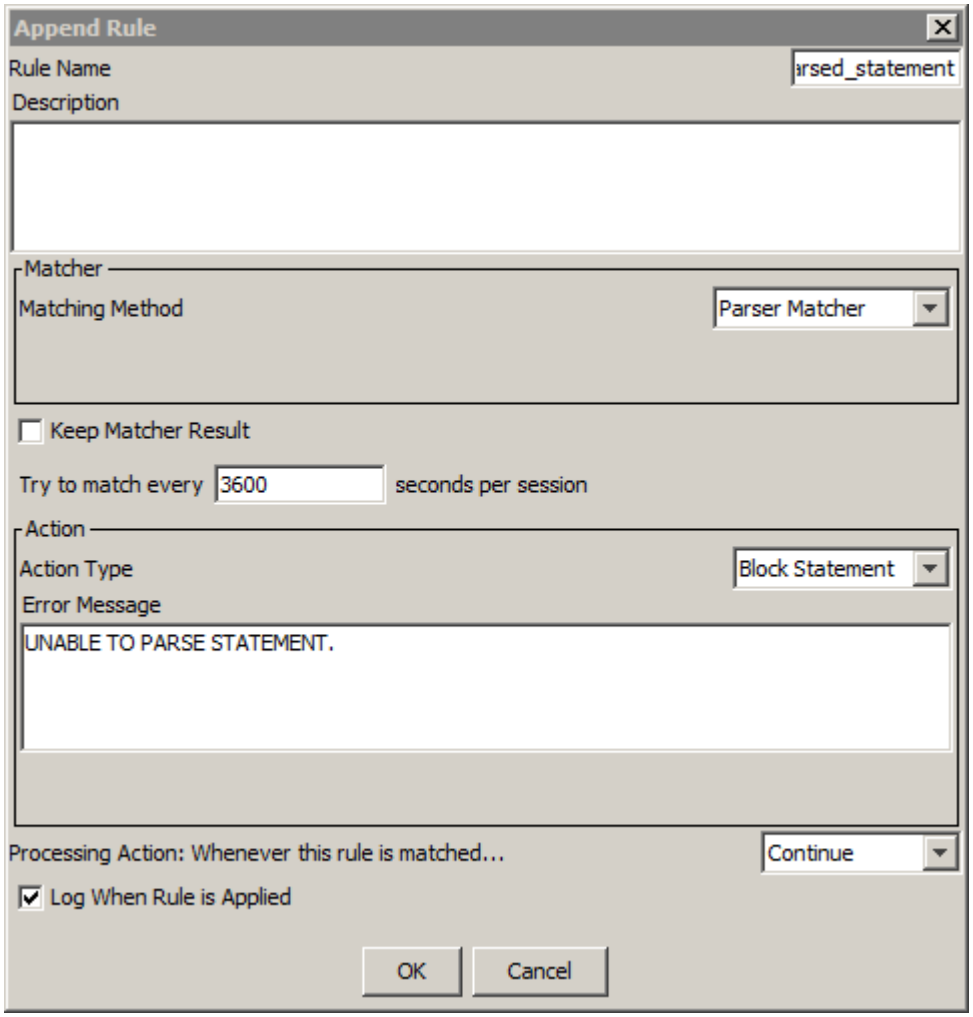
Processing Action

Select Continue. The Continue processing action indicates that the Rule Engine should continue to the next rule in the rule tree.

Log When Rule is Applied

Select the Log When Rule is Applied checkbox to create an entry in the rule.log file if the Rule Engine applies the rule action to the request.

The following image shows the rule parameters:



PL/SQL Function Matcher

The PL/SQL Function matcher returns a match based on the PL/SQL function that you define and the parameters of the function.

When you use the PL/SQL Function matcher, you enter the name or the name and path of the function. You must include the path if the stored object is not in the client schema or catalog, depending on the database. On a DB2 database, you must include the path if the stored object is not in the path that is returned by the following statement from the Dynamic Data Masking administrator:

```
SELECT CURRENT_PATH FROM SYSIBM.SYSDUMMY1
```

Select the parameters that the function uses. The Rule Engine returns an error if you do not select the correct parameters for the function. You can view the rule.log file or use the Security Rule Set Simulator to verify that you selected the correct parameters.

When the Rule Engine applies the PL/SQL Function matcher, the PL/SQL function returns a `1` or a `0`. If the PL/SQL function returns `1`, a match occurs and the Rule Engine applies the rule action. If the PL/SQL function returns `0`, a match does not occur and the Rule Engine does not apply the rule action.

Note: Informatica recommends that you create PL/SQL functions that have descriptive parameter names. If the parameter name describes the parameter value, the SQL script is comprehensible and easy to edit.

The PL/SQL Function matcher has the following properties:

Function Name

Enter the name and full path of the PL/SQL function. For example, you might enter `HR.dbo.get_employees`.

Include Database Username

Select if the function uses a database username parameter. The database username parameter defines the user for the session.

Include OS User

Select if the function uses an operating system user parameter. The operating system user parameter defines the user name for the user logged in to the operating system.

Include Client Host

Select if the function uses a client host parameter. The client host parameter defines the client host name or IP address.

Include Program Name

Select if the function uses a program name parameter. The program name defines the application or program name.

Include Statement

Select if the function uses a statement parameter. The statement parameter defines the SQL statement text. You must define the statement in VARCHAR.

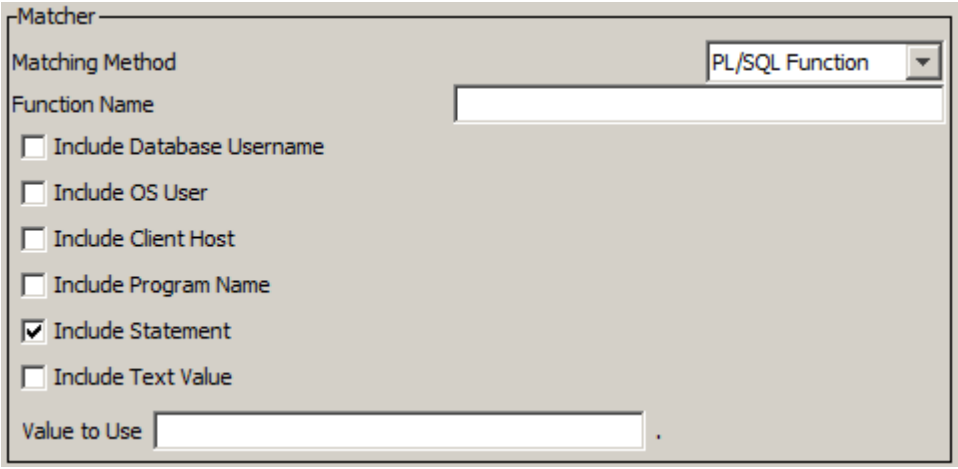
Include Text Value

Select if the function uses a text value parameter.

Value to Use

Enter a text value parameter if you selected the Include Text Value property. You can use text or a global symbol to define the text value.

The following image shows the PL/SQL Function matcher properties:



PL/SQL Function Matcher Examples

Use the PL/SQL Function matcher examples as guidelines to create PL/SQL functions and corresponding security rules.

Text Value Example

You want to mask the SALARY column of the EMP table based on the user that sends the request to the database. If the user is a senior manager, the result set is unmasked. If the user is not a senior manager, the result set contains a zero (0) in the SALARY column.

You create a PL/SQL function with the following code:

```
CREATE OR REPLACE FUNCTION test_senior_manager (position VARCHAR)
RETURN INTEGER AS
manager_type VARCHAR(50) ;
BEGIN
  manager_type := retrieve_manager_type(position) ;
  IF manager_type = 'SENIOR' THEN
     RETURN 0;
  ELSE
     RETURN 1;
  END IF;
END ;
```

The PL/SQL function returns `0` if the user is a senior manager and `1` if the user is not a senior manager.

You create the following security rule:

Append Rule

Rule Name: match_manager

Description

Matcher

Matching Method: PL/SQL Function

Function Name: usr1.test_senior_manager

- [ ] Include Database Username
- [ ] Include OS User
- [ ] Include Client Host
- [ ] Include Program Name
- [ ] Include Statement
- [x] Include Text Value

Value to Use: MANAGER

- [ ] Keep Matcher Result

Try to match every 3600 seconds per session

Action

Action Type: Mask

Columns to mask

Table Name	Column Name	Masking Function
emp	salary	'0'

+ -

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

- [x] Log When Rule is Applied

OK Cancel

If the user is a senior manager and the PL/SQL function returns `0`, Dynamic Data Masking does not apply the rule action and the result set is unmasked. If the user is not a senior manager and the PL/SQL function returns `1`, the rule action masks the values in the SALARY column according to the masking function.

Text Value With Multiple Arguments Example

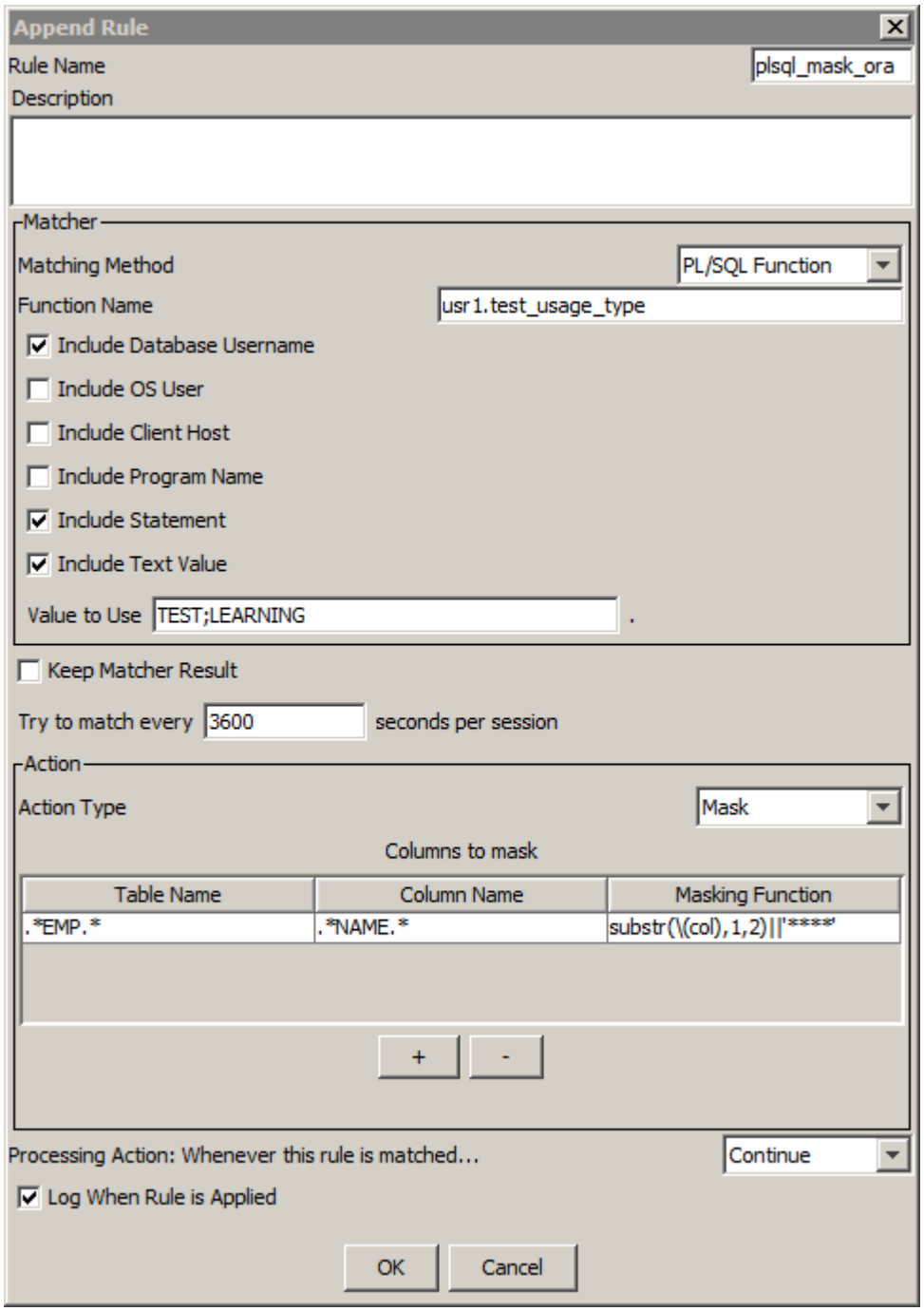
You want to mask data based on the user that sends the request and whether the statement returns columns that contain a specific type of data.

You create a PL/SQL function with the following code:

```
CREATE OR REPLACE TYPE LIST_TYPE IS VARRAY (5) OF VARCHAR(50);
CREATE OR REPLACE FUNCTION test_usage_type (db_user VARCHAR2, statement VARCHAR2, usages
VARCHAR2)
RETURN INTEGER AS
  usage_list LIST_TYPE;
BEGIN
  IF db_user = 'SCOTT' THEN
    IF is_usage_type_col_in_tables(statement) THEN
      usage_list := spiltSemicolonList(usages);
      IF check_usages(usage_list) THEN
        RETURN 1; -- If there is a usage in security rule that implies masking is needed
then return 1
      ELSE
        RETURN 0;
      END IF ;
    ELSE
      RETURN 0;
    END IF;
  ELSE
    RETURN 0;
  END IF;
END;
```

The PL/SQL statement returns `1` if the user is SCOTT and the returned columns contain data from the usage_list. Otherwise, the PL/SQL statement returns `0`.

You create the following security rule:



The rule contains Include Database Username and Include Statement parameters, which are fixed value arguments that you define in the PL/SQL function. The Value to Use parameter contains a list of delimiters that are separated by a semicolon. In this example, the delimiters are TEST and LEARNING. If the PL/SQL function matcher matches the request, the rule masks the NAME column of the EMP table.

OS User and Statement Example

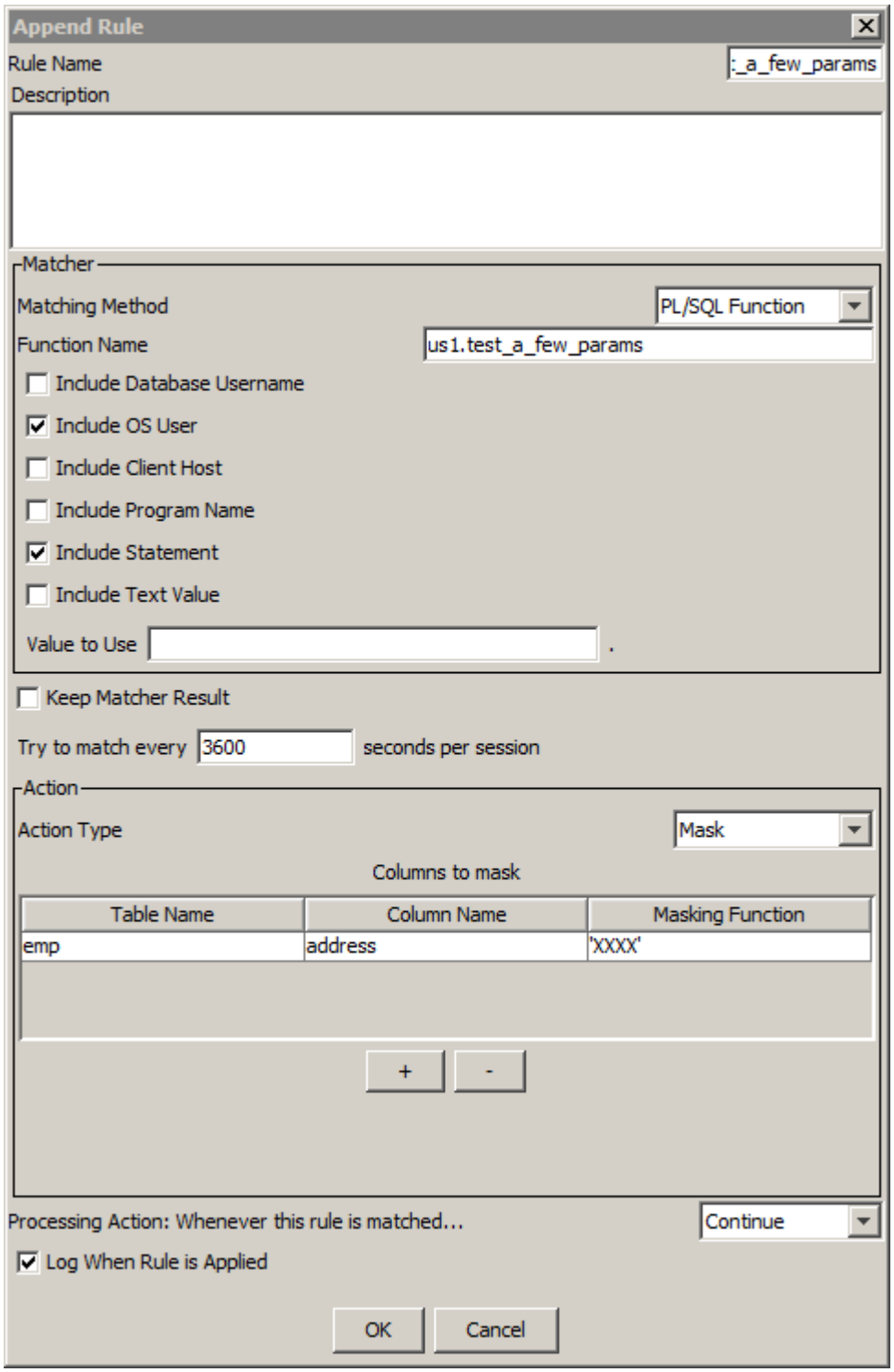
You want to mask data in the production database based on the operating system user that sends the request to the database.

You create a PL/SQL function with the following code:

```
CREATE OR REPLACE FUNCTION test_a_few_params (os_user VARCHAR2, statement VARCHAR2)
RETURN INTEGER AS
stmt_contains_prod_tables BOOLEAN ;
os_user_is_su BOOLEAN ;
needs_masking BOOLEAN ;
BEGIN
  stmt_contains_prod_tables := is_stmt_contains_prod_tables (statement) ;
  os_user_is_su := is_su (os_user) ;
  needs_masking := os_user_is_su AND stmt_contains_prod_tables;
  IF needs_masking THEN
     RETURN 1;
  ELSE
     RETURN 0;
  END IF;
END ;
```

The PL/SQL function returns `1` if the operating system user is su and the request contains production tables. Otherwise, the function returns `0`.

You create the following security rule:



If the operating system user is su and the request contains production tables, Dynamic Data Masking applies the masking function in the rule action. If the operating system user is not su or the request does not contain production tables, the PL/SQL function returns 0 and Dynamic Data Masking does not apply the rule action.

SQL Syntax Matcher

The SQL Syntax matcher compares the incoming SELECT statement to the SQL statement that you define within the matcher.

For a match to occur, the objects and predicates must be in the same order as the statement that you define in the rule. The objects and predicates that you define must ignore spaces, line breaks, and capitalization.

If you define an SQL statement that includes predicates that use literals, enter the literal value in the statement text. For SQL statements that include predicates that use bind variables, enter `Column name<operator>:[any string]` in the statement text. For example, if you enter `emp_number =:emp`, you replace statements that use any variation of emp_number.

The SQL Syntax matcher only matches SELECT statements. It does not match other data manipulation language commands such as INSERT, UPDATE, or DELETE.

Note: The SQL Syntax matcher ignores the alias of a table name.

The following table describes options for the SQL syntax matcher:

Option	Description
Allow bind variables to match any value in the statement	Enables the Rule Engine to match bind variables in the request statement. If you select the bind variable option, the Rule Engine matches any bind variable or literal. Bind variables can match functions, including `sysdate` or `to_date()`. For example, if the matching statement is `select * from cars where year=:year_no`, the Rule Engine matches all statements that use `select * from cars`. If you do not select the bind option, the Rule Engine matches request statements that exactly match `select * from cars where year=year_no`.
Allow any select list in the statement	Enables the Rule Engine to match request statements by using the defined statement, regardless of the contents of the select list.

Rewriting Statements With Bind Variables

Use bind variable naming conventions to rewrite incoming statements.

The Rule Engine identifies the values in the positions specified by bind variables in the incoming statement. You must enter the same bind variable names in the predicate of the rewrite statement. The Rule Engine identifies the values in the incoming statement and rewrites the statement with the inserted values.

Symbol Matcher

The Symbol matcher identifies and stores client and database variable values.

The Symbol matcher uses global variables to identify a match. A global variable has a name and a value. Global variables are case sensitive and have an assigned value for the database session. The database management system provides most of the values and there might be discrepancies between database versions of the same type.

For example, the Symbol matcher can identify a connection initiated by the database administration team. To identify team members that are not Dynamic Data Masking administrators and set a matching action that blocks access to the requested data, use the global variable AUTH_USERNAME. When the Rule Engine encounters the symbol, it blocks access to the data.

You can reference a global variable by placing the global variable in parentheses preceded with a backslash. For example, \(AUTH_USERNAME) is replaced with the database username.

The availability of a global variable is dependent on the database type. Use the Client Info entry in the rule.log file to identify global variable values.

You can use the following global variables:

AUTH_CURRENT_DATABASE

Defines the current database.

You can use AUTH_CURRENT_DATABASE on Sybase, IBM DB2, Informix, Teradata, and Microsoft SQL Server.

AUTH_DATABASE_NAME

Defines the Dynamic Data Masking database name that you define in the Management Console.

You can use AUTH_DATABASE_NAME on Oracle, IBM DB2, Sybase, Informix, Data Vault, Teradata, and Microsoft SQL Server.

AUTH_DRIVER_CLASS

The driver class name that the Generic JDBC client instruments at runtime.

You can use AUTH_DRIVER_CLASS with the DDM for JDBC service.

AUTH_DRIVER_NAME

The JDBC driver name.

You can use AUTH_DRIVER_NAME with the DDM for JDBC service.

AUTH_EXTERNAL_AUTHENTICATION

Defines whether the user is connected to the database with a password. If the user is connected to the database with a password, the value is FALSE. If the user is connected to the database without a password, the value is TRUE.

You can use AUTH_EXTERNAL_AUTHENTICATION on Informix.

AUTH_MACHINE

Defines the client host name and is not dependent on the client.

You can use AUTH_MACHINE on Oracle, Sybase, IBM DB2, Data Vault, and Microsoft SQL Server.

AUTH_PROGRAM_NM

Defines the program name.

You can use AUTH_PROGRAM_NM on Oracle, IBM DB2, Sybase, Microsoft SQL Server, and Teradata.

AUTH_SERIAL_NUM

Defines the session serial number.

You can use AUTH_SERIAL_NUM on Oracle.

AUTH_SESSION_ID

Defines the session ID configured for Dynamic Data Masking.

You can use AUTH_SESSION_ID on Oracle, Microsoft SQL Server, and Teradata.

AUTH_SID

Defines the client user name.

You can use AUTH_SID on Oracle and Teradata.

AUTH_TERMINAL

Defines the client host name and is dependent on the client. For example, if you use SQL Developer to connect to an Oracle database, AUTH_MACHINE contains the client host name, but AUTH_TERMINAL does not contain the client host name.

You can use AUTH_TERMINAL on Oracle.

AUTH_USERNAME

Defines the database user name.

You can use AUTH_USERNAME on Oracle, IBM DB2, Sybase, Informix, Microsoft SQL Server, Data Vault, and Teradata.

CLIENT_IP

Defines the IP address of the client machine.

You can use CLIENT_IP on Oracle, Sybase, Microsoft SQL Server, DB2, Informix, Teradata, Data Vault, and Hive databases.

Rewrite Select List Example

The following example is a simple method you can use to view the values of the global variable for a session.

Create a security rule that rewrites an SQL request, such as `SELECT * FROM ALL_GLOBALS`, and replaces the SQL request with the following query, based on the database management system type:

Oracle

```
select '\(CLIENT_IP)' "Client IP", '\(AUTH_USERNAME)' "Username",'\(AUTH_SID)' "OS
User",'\(AUTH_MACHINE)' "Host name", '\(AUTH_TERMINAL)' "OS user name", '\
(AUTH_SERIAL_NUM)' "SERIAL#", '\(AUTH_PROGRAM_NM)' "Program",'\
(AUTH_DATABASE_NAME)' "DDM Name",'\(AUTH_SESSION_ID)' "SID", '\
(AUTH_CURRENT_DATABASE)' "Current Database" from dual
```

IBM DB2

```
select '\(CLIENT_IP)' "Client IP", '\(AUTH_USERNAME)' "Username",'\(AUTH_SID)' "OS
User",'\(AUTH_MACHINE)' "Host name", '\(AUTH_TERMINAL)' "OS user name", '\
(AUTH_SERIAL_NUM)' "SERIAL#", '\(AUTH_PROGRAM_NM)' "Program",'\
(AUTH_DATABASE_NAME)' "DDM Name",'\(AUTH_SESSION_ID)' "SID", '\
(AUTH_CURRENT_DATABASE)' "Current Database" from sysibm.sysdummy1
```

Microsoft SQL Server

```
select '\(CLIENT_IP)' "Client IP", '\(AUTH_USERNAME)' "Username",'\(AUTH_SID)' "OS
User",'\(AUTH_MACHINE)' "Host name", '\(AUTH_TERMINAL)' "OS user name", '\
(AUTH_SERIAL_NUM)' "SERIAL#", '\(AUTH_PROGRAM_NM)' "Program",'\
(AUTH_DATABASE_NAME)' "DDM Name",'\(AUTH_SESSION_ID)' "SID", '\
(AUTH_CURRENT_DATABASE)' "Current Database"
```

Text Matcher

The Text matcher defines the statement string text that the Rule Engine uses to match a request statement.

The Rule Engine applies the rule action if the request statement fully matches the statement string text, including spaces and line breaks.

Wildcards

Use the percent sign (%) and the underscore character (_) to match parts of an incoming SQL statement.

The percent sign matches an undefined number of characters. The percent sign can match zero. The underscore character matches one character. The asterisk (*) is not a wildcard because it is a valid SQL syntax.

Regular Expression - Regex

You can use regular expression to identify incoming SQL statements that match the criteria that you define for the matcher. A regular expression consists of character literals and metacharacters.

The Rule Engine uses metacharacters to determine the algorithm for processing regular expression characters. The Rule Engine matches each character at least one time.

To match multiple instances of a character in a regular expression, apply a quantifier or a repetition operator. For example, to find a match that starts with the letter `a` and ends with the letter `b`, use the following the regular expression: `^a.*b$`. The asterisk metacharacter repeats the preceding match for any metacharacter (.) zero, one, or more times. If you want to use the like character to define the same pattern, use `a%b` with the percent sign to indicate zero, one, or multiple occurrences of any character.

The following table describes the regular expression metacharacters that you can use with Dynamic Data Masking:

Expression	Description
\t	Indicates a tab character.
\n	Indicates a new line.
.	Matches any character except for new lines.
|	Indicates the expression on the left or right side matches the target string.
[ ]	Defines the enclosed characters that can match the target character. For example, the expression `[ab]` matches the letters a and b. The expression `[0-9]` matches any digit.
[^]	Indicates that the enclosed characters do not match the target character. For example, `[^ab]` matches all characters except "a" and "b." `[^0-9]` matches any character that is not a digit.
*	Indicates that the character to the left of asterisk in the expression will match zero or more times. For example, `be*` matches "b," "be," and "bee."
+	Indicates that the character to the left of the plus sign matches one or more times. For example, `be+` matches `be` and `bee`, but not `b`.
?	Indicates that the character to the left of question mark in the expression matches zero or one time. For example, `be?` matches `b` and `be`, but not `bee`.
^	Indicates that the expression to the right of `^` matches when `^` is at the beginning of a line. For example, `^A` matches an `A` at the beginning of a line.
$	The expression to the left of `$` matches when `$` is at the end of a line. For example, the expression `e$` matches the letter `e` that is at the end of a line.
( )	Defines tagging and the processing order of expressions.

Expression	Description
\	Indicates an escape character. Use double backslashes to view the backslash character.
\Q	Indicates the start of a quoted expression. The parser ignores special characters.
\E	Indicates the end of a quoted expression.
\S	Defines a separator for space, tab, or new line.

Backslashes

Use the backslash character to apply escaped constructs and to quote characters that are interpreted as unescaped constructs. For example, the expression `\\` matches a single backslash, and the expression `\{` matches a left brace.

Java interprets backslashes that are used as string literals as Unicode or character escapes. To prevent the Java byte code compiler from interpreting them, use a double backslash in string literals that use regular expressions.

For example, the string literal `\b` matches a single backspace character when interpreted as a regular expression. However, `\\b` matches a word boundary. The string literal `\(hello\)` is illegal and results in a compile-time error. To match the string "(hello)," use the string literal `\\(hello\\)`.

Note: Do not use a backslash before any alphabetic character that does not denote an escaped construct. Use a backslash before a character if the character is not alphabetic and not part of an unescaped construct.

Character Classes

A character class contains a list of one or more characters. Use the bracket expression to create a character class and to match an expression that has a character in a given space.

You can use metacharacters, such as `-, ^, \, [:...:]`, to perform an advanced search on character classes. Use class shorthands, such as \d, \D, \s, \S, \w, and \W, as shortcuts for character classes.

The following table lists the precedence of character-class operators from highest to lowest:

Priority	Description	Character Class
1	Literal escape	\x
2	Group of characters	[aeiou]
3	Range of characters	[a-z]
4	Union of characters	[a-e][i-u]
5	Intersection of characters	[a-z&&[aeiou]]

Line Terminators

A line terminator is a one or two character sequence that indicates the end of a line for the input character sequence.

The following table describes the characters that you can use as line terminators:

Character	Description
\n	Newline character
\r\n	Carriage return character followed immediately by a new line character
\r	Standalone carriage return character
\u0085	Next line character
\u2028	Line separator character
\u2029	Paragraph separator character

Rules and Guidelines for Line Terminators

Consider the following rules and guidelines when you use a line terminator:

- If the `UNIX_LINES` mode is active, the Rule Engine recognizes the line terminators as new line characters.
- You can specify the `DOTALL` flag to indicate that the period character matches any character except for a line terminator.
- You can use the caret and dollar characters to ignore line terminators and to match the beginning and end of the input sequence.
- If the `MULTLINE` mode is active, the caret characters match the beginning of the input sequence and after any line terminator. The end of the input is not included. The dollar symbol matches before a line terminator and at the end of the input sequence.

Capturing Groups

A capturing group is a method that you can use to group multiple characters inside a set of parenthesis. Count the opening parentheses from left to right to enumerate the capturing group.

In the expression ((A)(B(C))), there are four groups: ((A)(B(C))), (A), (B(C)), and (C). Group zero stands represents the whole expression.

The captured characters in a group are the most recent sub-sequence that the group matched. All captured input is discarded at the beginning of each match. If a group is matched again, the previously captured value is retained if the second evaluation fails. For example, the matching the string "aba" in the expression `(a(b)?)+`changes group two set to `(B)`.

Groups that start with the expression `(?` do not capture text and do not count toward the group total.

Capturing Groups That Rewrite or Search and Replace

You can use line terminators and capturing groups to rewrite or search and replace expressions.

You can use parentheses to enclose tagged expressions to indicate a capturing group. Then, you can use `\0`, `\1`, `\2`, and `\3` to reference tagged expressions. The expression \0 indicates a tagged expression that

represents the entire substring that was matched. The expression `\1` denotes the first tagged expression, `\2` is the second expression, and `\3` is the third expression.

The following table describes how you can use capturing groups and line terminators to rewrite or search and replace text:

Original	Search	Replace	Result
abc	(ab)(c)	\0-\1-\2	abc-ab-c
abc	a(b)(c)	\0-\1-\2	abc-b-c
abc	(a)b(c)	\0-\1-\2	abc-a-c

Sample Expressions

To identify strings of text in an incoming SQL request, define regular expression character literals and metacharacters in the Text matcher.

The following table provides examples of regular expressions:

Regular Expression Matching	SQL Text	Description

`invoice \.number`
or
`sqlplus\.exe`
	`SELECT invoice.number FROM invoices`	The `\.` expression defines a period. The period matches any character. The backslash before the period character removes the value of the following character in regular expression.

`.*(customers)\s+.*`	`SELECT customers FROM dual`	The pattern matches statements with the text string `customers`. The expression `.*` indicates that any text can appear before the text string `customers`. The expression `\s+` after `customers` indicates that there must be one or more spaces. The expression `.*` indidcates that any text can follow the spaces.
`.*\s+(cust)+\s+.*`	`SELECT CustCustcust FROM dual`	The pattern matches text strings with one or more `cust` texts.
`.*WHERE.*(\W+TR\w* \s*)((=\s*')|(IN)| (between)|(>)|(, \s*')|(<)|(= \s*DMN_TABLE)).*`	`SELECT * FROM dual WHERE tr = 'sdf`	The pattern matches text strings that use nonword characters. A nonword character does not use [a-zA-Z_0-9]). The pattern defines `\W` followed by `TR` and any number of characters or spaces `\w*\s*`.
`create\s+or\s+replace \s+view\s+C\ $_C100008J13\s+\s+As \s+select\s+(.*)From \s+OA.S_CAMP_CON\s +S_C,\s+OA.S_ORG_EXT \s+S_O,\s +OA.S_ORG_EXT_X\s +Where\s+(.*)`	`CREATE or REPLACE VIEW C$_C100008J13 AS SELECT a, b, c, d, e, f FROM OA.S_CAMP_CON S_C, OA.S_ORG_EXT S_O, OA.S_ORG_EXT_X WHERE a=b and c = d`	The regular expression uses `\s+` to identify one or more spaces or the end of line in the original statement. The expression `C\$_C100008J13\` changes to `C$_C100008J13` because the dollar sign is a special sign in regular expression. If you do not want to use the regular expression version of the dollar sign, you must use the backslash character before the dollar sign.

Time of Day Matcher

The Time of Day matcher defines a time frame during which the Rule Engine matches statements.

The time format depends on the time zone defined for the operating system. To specify a different time zone, select a time zone from the **Time Zone** parameter.

For example, you can configure the Time of Day matcher to block requests between 6:00 p.m. and 8:00 a.m.

Security Rule Actions

Security rule actions define the masking technique applied to an SQL statement.

When an incoming SQL statement matches the matcher criteria, the Rule Engine applies the rule action that you define for the security rule. The rule action defines the masking technique that the Rule Engine applies to the SQL statement.

To determine how the Rule Engine processes the SQL statement, use the following actions :

- Block Statement
- Define Symbol
- Folder
- Java Action
- Log Message
- Mask
- Nothing
- Replace Table
- Rewrite
- Search and Replace

Block Statement Action

The Block Statement action does not allow the SQL request statement to reach the database.

If the Rule Engine applies a Block Statement action, the Dynamic Data Masking service sends an error message to the client or application.

Applications and tools can present the database error message and the Dynamic Data Masking error, which you can use to determine the reason that access is blocked.

Note: If you create a rule that uses the Any matcher and the Block rule action, the application will fail to start and will not return the Block message you specify in the Rule Editor. You can log the internal application SQL requests before you block access to the database with a rule that uses the Nothing rule action followed by a rule that uses the Block rule action. The application will initiate and send the Dynamic Data Masking message to the user before it blocks the request.

The following table describes the Block Statement action parameters:

Parameter	Description
Error message	Defines the error message sent to the client or application when the Rule Engine applies the block statement action.

Define Symbol Action

Use the Define Symbol rule action to define symbols and set the symbol value.

You can define multiple symbols with the Define Symbol rule action. Symbol values can contain symbols that you previously defined.

To define a global symbol, enter Yes in the Keep Per Session field. A global symbol lasts throughout the client session and exists until the client disconnects from the database.

If you do not want the symbol to be a global symbol, enter No in the Keep Per Session field. If you enter No in the Keep Per Session field, the symbol exists only during the SQL request. To use the symbol value in the subsequent rules in the security rule set, select the Continue processing action. The Rule Engine continues to the next rule and uses the symbol replacements.

If you want to define a symbol value based on a symbol that you previously defined, use the expression `\(Symbol_name)` to call the symbol value.

The following table describes the Define Symbol rule action parameters:

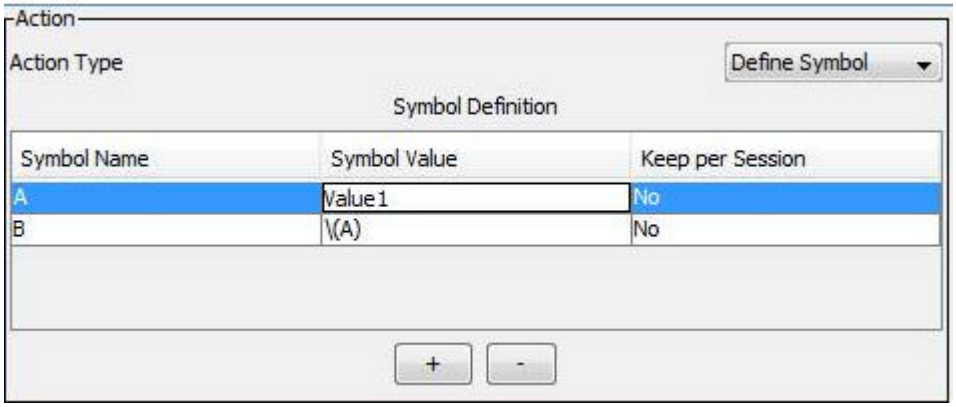
Parameter	Description
Symbol Name	The name of the symbol. The symbol name must begin with a letter and can contain ASCII characters, numbers, and underscores.
Symbol Value	The value of the symbol.
Keep Per Session	Defines whether the symbol is a global symbol. You can enter the following Keep Per Session values: - Yes. The symbol exists during the client session. - No. The symbol exists during the SQL request.

Defining Multiple Symbols

You can define symbols based on another symbol that you define in the same rule or a symbol that you defined in a previous rule.

To define a symbol based on another symbol in the same rule, use the expression `\(Symbol_name)` to call the symbol value in a subsequent row.

The following image shows a rule that defines two symbols:



In the image, the value of symbol A is set to Value1. Because symbol B calls the value of symbol A, the value of symbol B is also set to Value1.

However, if you reverse the order of the rows in the rule, symbol A is not defined when the value of symbol B is set, and symbol B has a null value.

Folder Action

The Folder rule action creates a folder in the rule tree. Use folders to nest and group rules.

For example, you can create a folder to group rules for a group of users or for an application. Use folders to manage rules and improve rule readability.

Java Action

The Java action runs a Java class that returns a string. The string value is an SQL statement that replaces the original SQL statement from the client request.

The Java class must contain a public static method with the following signature:

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

You must enter the correct class in the Class Path field of the Java action. The Rule Engine reads every rule in the rule set and returns an error if the class path is incorrect, even if the rule is disabled.

When an error occurs with the Java class, the Rule Engine does not apply the rule. Instead, the Rule Engine continues to the next rule. The Rule Engine logs a detailed error message in the following file:

```
<Dynamic Data Masking installation>\log\DDMError.txt
```

The Java action has the following parameters:

Class Path

Defines a list of jars that contain the Java action class and necessary libraries. Use semicolons to separate multiple jar files.

Class Name

Defines the fully qualified class name. The class may be built within a named package or an unnamed package. The class must include a method named `execute` with the following signature: `public static String execute(RuleContext ctx)`.

The `RuleContext` object contains data from the Rule Engine. The `RuleContext` object has one method, `public String getSymbol(String symbol);`, that returns the symbol value from the symbol name. Use this

method to retrieve symbol values defined by a Symbol rule action. Specify the name of the symbol as an argument to the method.

You can also use the following predefined symbols:

RuleContext.USERNAME

Retrieves the Oracle user name.

RuleContext.OS_USER

Retrieves the operating system user on the client machine.

RuleContext.CLIENT_HOST

Retrieves the name of the client machine.

RuleContext.PROGRAM

Retrieves the program name on the client machine.

RuleContext.STATEMENT

Retrieves the statement being processed.

Note: The Rule Engine runs the Java class with the JVM that Dynamic Data Masking uses. You can compile the Java action class with a JDK of the same major version or lower version of the JVM that Dynamic Data Masking uses.

Java Class Example

The following text is an example of a Java class. The Java action runs the execute method.

```
import com.activebase.rule.*;
public class ExampleJA {
    public static String execute(RuleContext ctx) {
        return "select 'OK' from dual";
    }
    public static boolean match(RuleContext ctx) {
        String user = ctx.getSymbol(RuleContext.USERNAME).toUpperCase();
        if ("SCOTT".equals(user)) {
            return true;
        }
        return false;
    }
```

Log Message Action

The Log Message action logs the security rule event.

You can use the Log Message rule action with loggers that you create in the Management Console tree. When you create the security rule, you include the name of the logger. The logger creates output based on its appenders.

The Send As property defines the severity of the event. The severity corresponds to the Log Level property that you set in the Dynamic Data Masking Management Console. 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.

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.

The following table describes the Log Message action parameters:

Parameter	Description
Logger Name	The name of the logger. The logger name corresponds to a logger in the Management Console tree.
Send As	The severity of the event. You can choose one of the following severities: - Information - Warning - Error
Logger Message	A message to write to the log. You can enter text or a regular expression logger message.

Mask Action

The Mask action rewrites the SELECT request. The Mask action uses a masking function to define how the Rule Engine rewrites the SELECT request.

The Rule Engine uses regular expression matching to verify whether the SQL statement contains sensitive object names. If a table name match occurs, the Rule Engine parses the SQL statement to identify select lists, which include the columns, aliases, objects, and inline queries. Based on the column names, objects, and mask functions you specify in the masking rule, Dynamic Data Masking rewrites the incoming SELECT request and changes the select list accordingly.

For masking functions, use the `\(col)` and `\(alias)` regular expressions to identify columns. For example, if you want to mask all columns that include a Social Security number, you can identify columns that you want to mask as `.*SSN.*`. The Mask rule action can include different column names, but you must define the Mask rule action only once. The Mask rule action might be `substring(\(col),1,2)||'xxxx')`.

The following table describes the Mask action parameters:

Parameter	Description
Table name	Defines the table name used in the statement request.
Column name	Defines the column name used in the statement request.
Masking function	Defines how the Rule Engine masks the select list. Use SQL functions to define the masking function.
Keep original number of rows	Preserves the original number of rows in the masked output. Select **Keep original number of rows** if the query contains the DISTINCT operator or a GROUP BY, HAVING, or ORDER BY clause and you want the output to contain the same number of rows as the original data set. Default is unchecked and the original number of rows is not preserved.

Keep Original Number of Rows Example

The following example shows possible outputs with and without the **Keep original number of rows** checkbox selected.

You have a data set that contains first name values:

FIRST_NAME	CONT(*)
LYNN	1
CYNTHIA	1
KEVIN	1
LESLIE	1
JOHN	1
JEAN	1

You create a masking rule with the following values in the Mask rule action:

Table Name	Column Name	Masking Function
.*EMPLOYEE.*	.*NAME.*	substr(\(col),1,2)||'****'

You send a SELECT query to the database:

```
SELECT FIRST_NAME,count(*) from EMPLOYEE group by FIRST_NAME
```

The following data set is an example of masked data that Dynamic Data Masking might return if you do not select the **Keep original number of rows** checkbox:

FIRST_NAME	CONT(*)
J****	2
K****	1
L****	2
C****	1

The masked output does not have the same number of rows as the original data set.

The following data set is an example of masked data that Dynamic Data Masking returns if you select the **Keep original number of rows** checkbox:

FIRST_NAME	CONT(*)
L****	1
C****	1

FIRST_NAME	CONT(*)
K****	1
L****	1
J****	1
J****	1

The masked output has the same number of rows as the original data set.

Masking SQL Batch

If you use Dynamic Data Masking with a Sybase or Microsoft SQL Server database, you can mask SQL batch.

Dynamic Data Masking masks the following types of SQL batch use cases:

- SQL batch without a delimiter.
  For example:

```
select * from account
select  * from employee
```

- SQL batch with a delimiter.
  For example:

```
select * from account;
select  * from employee;
```

- SQL batch that contains an IF ... ELSE block.
  For example:

```
if (select 123) < 7000
 select * from EMPLOYEE where EMPLOYEE_ID = 15120
else
 select EMPLOYEE_ID from EMPLOYEE
```

- SQL batch that contains an IF ... ELSE block and a BEGIN ... END statement.
  For example:

```
if (select EMPLOYEE_ID from employee where EMPLOYEE_ID=123) < 7000
BEGIN
 select *  from account;
 END
else
 BEGIN
  select *  from employee;
 END
```

- SQL batch that contains an IF EXISTS or IF NOT EXISTS statement.
  For example:

```
select * from employee
select top 2 * from employee
select a.Last_name  as name from employee a

select distinct a.first_name as name
from employee a inner join employee b on a.first_name=b.first_name

if exists (select  distinct (last_name) from employee where last_name = 'SMITH')
select top 3 a.last_name as name
from employee a left outer join employee b on a.last_name=b.last_name

select last_name from employee group by last_name
```

- SQL batch that contains an IF EXISTS or IF NOT EXISTS statement and a BEGIN ... END statement.
  For example:

```
if exists (select distinct (empid) from employee where empid = 101)
BEGIN
select * from employee
select * from account
END
```

- SQL batch that contains a GO statement.
  For example:

```
select * from account
select  * from employee
GO
select * from account
```

- SQL batch that contains a GO statement with an argument.
  For example:

```
select * from account
select  * from employee
GO 3
```

Dynamic Data Masking does not mask the following types of SQL batch use cases:

- SQL batch that contains an anonymous SQL block.
  For example:

```
BEGIN
select * from employee
select * from account
END
```

- SQL batch that declares a stored procedure.
  For example:

```
Create procedure SP1 as
select * from employee
```

- SQL batch that contains a SET command.
  For example:

```
SET QUOTED_IDENTIFIER OFF
select * from employee
```

- SQL batch that contains a USE command.
  For example:

```
USE DDM
select * from employee
```

- SQL batch that declares a statement.
  For example:

```
DECLARE       @return_value int
SET    @return_value = 101
SELECT * FROM EMPLOYEE where EMPID = @return_value
```

- SQL batch that calls a stored procedure.
  For example:

```
select * from employee
EXEC SP1 10,20
select * from account
```

Nothing Action

The Nothing rule action does not apply an operation on the processed SQL request.

You can use the Nothing action to audit the SQL request by selecting Log When Rule is Applied.

If Dynamic Data Masking does not need to process data manipulation language commands, you can use the Nothing action to make SQL requests bypass rule processing sections. For example, you can create a rule tree in which the first rule identifies SQL requests with data manipulation language commands. In the rule, select the Nothing rule action and the Stop if Applied processing action. Dynamic Data Masking sends SQL requests with data manipulation language commands to the database instead of to the next rule in the tree.

For example, if you do not want to mask data manipulation language commands for a specific table, apply the Nothing action to data manipulation language commands. When the Rule Engine applies the rule to a request statement that uses data manipulation language commands, the Rule Engine does not mask the statement.

PL/SQL Function Action

The PL/SQL Function rule action replaces an incoming request with a valid query that Dynamic Data Masking sends to the database for execution.

When you use the PL/SQL Function action, you enter the name or the name and path of the function. You must include the path if the stored object is not in the client schema or catalog, depending on the database. On a DB2 database, you must include the path if the stored object is not in the path that is returned by the following statement from the Dynamic Data Masking administrator:

```
SELECT CURRENT_PATH FROM SYSIBM.SYSDUMMY1
```

Select the parameters that the function uses. The Rule Engine returns an error if you do not select the correct parameters for the function. You can view the rule.log file or use the Security Rule Set Simulator to verify that you selected the correct parameters.

The PL/SQL Function rule action has the following properties:

Function Name

Enter the name and full path of the PL/SQL function. For example, you might enter `HR.dbo.get_employees_masked`.

Include Database Username

Select if the function uses a database user name parameter. The database username parameter defines the user for the session.

Include OS User

Select if the function uses an operating system user parameter. The operating system user parameter defines the user name for the user logged in to the operating system.

Include Client Host

Select if the function uses a client host parameter. The client host parameter defines the client host name or IP address.

Include Program Name

Select if the function uses a program name parameter. The program name defines the application or program name.

Include Statement

Select if the function uses a statement parameter. The statement parameter defines the SQL statement text. You must define the statement in VARCHAR.

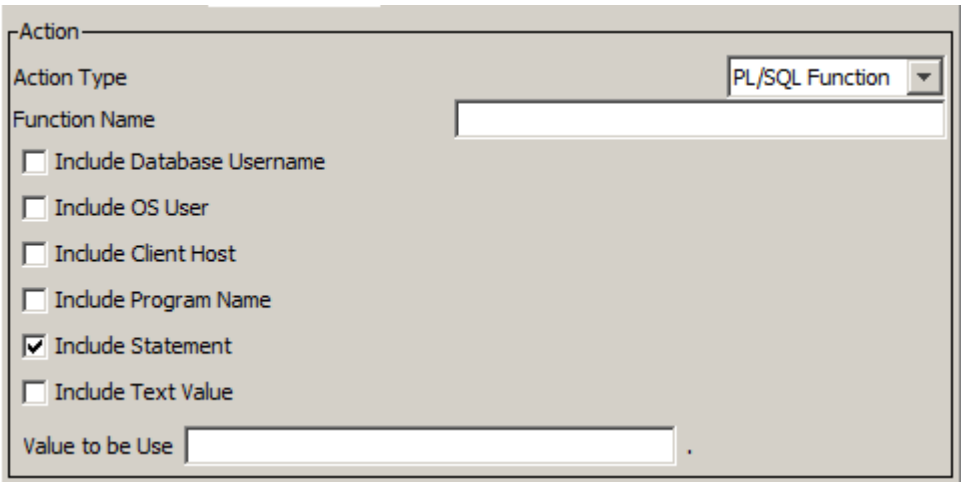
Include Text Value

Select if the function uses a text value parameter.

Value to Use

Enter a text value parameter if you selected the Include Text Value property. You can use text or a global symbol to define the text value.

The following image shows the PL/SQL Function rule action properties:



Replace Table Action

The Replace Table rule action removes rows that contain sensitive data from the result set.

You can use the Replace Table action to change the FROM clause of the original SQL statement to provide row-level security. You can replace the table reference with a table reference or subquery that you define in the Alternative Object or Query field.

The Alternative Object or Query filed can contain global symbols or the Replace Table rule action symbols. You can define global symbols in other security rules that you wan to use in the Replace Table rule action.

The Replace Table rule action symbols relate to the current SQL statement FROM clause. Use the \(prefix), \(tableOnly), and \(table) symbols to specify that you want to preserve the table name, preserve the path, or replace the path and the table name.

The following table describes the Replace Table action parameters:

Parameter	Description
Table Name	A regular expression that represents the database object names that you want to replace.
Alternative Object or Query	The object name or query that you want to replace the original table name.

Replace Table Action Symbols

You can use symbols to specify which part of the SQL statement you want to replace.

Use the \(table), \(prefix), and \(tableOnly) symbols in the Alternative Object or Query field to replace part or all of the FROM clause of the current SQL request.

The following table describes the Replace Table action symbols:

Symbol	Description
\(table)	Replaces the full object reference in the original SQL statement with the Table Name parameter value.
\(prefix)	Replaces the prefix of the object reference, including the separator, with the Table Name parameter value.
\(tableOnly)	Replaces the table name with the Table Name parameter value, but preserves the path.

For example, you want to use symbols to alter the following statement:

```
SELECT * FROM [HR].[DBO].employee
```

The following table describes which parts of the SQL statement the symbols replace:

Symbol	Replaced Value
\(table)	[HR].[DBO].employee
\(prefix)	[HR].[DBO].
\(tableOnly)	employee

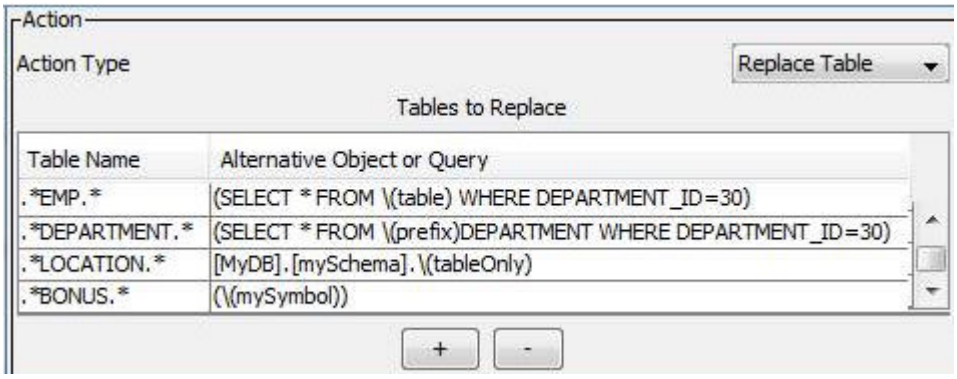
Replace Table Example

You want to use the Replace Table rule action to mask a variety of SQL requests.

You want to mask incoming requests to the EMP, DEPARTMENT, and LOCATION tables. In a previous security rule, you defined the global symbol (\mySymbol) with the following value:

```
SELECT * FROM BONUS WHERE ENAME LIKE 'S%'
```

The following figure shows the Replace Table rule action parameters that you define:



The following table describes sample SQL requests and how Dynamic Data Masking alters the requests based on the Replace Table rule action parameters:

Original Statement	Replaced Statement
SELECT * FROM [HR].[DBO].employee	SELECT * FROM ((SELECT * FROM "HR"."DBO"."EMPLOYEE" WHERE DEPARTMENT_ID=30)) "EMPLOYEE"
SELECT * FROM [HR]..EMP1	SELECT * FROM ((SELECT * FROM "HR".."EMP1" WHERE DEPARTMENT_ID=30)) "EMP1"
SELECT * FROM [HR].[DBO].Department	SELECT * FROM ((SELECT * FROM "HR"."DBO".DEPARTMENT WHERE DEPARTMENT_ID=30)) "DEPARTMENT"
SELECT * FROM Department	SELECT * FROM ((SELECT * FROM DEPARTMENT WHERE DEPARTMENT_ID=30)) "DEPARTMENT"
select * from [hr].[dbo].location	select * from [MyDB].[mySchema]."LOCATION" "LOCATION"
select * from location	select * from [MyDB].[mySchema]."LOCATION" "LOCATION"
SELECT * FROM BONUS	SELECT * FROM (SELECT * FROM BONUS WHERE ENAME LIKE 'S%') "BONUS"

Rewrite Action

The Rewrite action replaces the SQL statement with an alternate statement. You can rewrite any type of SQL request, including data manipulation language commands, DIGITAL Command Language commands, and data definition language commands.

You can use symbols and reference bind variables to rewrite predicates in the incoming SQL statement.

If you want to use a symbol value to rewrite all or part of the SQL statement, use the expression `\(Symbol_name)` in the Alternate Statement field.

The following table describes the Rewrite action parameters:

Parameter	Description
Alternate statement	Defines the SQL statement that the Rule Engine uses to rewrite the original SQL statement.

Rewriting Statements With Bind Variables

Use bind variable naming conventions to rewrite incoming statements.

The Rule Engine identifies the values in the positions specified by bind variables in the incoming statement. You must enter the same bind variable names in the predicate of the rewrite statement. The Rule Engine identifies the values in the incoming statement and rewrites the statement with the inserted values.

Search and Replace Action

The Search and Replace action searches for specific text within a statement request and replaces it with the text that you define in the replacement string.

You can use symbols to rewrite predicates in the incoming SQL statement.

The following table describes the Search and Replace action parameters:

Parameter	Description
Search Text	Defines the search text. Enter the search text by using a string, wildcard, or regular expression.
Replacement String	Defines the replacement text. Enter string values or symbols for the replacement text.

Processing Actions

A processing action defines what the Dynamic Data Masking service does after the rule action has been performed. In the **Edit Rule** window, the processing action is referred to as Whenever this Rule is Matched.

The following table describes the processing actions after an SQL request is matched by a rule:

Processing Rule Action	Description
Continue	The Rule Engine continues to the next rule in the rule tree. **Note:** Assign the Continue processing action to rule folders so that the Rule Engine applies the rules within the rule folder.
Stop if Applied	If the rule is applied, the Rule Engine does not continue to the next rule in the rule tree. The SQL request is sent to the database for execution.
Stop if Matched	If the rule is matched, the Rule Engine does not continue to the next rule in the rule tree. The SQL request is sent to the database for execution.

Creating a Security Rule

Create a security rule to define the criteria that the Rule Engine uses to parse an SQL request. To create a security rule, configure a matcher, action, and processing action.

Use the following high-level steps to create a security rule. The steps vary based on the types of matcher methods and actions that the rule uses.

1. In the Management Console, click the security rule set that you want to add the rule to.
2. Select **Tree** > **Security Rule Set**.

   The **Rule Editor** appears.
3. Click **Action** > **Append Rule**.

   The **Append Rule** window appears.
4. Enter a name for the rule in the **Rule Name** field.
5. Enter an optional description for the rule.
6. Select a matcher type and define the matching criteria.
7. Select **Keep matcher result** to store the result if a match occurs.
8. Choose a rule action and define the rule action criteria.

9. Choose a rule processing action.
10. Select **Log when rule is applied** to log the rule information in the rule log.
11. Click **OK**.

    The rule appears in the rule tree.
12. Select **File** > **Update Rules** to save the rule in the rule tree.

Security Rule Export and Import

Export security rules within security rule folders. You can save security rule configurations in an XML file. You can import XML files to use saved security rule configurations.

Security Rule Export

You can export a security rule folder to an XML file to back up security rule definitions. Make a copy of a rule folder for complex rule configurations that you want to reuse. Use the export utility to make a copy of a rule folder and import the copy to the security rule tree.

An exported file contains the security rules and the security rule components, such as the security rule matchers, security rule actions, and security rule processing actions. You must export security rules within a folder. You must export security rules inside a security rule folder or security rule set. To export a single security rule, create a folder that contains only the security rule.

Exporting Security Rules

Use the Management Console to export security rules within a security rule folder.

1. In the Management Console, click **Tree** > **Security Rule Set**.

   The **Rule Editor** appears.
2. Click the security rule folder that you want to export.
3. Click **Action** > **Export**.

   The **Export** window appears.
4. Browse to a location to save the export file.
5. Enter a name for the export file.
6. Click **Export.**

Security Rule Import

When you import a security rule file, you import the security rules and the security rule components that the file defines. The security rule components include matchers, actions, and processing actions. You can import security rules into an empty rule folder or to the top of the security rule tree. After you import the security rules, update the rule tree.

Before you import security rules, you must create an empty rule folder in the security rule tree. If you import into a rule folder that is not empty, the imported rules override the rules in the rule folder.

If you choose to import the security rules to the top of the security rule tree, the contents of the imported file overrides the existing rules in the rule tree. The existing rules are deleted.

Importing Security Rules

Import security rules through the Management Console.

1. In the Management Console, click **Tree** > **Security Rule Set**.

   The **Rule Editor** appears.
2. Click an empty rule folder or the top of the rule tree. Rules that you import appear in the location that you choose.
3. Click **Action** > **Import**.

   The **Import** window appears.
4. Browse to the location of the XML file that defines the security rules that you want to import and double-click the file.
5. Select the file and click **Import**.

   The rule tree displays the security rules.
6. Select **File** > **Update Rules**.

   The rule tree is updated.

Security Rule Set Export and Import

Export security rule sets within the Management Console. You can save security rule set configurations in an XML file. You can import XML files to use saved security rule set configurations.

Security Rule Set Export

You can export a security rule set to an XML file to back up security rule definitions. Additionally, you can use the export utility to make a copy of a rule set and import the copy to the security rule tree. You might want to make a copy if you have a complex rule set structure that you want to reuse. Exported security rule sets retain the tree rule structure.

An exported file contains the components of a security rule set, such as the rule folders, security rule matchers, actions, and processing actions for each rule in the rule set.

Exporting Security Rule Sets

Use the Management Console to export security rule sets.

1. In the Management Console, click the security rule set that you want to export.
2. Click **Tree** > **Security Rule Set**.

   The **Rule Editor** appears.
3. Click the rule set name at the top of the rule tree.
4. Click **Action** > **Export**.

   The **Export** window appears.
5. Browse to the location where you want to save the rule set.
6. Enter a name for the exported file.
7. Click **Export.**

Security Rule Set Import

You can import a security rule set into the rule tree or into a rule folder. When you import a rule set, the rule tree displays the original structure of the rule set. After you import the security rules, save and update the rule set to the rule tree.

Before you import a security rule set, create an empty security rule set in the rule tree. Rules that you import appear in this rule set. If you import security rules into a rule set that is not empty, the imported rule set overrides the existing rule set.

You can also import a rule set into a rule folder that is part of an existing rule set.

Importing Security Rule Sets

Import security rule sets through the Management Console.

1. In the Management Console, click the rule set to which you want to add the imported rule set.
2. Click **Tree** > **Security Rule Set**.

   The **Rule Editor** appears.
3. Click the rule set name at the top of the rule tree to import the rule set directly into the security rule tree. Or click a rule folder to import the rule set into a rule folder.
4. Click **Action** > **Import**.

   The **Import** window appears.
5. Browse to the location of the XML file that contains the rule set and double-click it.

   The rules appear in the security rule tree.
6. Select **File** > **Update Rules**.

   The rule tree is updated.

CHAPTER 5

Security Rule Set Simulator

This chapter includes the following topics:

- Security Rule Set Simulator Overview, 72
- Running the Security Rule Set Simulator, 72
- Working with the Security Rule Set Simulator, 73
- Security Rule Set Simulator Parameters , 73
- Simulating a Rule, 75
- Security Rule Set Simulator Example, 76

Security Rule Set Simulator Overview

Use the Security Rule Set Simulator to test security rules and regular expressions within the Management Console.

You can access the Security Rule Set Simulator through the Rule Editor. Within the Security Rule Set Simulator, you select a database, enter database user information, define symbols, and enter an SQL statement that you want to use to test the rule. The simulator displays log output that you review to verify whether you created the security rule correctly. If you want to view detailed log information, you can run the simulator in debug mode.

Running the Security Rule Set Simulator

You can run the Security Rule Set Simulator on a rule, a rule folder, or a rule set.

Access the Security Rule Set Simulator from the **Action** menu in the Rule Editor. Select **Action** > **Security Rule Set Simulator**.

Run the Security Rule Set Simulator on the following rule tree nodes:

Rule

To run the simulator on a rule, select the rule in the Rule Editor and open the Security Rule Set Simulator. If you run the simulator on a rule, the simulator tests only the rule, even if the rule uses the Continue processing action.

Rule Folder

To run the simulator on the rules in a rule folder, select the rule folder in the Rule Editor and open the Security Rule Set Simulator. If you run the simulator on a rule folder, the simulator tests all the rules in the folder. Even if the last rule in the folder uses the Continue processing action, the simulator stops at the last rule.

Rule Set

To run the simulator on all the rules in the rule set, select the rule set in the Rule Editor and open the Security Rule Set Simulator. The simulator runs through the rules in the rule set and applies the processing actions.

Working with the Security Rule Set Simulator

You can choose to run the Security Rule Set Simulator without a database or a database user.

When you run the Security Rule Set Simulator, you enter a database name and database user login information. To receive authentic results, Informatica recommends that you enter a privileged user name and password when you run the Security Rule Set Simulator. If you choose to run the simulator without database information, the simulator displays the log output with a warning that you did not specify the database or database user.

Some SQL requests require the Dynamic Data Masking Server to access the database. To receive accurate log output, you must define the database and the user name and password of a privileged user in the simulator. The following types of requests require the Dynamic Data Masking Server to access the database:

- Security rules that use the PL/SQL rule matcher
- SQL statements that contain an asterisk (*)
- Requests that retrieve information about the case sensitivity of the database

Security Rule Set Simulator Parameters

The Security Rule Set Simulator contains fields for database information, symbol definitions, an SQL statement, and log output.

The Security Rule Set Simulator contains the following parameters:

DDM Database Name

The name of the Dynamic Data Masking database that you defined in the Management Console. If you do not enter a database name, the simulator runs without a database.

DBA Username

The name of the database user that you want to use to connect to the database. The user should be a privileged user that has access to all the tables in the database. Generally it is the user that you used to connect to the database in the Management Console.

Note: If you want to simulate the rule for a non-privileged user, enter login information for a privileged user and enter the non-privileged user name in the AUTH_USERNAME symbol in the **Symbol Definition** box.

DBA Password

The password for the database user.

Symbol Definition

A list of Dynamic Data Masking symbols or variables. When you open the Security Rule Set Simulator, the **Symbol Name** column lists the default global symbols. You can enter a value for the symbol in the **Symbol Value** column, or you can choose not to define the symbol. If you want to add additional symbols, click the plus (+) icon below the **Symbol Definition** box.

To run the simulator as a database user that is different from the database user that you entered in the **DBA Username** and **DBA Password** fields, define a value for the AUTH_USERNAME symbol. For example, you might enter a non-privileged user name in the AUTH_USERNAME **Symbol Value** field. The simulator runs the SQL statement as if the non-privileged user sent the request.

Note: Symbols that you define in the Security Rule Set Simulator only exist within the simulator and do not affect the Dynamic Data Masking Server configuration.

SQL Statement

The SQL statement that you want to run in the simulator.

Log Output

The log output from the last simulation. Dynamic Data Masking does not save the Security Rule Set Simulator log output in the Dynamic Data Masking log files. To save the log output from the simulator, click **Save File** at the bottom of the Security Rule Set Simulator.

Debug Mode

Runs the simulator in debug mode and creates a detailed log output that displays information similar to the log output in the server.log file.

Run

Runs the simulator.

Note: When you run the simulator, the simulator overwrites the log output from the previous simulation. If you want to save the log output from the previous simulation, click **Save File** before you run the simulator.

Open File

Opens a dialog box that you use to import a saved simulator file into the Security Rule Set Simulator.

Save File

Opens a dialog box that you use to save the save the simulator configuration and log output as an xml file. By default, the dialog box opens to the user root directory.

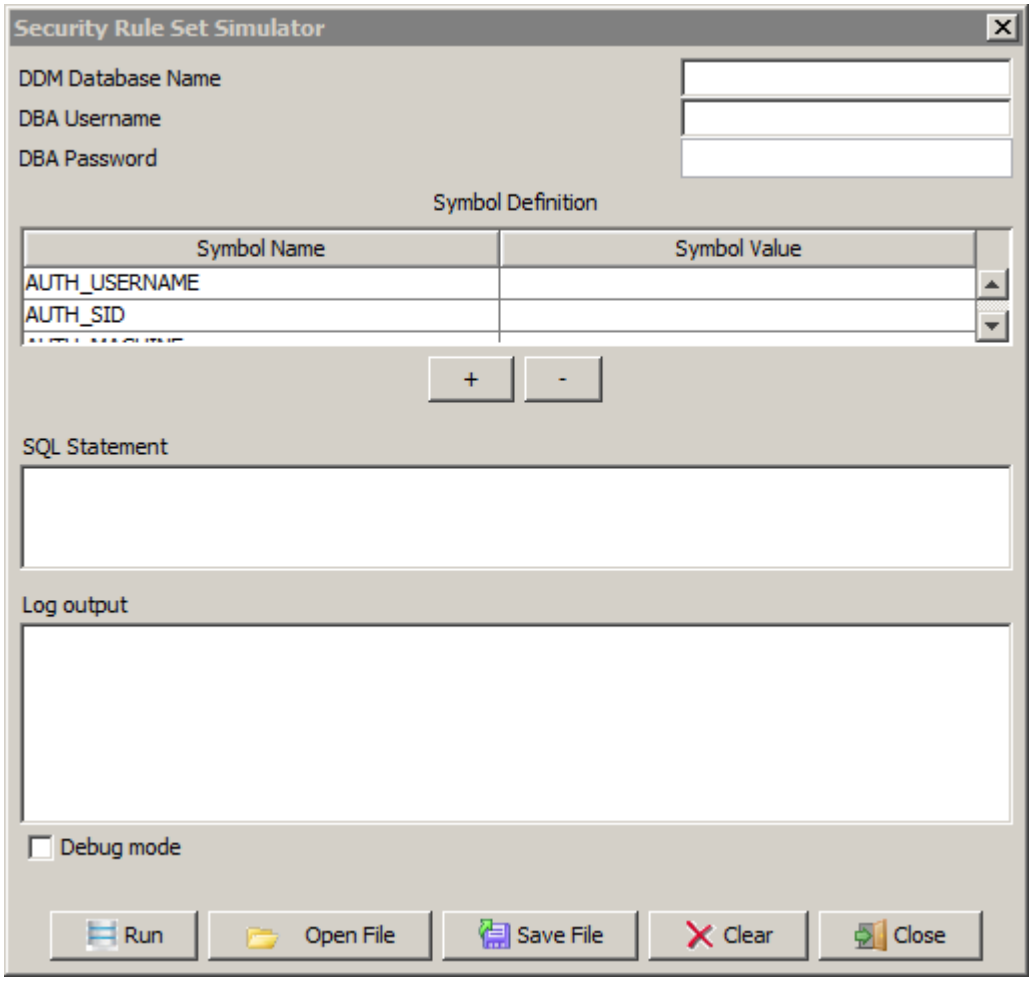
Clear

Clears the simulator configuration and log output.

Close

Closes the Security Rule Set Simulator. The Security Rule Set Simulator saves the simulator configuration and log output until you close the security rule set. When you close the rule set, the simulator resets.

The following image shows the Security Rule Set Simulator:



Simulating a Rule

Use the Security Rule Set Simulator to test a Dynamic Data Masking security rule before you use it in production.

1. In the Management Console, open a security rule set.
2. In the Rule Editor, select the rule, folder, or rule set that you want to test.

   For example, if you want to test the entire rule set, click the rule set name. If you want to test rules in a rule folder, click the folder name. If you want to test a single rule, click the rule name.
3. Click **Action** > **Security Rule Set Simulator**.

   The **Security Rule Set Simulator** opens.
4. Enter the following database details:

DDM Database Name

The name of the database that you saved in the Management Console tree.

DBA Username

The name of the database user that you want to use to connect to the database. The user should be a privileged user that has access to all the tables in the database. Generally it is the user that you used to connect to the database in the Management Console.

Note: If you want to test the rule for a non-privileged user, enter login information for a privileged user and enter the non-privileged username in the AUTH_USERNAME symbol in the **Symbol Definition** box.

DBA Password

The password for the database user.

5. Optionally, enter one or more symbol values in the **Symbol Definition** box. The simulator lists the global symbols. To add a symbol, click the plus (+) button.

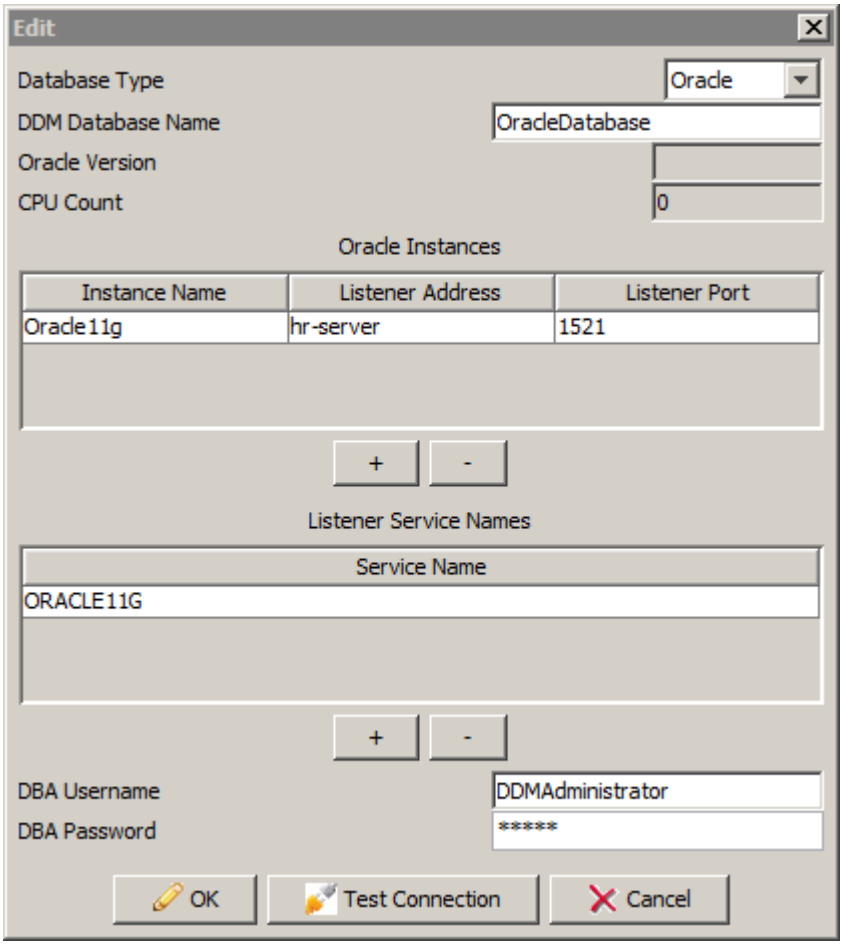
   If you want to validate whether the security rule works for a non-privileged user, enter the database user name of the non-privileged user as the AUTH_USERNAME symbol value.
6. Enter the SQL statement that you want to use to test the rule or rules.
7. Click **Run**.

   The simulator runs and displays the log information in the **Log Output** box.
8. If you want to save the simulator configuration and the log output, click **Save File**. Dynamic Data Masking does not save the simulator log output to the Dynamic Data Masking log files.

Security Rule Set Simulator Example

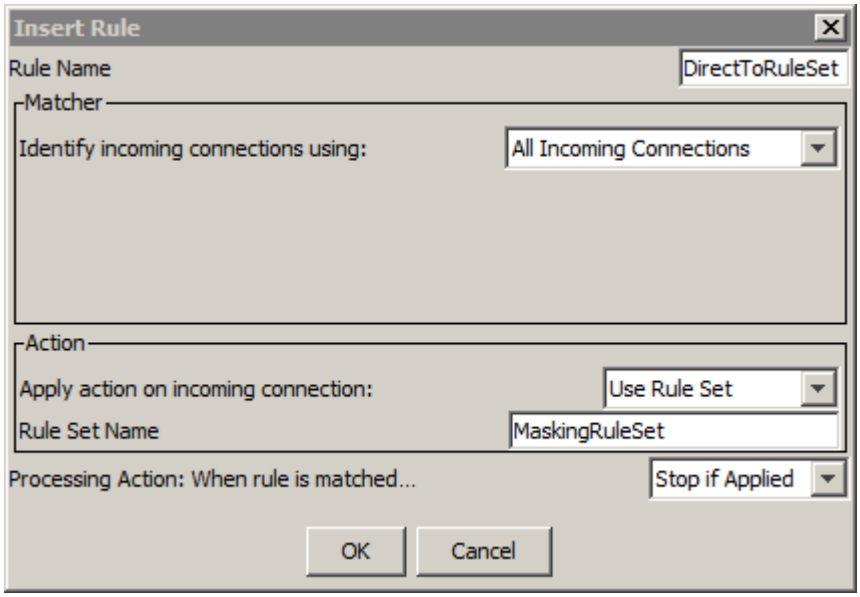
You have a database that contains sensitive employee information and you want to create and test a masking rule that prevents non-privileged database users from viewing the sensitive data.

You create a database connection in the Management Console. The following image shows the database connection parameters:



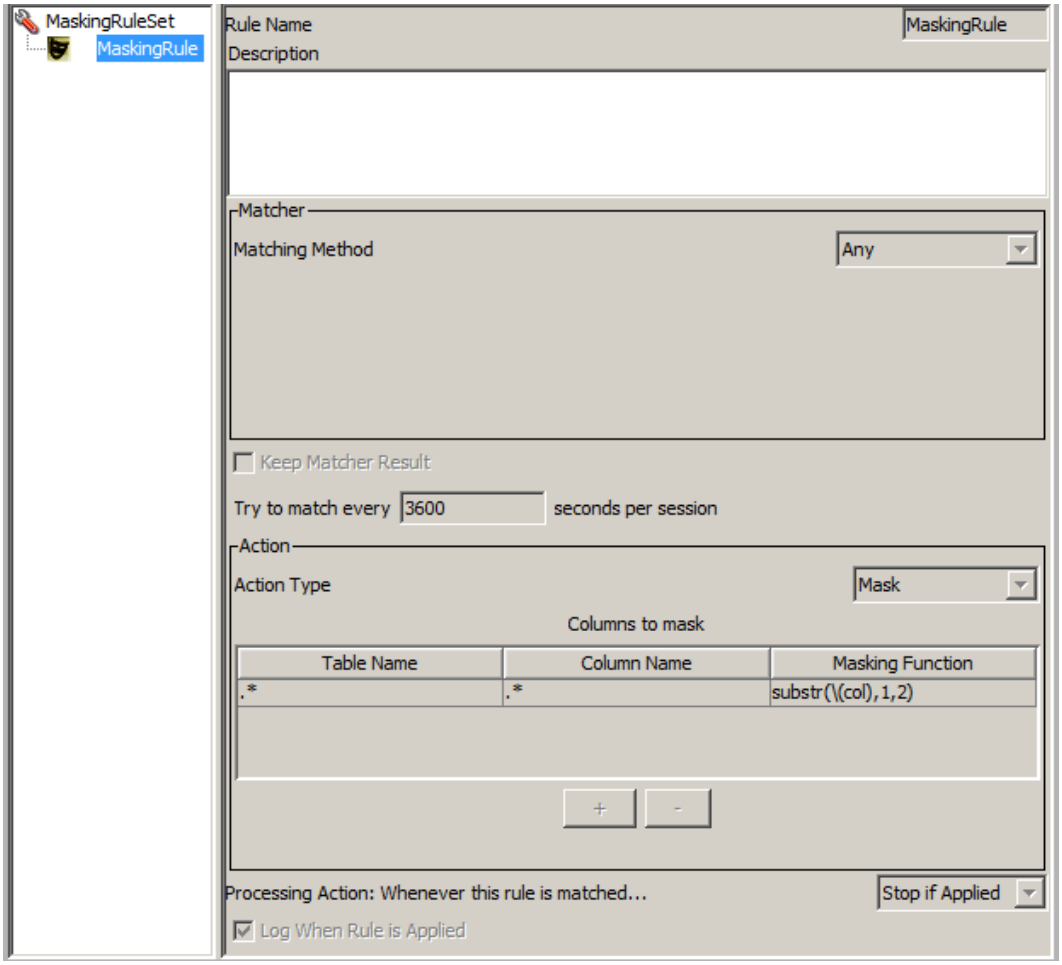
Note the **DDM Database Name** and **DBA Username** values. You use the database connection parameters when you run the Security Rule Set Simulator.

You add the Dynamic Data Masking service for the database in the Management Console and create a connection rule. The connection rule directs requests to the database to a security rule set. The All Incoming Connections rule matcher creates a match for all connections to the database. The Use Rule Set rule action directs all connections to the rule set that you specify in the Rule Set Name parameter. The following image shows the connection rule parameters:



Note the **Rule Set Name** value. You use the value to name the security rule set.

You create a masking security rule that uses the column alias to truncate all values in the database. The Any rule matcher matches all requests that go through the rule. The mask rule action uses a masking function to mask the sensitive data. The following image shows the rule set and the masking rule parameters:



In the Rule Editor, you highlight the masking rule in the rule tree and open the Security Rule Set Simulator.

In the Security Rule Set Simulator you enter the following information:

DDM Database Name

The database name that you used to create the database node in the Management Console.

DBA Username

The database user name that you used to connect to the database in the Management Console.

DBA Password

The password for the database user.

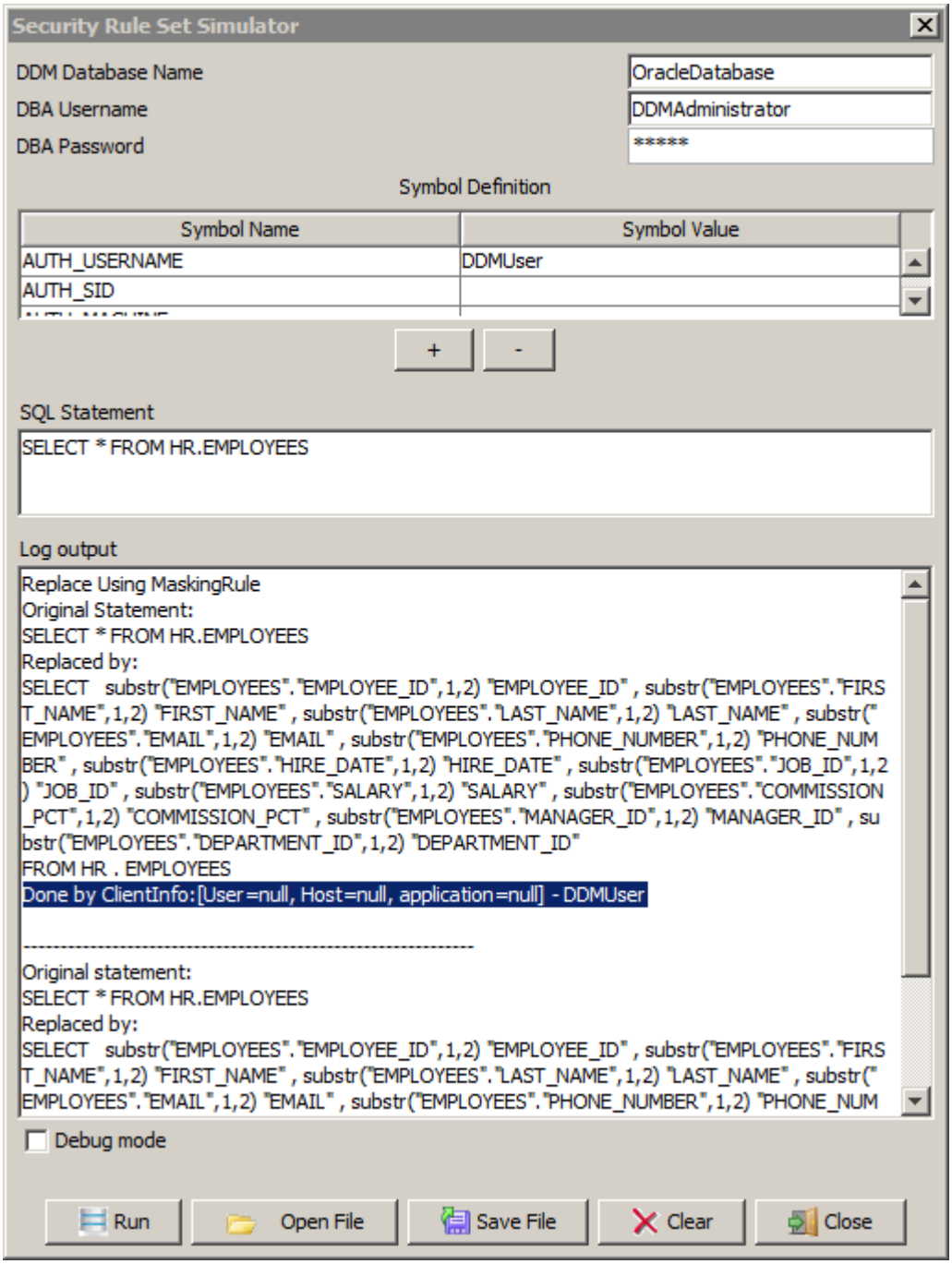
AUTH_USERNAME Symbol Value

The database user name that you want to use to test the masking rule. If you do not enter a database user name, the simulator uses the Dynamic Data Masking administrator user name and password that you used to connect to the database in the Management Console.

SQL Statement

The SQL request that you want to use to test the masking rule.

Click **Run** to run the simulator and test the rule. The following image shows the Security Rule Set Simulator parameters and the simulator log output:



The log output shows that the simulator tested the rule as the user that you entered for the AUTH_USERNAME symbol.

The following text shows the complete simulator log output:

```
Replace Using MaskingRule
Original Statement:
SELECT * FROM HR.EMPLOYEES
Replaced by:
SELECT   substr("EMPLOYEES"."EMPLOYEE_ID",1,2) "EMPLOYEE_ID" ,
```

```
substr("EMPLOYEES"."FIRST_NAME",1,2) "FIRST_NAME" , substr("EMPLOYEES"."LAST_NAME",1,2)
"LAST_NAME" , substr("EMPLOYEES"."EMAIL",1,2) "EMAIL" ,
substr("EMPLOYEES"."PHONE_NUMBER",1,2) "PHONE_NUMBER" , substr("EMPLOYEES"."HIRE_DATE",
1,2) "HIRE_DATE" , substr("EMPLOYEES"."JOB_ID",1,2) "JOB_ID" ,
substr("EMPLOYEES"."SALARY",1,2) "SALARY" , substr("EMPLOYEES"."COMMISSION_PCT",1,2)
"COMMISSION_PCT" , substr("EMPLOYEES"."MANAGER_ID",1,2) "MANAGER_ID" ,
substr("EMPLOYEES"."DEPARTMENT_ID",1,2) "DEPARTMENT_ID"
FROM HR . EMPLOYEES
Done by ClientInfo:[User=null, Host=null, application=null] - DDMUser

------------------------------------------------------------
Original statement:
SELECT * FROM HR.EMPLOYEES
Replaced by:
SELECT   substr("EMPLOYEES"."EMPLOYEE_ID",1,2) "EMPLOYEE_ID" ,
substr("EMPLOYEES"."FIRST_NAME",1,2) "FIRST_NAME" , substr("EMPLOYEES"."LAST_NAME",1,2)
"LAST_NAME" , substr("EMPLOYEES"."EMAIL",1,2) "EMAIL" ,
substr("EMPLOYEES"."PHONE_NUMBER",1,2) "PHONE_NUMBER" , substr("EMPLOYEES"."HIRE_DATE",
1,2) "HIRE_DATE" , substr("EMPLOYEES"."JOB_ID",1,2) "JOB_ID" ,
substr("EMPLOYEES"."SALARY",1,2) "SALARY" , substr("EMPLOYEES"."COMMISSION_PCT",1,2)
"COMMISSION_PCT" , substr("EMPLOYEES"."MANAGER_ID",1,2) "MANAGER_ID" ,
substr("EMPLOYEES"."DEPARTMENT_ID",1,2) "DEPARTMENT_ID"
FROM HR . EMPLOYEES
```

You select **Debug Mode** and run the simulator to view the detailed log output. The detailed log output shows additional connection and Dynamic Data Masking Server information. The following text shows the debug log output:

```
Database set to: OracleDatabase
FolderAction: trying: MaskingRule for: []
MaskingRule.match: matcher: AnyMatcher h: []
Applying rule: MaskingRule
MaskingAction.execute: SELECT * FROM HR.EMPLOYEES
OracleDatabase: Getting connection for: DDMAdministrator
OracleDatabase.buildURL: Finished being built URLjdbc:informatica:oracle://hr-
server;ServiceName=Oracle11g
OracleDatabase: Is Setting schema: true
Try and impersonate using: user=DDMAdministrator and null catalog.
Starting to impersonate user: DDMAdministrator -- Null catalog
Adding for execution SQL statement: ALTER SESSION SET CURRENT_SCHEMA = DDMAdministrator
Here are the 1 return codes after executing the statement list:-
0:
Statement returned with number of rows:
0
Completed impersonation to user: DDMAdministrator
Completed successfully impersonating user: DDMAdministrator -- Null catalog
MaskingAction: Alias set is: ["EMPLOYEES"]
OracleDatabase: Impersonating
OracleDatabase: Getting connection for admin user = DDMAdministrator
OracleDatabase: Getting connection for ddm admin user and password
OracleDatabaseUser = DDMAdministrator ; Password = GCMNFIENGEBPNCBI
OracleDatabaseAbout to obtain non-pooled connection.
OracleDatabase.buildURL: Finished being built URLjdbc:informatica:oracle://hr-
server;ServiceName=Oracle11g
OracleDatabaseObtained connection successfully.
OracleDatabase: Using ddm admin user DDMAdministrator, testing if impersonating
supported for: client user = DDMUser ;client catalog = null ;client path = null
OracleDatabase: Using ddm admin user DDMAdministrator, impersonating supported for:
client user = DDMUser ;client catalog = null ;client path = null
Try and impersonate using: user=DDMUser and null catalog.
Starting to impersonate user: DDMUser -- Null catalog
Adding for execution SQL statement: ALTER SESSION SET CURRENT_SCHEMA = DDMUser
Here are the 1 return codes after executing the statement list:-
0:
Statement returned with number of rows:
0
Completed impersonation to user: DDMUser
Completed successfully impersonating user: DDMUser -- Null catalog
MaskingAction: Rewriting start to:  SELECT   "EMPLOYEES" . "EMPLOYEE_ID" , "EMPLOYEES" .
"FIRST_NAME" , "EMPLOYEES" . "LAST_NAME" , "EMPLOYEES" . "EMAIL" , "EMPLOYEES" .
```

```
"PHONE_NUMBER" , "EMPLOYEES" . "HIRE_DATE" , "EMPLOYEES" . "JOB_ID" , "EMPLOYEES" .
"SALARY" , "EMPLOYEES" . "COMMISSION_PCT" , "EMPLOYEES" . "MANAGER_ID" , "EMPLOYEES" .
"DEPARTMENT_ID"
FROM HR . EMPLOYEES
Replace Using MaskingRule
Original Statement:
SELECT * FROM HR.EMPLOYEES
Replaced by:
SELECT   substr("EMPLOYEES"."EMPLOYEE_ID",1,2) "EMPLOYEE_ID" ,
substr("EMPLOYEES"."FIRST_NAME",1,2) "FIRST_NAME" , substr("EMPLOYEES"."LAST_NAME",1,2)
"LAST_NAME" , substr("EMPLOYEES"."EMAIL",1,2) "EMAIL" ,
substr("EMPLOYEES"."PHONE_NUMBER",1,2) "PHONE_NUMBER" , substr("EMPLOYEES"."HIRE_DATE",
1,2) "HIRE_DATE" , substr("EMPLOYEES"."JOB_ID",1,2) "JOB_ID" ,
substr("EMPLOYEES"."SALARY",1,2) "SALARY" , substr("EMPLOYEES"."COMMISSION_PCT",1,2)
"COMMISSION_PCT" , substr("EMPLOYEES"."MANAGER_ID",1,2) "MANAGER_ID" ,
substr("EMPLOYEES"."DEPARTMENT_ID",1,2) "DEPARTMENT_ID"
FROM HR . EMPLOYEES
Done by ClientInfo:[User=null, Host=null, application=null] - DDMUser

FolderAction: rule matched: true
FolderAction: rule applied: true
DNRStatementProcessor.getStatementHandler: rule match: [] on: SELECT * FROM HR.EMPLOYEES
-------------------------------------------------------------
Original statement:
SELECT * FROM HR.EMPLOYEES
Replaced by:
SELECT   substr("EMPLOYEES"."EMPLOYEE_ID",1,2) "EMPLOYEE_ID" ,
substr("EMPLOYEES"."FIRST_NAME",1,2) "FIRST_NAME" , substr("EMPLOYEES"."LAST_NAME",1,2)
"LAST_NAME" , substr("EMPLOYEES"."EMAIL",1,2) "EMAIL" ,
substr("EMPLOYEES"."PHONE_NUMBER",1,2) "PHONE_NUMBER" , substr("EMPLOYEES"."HIRE_DATE",
1,2) "HIRE_DATE" , substr("EMPLOYEES"."JOB_ID",1,2) "JOB_ID" ,
substr("EMPLOYEES"."SALARY",1,2) "SALARY" , substr("EMPLOYEES"."COMMISSION_PCT",1,2)
"COMMISSION_PCT" , substr("EMPLOYEES"."MANAGER_ID",1,2) "MANAGER_ID" ,
substr("EMPLOYEES"."DEPARTMENT_ID",1,2) "DEPARTMENT_ID"
FROM HR . EMPLOYEES
```

You view the log output to verify that you defined the rule correctly. You can save the Security Rule Set Simulator parameters and log output to a file. You can load the saved parameters and log output to a new instance of the Security Rule Set Simulator.

The log output shows that you created the rule properly. You exit the Security Rule Set Simulator and save the rule.

CHAPTER 6

Masking Functions

This chapter includes the following topics:

- Masking Functions Overview, 83
- Column and Alias Regular Expression Symbols, 84
- Scrambling Functions, 85
- Reverse Masking Function, 85
- Column Level Masking, 86

Masking Functions Overview

A masking function is a component of the security rule Mask action. Masking functions define how sensitive information is returned to an unauthorized user.

Masking functions mask, scramble, and randomize data.

Masking Function Examples

A masking function defines how the Rule Engine masks data. The Rule Engine applies masking functions to incoming SELECT SQL statements that an unauthorized client sends to the database. Dynamic Data Masking rewrites the SQL statement. When the database receives the request, the database sends masked data to the client.

To create masking functions, use regular expressions and database management system SQL functions on the SELECT LIST columns.

The following table describes examples of masking functions:

Masking Function	Description
"	Hides the contents of the table column.
`xxxx`	Replaces the returned value inside ' '.
`to_date('YYYY-MM-DD')`	Applies a date to the masking function.

Masking Function	Description
`customer_codecase customer_code when 4 then 'confidential vip customer' when 2 then 'confidential vip customer' else customer_name end`	Applies row level security and masks VIP customers identified by `customer_code = 4`.
`decode ( substr ( p.bankaccno , 1 , 4 ) , 4500 , 450014265523 , 5314 , 531431925755 , 3210 , 321014760512 , 312331322 ) bankaccno`	Maintains the first four digits of account numbers and replaces the rest of the numbers with masked values.

Column and Alias Regular Expression Symbols

You can parse column names with the regular expression symbols `\(col)` and `\(alias)`.

Column Alias \(alias)

Use the column alias, `\(alias)`, for columns in multiple tables to prevent ambiguity.

Using \(alias) Example

You want to mask a column, `deptno`, in the `emp` and `dept` tables. Use the column alias, `\(alias)`, with the masking function `substr(deptno, 1,2)||'**'`.

The masking function `mask_function(\(alias)deptno)` changes `select e.deptno, d.deptno from emp e, dept d` to `select substr("E".deptno,1,2)||'**' "DEPTNO" , substr("D".deptno,1,2)||'**' "DEPTNO" from "EMP" "E" , "DEPT" "D"`.

Create a rule that searches for tables with the column name `deptno`. When a match occurs, the Rule Engine applies the masking function to replace the data in the `deptno` column.

In the security rule, use the with the Mask rule action. The table name is `.*`, the column name is `deptno`, and the masking function is `substr(deptno, 1,2)||'**'`

Column Alias \(col)

Use the column alias, `\(col)`, to mask columns with similar names.

When you use the column alias, the Rule Engine rewrites each identified column name with the respective name of the column. For example, the Rule Engine rewrites `select credit from customer` to `select mask_function(credit) from customer`.

Using the Column Alias Example

For example, you want to mask credit card numbers. Use the column alias to mask columns that start with `credit.*`.

Create a security rule that uses the Mask action and searches all tables for the column name `credit.*`. Define the table name as `.*`, the column name as `CREDIT.*`, and the masking function as `nvl2\(col), ('****-****-****-'||substr(CREDIT_CARD,length(trim(CREDIT_CARD))-3)),'')`. If the Rule Engine matches the column name, it rewrites the statement as `nvl2\(col),('****-****-****-'||substr(CREDIT_CARD,length(trim(CREDIT_CARD))-3)),'')`.

Scrambling Functions

A scrambling function is a masking technique that rearranges values in the requested data. Scrambling functions use Oracle database syntax and are created by using the security rule Mask action.

The following table describes examples of scrambling functions.

Scrambling Function	Description
`translate(reverse(\(col)), '1234567890', '97865301')`	Scrambles the locations of the digits.
`max(\(col)) over(order by empno rows between 1 and preceding and 1 following)`	Provides list scrambling. The Rule Engine replaces the name with the letter value of the preceding and subsequent records.
`nvl( lead(ename) over (order by ename), first_value(ename) over ( order by ename)`	Offsets the address by a single row. If the value is null, the Rule Engine assigns the `first_value` of `ename` found in the `emp` table after sorting by `ename`.

Reverse Masking Function

The reverse masking function rewrites the `WHERE` clause of masked request statements to return the actual value stored in the database.

You can use the reverse masking function to prevent requests that use masked data. For example, you send a request to the database for an account number and the database returns a masked value. You send another request to the database. The request references the masked account number and the database returns an error message. The reverse masking function replaces the masked `WHERE` clause with actual data and you do not receive an error message.

Reverse Masking Example

The following example shows how to create a reverse masking function. The reverse masking function uses the Search and Replace security rule action to replace the `WHERE` clause in a request statement.

To create reverse masking functions, use nested security rules. The top-level folder uses the Any matcher, the Folder action, and the Continue processing action.

Create two security rules. The first rule performs reverse masking. If the request statement does not match the criteria for the first rule, the Rule Engine applies the second rule. The second rule masks the account information.

In the top level folder, create a security rule called Unmask. The Unmask rule rewrites the `WHERE` clause with the reverse masked value by using regular expression. Configure the security rule to use the Any matcher and the Search and Replace rule action. In the Search Text field, enter `ACCOUNT\s*=\s*(\:\w+)`. Select the Regular Expression identification method. In the Replacement String field, enter `ACCOUNT = substr(\(1), 1,6) || translate('3412', '1234', substr(\(1),7)) || subst.(\(1),11).`

Select the Continue processing action so that the Rule Engine continues to the next rule in the rule tree.

In the top level folder, create a security rule called Account Masking. Select the Any matcher and the Mask rule action.

In the Table name field, enter `.*` for all tables. In the Column Name field, enter `ACCOUNT`. In the Masking Function field, enter `substr(\(col),1,6) || translate('1234', '3412', substr(\(col),7)) || substr(\(col),11)`.

Select the Stop if Applied processing action so that the Rule Engine does not continue to the next rule in the rule tree.

Column Level Masking

Column level masking prevents users from seeing data contained in a specific column and table.

In this example, the masking function shows the `sal` column in the `emp` table for employees that do not have the name King. To apply column level security, add the following masking function to the stored procedure in the database:

```
    create or replace function mask_sal(sal in number, name in varchar) return integer as
begin
if name = 'KING'
then
return -1;
end if;
return sal;
end;
```

Next, create a security rule masking function which causes the Rule Engine to mask select statements containing the `emp` table and `sal` column by using the `mask_sal(\(col). \(alias)ename)` masking function. Specify the Table name as `emp`, the Column Name as `sal`, and the Masking Function as `mask_sal(\(col). \(alias)ename)`.

APPENDIX A

XML Functions Reference

This appendix includes the following topics:

- XML Functions Overview, 87
- IBM DB2, 87
- Microsoft SQL Server, 88
- Oracle, 89
- Sybase, 90

XML Functions Overview

Dynamic Data Masking parses and masks SQL statements that contain XML functions. The XML functions that you can use with Dynamic Data Masking vary based on the database type. Use the tables below to determine which XML functions Dynamic Data Masking parses and masks for each database type.

IBM DB2

The following table lists the XML functions you can use with an IBM DB2 database:

XML Function	Parser	Masking Action	Comments
XMLATTRIBUTES	Yes	Yes	For masking, the expression must be followed by an AS alias clause.
XMLELEMENT	Yes	Yes	For masking, the XMLATTRIBUTES function is an optional clause in the XMLELEMENT function.
XMLEXISTS	Yes	Yes	For masking, the expression must be followed by an AS alias clause.
XMLQUERY	Yes	Yes	For masking, the expression must be followed by an AS alias clause.
XMLTABLE	Yes	No	

XML Function	Parser	Masking Action	Comments
XMLAGG	Yes	Yes	
XMLCAST	Yes	Yes	For masking, you must verify that masking does not violate the requirement of the function. The type of the cast operand or the specified datatype must be XML.
XMLCOMMENT	Yes	Yes	For masking, the expression cannot be a quoted string.
XMLCONCAT	Yes	Yes	
XMLDOCUMENT	Yes	Yes	
XMLFOREST	Yes	Yes	For masking, the expression must be followed by an AS alias clause.
XMLNAMESPACES	Yes	Yes	
XMLPI	Yes	Yes	For masking, you must verify that the optional value_expr object resolves to a string.
XMLROW	Yes	Yes	For masking, the expression must be followed by an AS alias clause.
XMLTEXT	Yes	N/A	
XMLXSROBJECTID	Yes	No	
XMLPARSE	Yes	Yes	For masking, the expression cannot be a quoted string.
XMLSERIALIZE	Yes	Yes	

Microsoft SQL Server

The following table lists the XML functions you can use with a Microsoft SQL Server database:

XML Function	Parser	Masking Action
FOR XML Clause	Yes	N/A
OPEN XML	Yes	No
XQUERY.VALUE	Yes	N/A
XQUERY.EXISTS	Yes	N/A

Oracle

The following table lists the XML functions you can use with an Oracle database:

XML Function	Parser	Masking Action	Comments
XMLATTRIBUTES	Yes	Yes	For masking, the expression must be followed by an AS alias clause.
XMLELEMENT	Yes	Yes	
XMLEXISTS	Yes	Yes	For masking, the expression must be followed by an AS alias clause.
XMLQUERY	Yes	Yes	For masking, the expression must be followed by an AS alias clause.
XMLTABLE	Yes	No	
XMLAGG	Yes	Yes	
XMLCAST	Yes	Yes	For masking, you must verify that the masked expression can be casted to the specified datatype.
XMLCOLATTVAL	Yes	Yes	For masking, the expression must be followed by an AS alias clause.
XMLCOMMENT	Yes	Yes	For masking, the expression cannot be a quoted string.
XMLCONCAT	Yes	Yes	
XMLFOREST	Yes	Yes	For masking, the expression must be followed by an AS alias clause.
XMLNAMESPACES	Yes	Yes	
XMLPI	Yes	Yes	For masking, the expression cannot be a quoted string.
XMLSEQUENCE	Yes	Yes	
XMLPARSE	Yes	Yes	For masking, the expression cannot be a quoted string.
XMLSERIALIZE	Yes	Yes	

Sybase

The following table lists the XML functions you can use with a Sybase database:

XML Function	Parser	Masking Action	Comments
XMLATTRIBUTES	Yes	Yes	For masking, the expression must be followed by an AS alias clause.
XMLELEMENT	Yes	Yes	For masking, the XMLATTRIBUTES function is an optional clause in the XMLELEMENT function.
XMLEXISTS	Yes	Yes	For masking, the expression must be followed by an AS alias clause.
XMLTABLE	Yes	No	
XMLAGG	Yes	Yes	
XMLCOMMENT	Yes	Yes	For masking, the expression cannot be a quoted string.
XMLCONCAT	Yes	Yes	
XMLPI	Yes	Yes	For masking, you must verify that the optional value_expr object resolves to a string.
XMLPARSE	Yes	Yes	For masking, the expression cannot be a quoted string.
XMLSERIALIZE	Yes	Yes	

APPENDIX B

Glossary

All Incoming Connections matcher

A matcher that applies the rule matching action to all incoming connections, regardless of target database, listener port, or application information.

Any matcher

Security rule matcher that applies the rule action to all incoming SQL requests.

appender

Management Console tree node that uses log4j classes to define the output format of log information.

Block Statement rule action

Security rule action that applies the security processing action, but does not apply a rule action.

Client/Application Information matcher

A matcher that defines the client or application information that the Rule Engine uses to identify a connection rule match.

connection rule

A rule that defines the criteria that the Rule Engine uses to identify the target database for the request. A connection rule consists of a matcher and an action that identify and route a connection request from an application.

connection rule matcher

A matcher that defines the matching method that the Rule Engine applies to identify a database request to be routed.

connection rule tree

A rule tree that defines the hierarchy that the Rule Engine uses to processes connection rules. The connection rule tree contains all the connection rules that you define for the target databases. The Rule Engine processes the first rule or rule folder in the connection rule tree and stops when there is a stop processing action.

Continue processing action

Processing action that causes the Rule Engine to continue to the next rule in the rule tree. You must assign the Continue processing action to rule folders so that the Rule Engine applies the rules within the folder.

Current Target Database matcher

A matcher that defines the target database that the Rule Engine uses to identify a connection rule match.

data masking

The process of disguising data to prevent access to sensitive information.

Define Symbol rule action

Security rule action that replaces strings or variables with the symbol that you specify.

Direct action

Connection rule action that connects applications and clients directly to the target database. When the Rule Engine applies the Direct action, the connection bypasses the Dynamic Data Masking service.

domain

A virtual node in the Management Console tree that you use to group other nodes. The Management Console contains a default root domain. You can use domains to create a visual representation of the structure of the databases within an organization.

Dynamic Data Masking Server

A server that provides services and resources to intercept database requests and perform data masking tasks.

Dynamic Data Masking service

A service that routes SQL queries to Oracle, DB2, Microsoft SQL Server, and Informix databases.

Folder rule action

A rule action that creates a rule folder. Use the folder action to nest and group rules.

From Clause Object matcher

Security rule matcher that identifies a request statement that uses an object or an alias in the FROM clause. Use the From Clause Object matcher to identify a reference to a view that does not appear in the execution plan.

Incoming Dynamic Data Masking Listener Port matcher

Defines the listener port that the Rule Engine uses to identify a connection rule match.

Informatica Dynamic Data Masking

A data security software that operates between an application and a database to prevent unauthorized access to sensitive information.

Java matcher

Security rule matcher that uses a Java class as a matcher.

Java rule action

Security rule action that uses a custom Java class as a matching action.

logger

Management Console tree node that uses Apache log4j to create a log of events.

log level

The severity of an event that you want to log. Define the Dynamic Data Masking Server log level to specify the events that you want to log.

Log Message rule action

Security rule action that uses a logger to log security rule event information.

Management Console

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.

masking function

Defines the SQL function that the Rule Engine applies to rewrite the SQL SELECT statement.

Mask rule action

Security rule action that rewrites the SELECT statement in the SQL request. The Mask rule action uses a masking function to define how the Rule Engine rewrites the statement request.

matcher

Defines the criteria that the Rule Engine uses to identify a match. The matcher defines the matching method used for incoming connections.

Nothing rule action

A rule action that applies the processing action, but does not apply a rule action.

Parser matcher

Security rule matcher that identifies requests that Dynamic Data Masking cannot parse.

personally identifiable information (PII)

Information that uniquely identifies an individual, such as credit card information, social security number, birthday, and name.

PL/SQL Function matcher

Security rule matcher that defines a PL/SQL function name and searches for specific parameter values in the PL/SQL code.

processing action

Defines the action that the Rule Engine applies to the request after the Rule Engine applies the rule. The processing action manages how the Rule Engine processes the request through the rule tree.

Refuse action

Connection rule action that defines the access that an application has to the database. The application receives an error that indicates that the database refuses the connection.

Replace Table action

A security rule action that replaces the FROM clause of the SQL request to remove rows that contain sensitive data from the result set.

Rewrite rule action

Security rule action that replaces the SQL statement with an alternate statement.

rule

The conditions and actions that you want to apply to a request. A rule can be a security rule or a connection rule. You can create an individual rule or create a rule as part of a rule folder. A rule consists of a matcher, action, and processing action

rule action

Defines the action that the Rule Engine applies to incoming connections. The rule action determines where the Rule Engine routes the connection.

Rule Engine

A Dynamic Data Masking Server component that 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 Management Console.

rule folder

An organizational tool that you use to group conditional rules. The Rule Engine processes the contents of a rule folder hierarchically. A connection rule folder contains connection rules. A security rule folder contains security rules.

rule tree

An organizational tool that defines the organizational structure of rules and determines the order in which the Rule Engine processes rules.

Search and Replace rule action

Security rule action that searches for specific text within a statement request and replaces it with the text that you define in the replacement string.

security rule

A rule that defines the criteria that the Rule Engine uses to parse and alter the SQL statement request. A security rule consists of a matcher and action that you define to identify and mask an SQL request.

security rule set

A tree node in the Management Console tree that contains references to one or more security rules. You must create security rule sets to secure data by using security rule actions. The Rule Engine processes the SQL statement through the rule set until the Rule Engine encounters a Stop processing action.

security rule tree

Each security rule set has an individual security rule tree. The security rule tree defines the order in which the Rule Engine processes security rules. The security rule tree contains the security rules for a particular

security rule set. The Rule Engine processes the first rule or rule folder in the security rule tree and stops when there is a stop processing action.

Server Control

A command line program that you can use to configure and manage Dynamic Data Masking Server properties. Server Control is one of the management components of Dynamic Data Masking. It is installed with the Dynamic Data Masking Server.

SQL Syntax matcher

Security rule matcher that compares the incoming request statement to the SQL statement that you define within the matcher.

Stop if Applied processing action

Processing action. If the connection or security rule is applied, the Rule Engine does not continue to the next rule in the rule tree.

Stop if Matched processing action

Processing action. In a connection rule, if the rule is matched, the Rule Engine does not process the rule action. In a security rule, the Rule Engine does not continue to process the request.

Switch to Database rule action

Connection rule action that forwards connections to the specified database.

Symbol matcher

Security rule matcher that identifies and stores client and database variable values.

system logger

Pre-defined logger node in the Management Console tree that logs Dynamic Data Masking Server, service, and rule events.

Text matcher

Security rule matcher that defines the statement string text that the Rule Engine uses to match a request statement.

Time of Day matcher

Security rule matcher that defines a time frame during which the Rule Engine matches statements.

Use Rule Set action

Connection rule action that applies the specified security rule set to the SQL statement request.

INDEX

A

action
- Block Statement 56
- connection rule 31
- Define Symbol 57
- Folder 58
- Java 58
- Log Message 59
- Mask 60
- Nothing 63
- PL/SQL Function 64
- Replace Table 65
- Rewrite 67
- Search and Replace 67
- security rule 56

All Incoming Connections matcher 28
Any matcher 37

B

backslash 53
batch
- masking 62

Block Statement action 56

C

capturing group 54
character class 53
Check Database Connection matcher 28
Check Database DSN matcher 28
Check Database URL matcher 29
Check Property matcher 29
Check Server Name matcher 29
Client Information matcher 30
Client/Application Information matcher 30
connection rule
- creating 34
- export 35
- import 35
- matcher 27
- overview 26
- parameters 26
- processing action 34

connection rule action 31
connection rule folder
- creating 21

connection rule tree 19
connection rules
- Check Database Connection 28
- Check Server Name matcher 29

Current Target Database matcher 31

D

database
- target database 31

Define Symbol action 57
domains
- management 17

Dynamic Data Masking
- architecture 11
- components 11
- overview 10
- process 13

Dynamic Data Masking service
- Data Vault 11
- Hive 11
- IBM DB2 11
- Informix 11
- Microsoft SQL Server 11
- Oracle 11
- Sybase 11
- Teradata 11

E

escapes 53
example
- keep original number of rows 61
- Mask action 61
- PL/SQL Function matcher 43

export
- connection rule 35
- security rule 69
- security rule set 70

F

Folder action 58
From Clause Object matcher 38

G

global variables 49

I

IBM DB2
- XML functions 87

import
- connection rule 35
- security rule 69
- security rule set 71

Incoming DDM Listener Port matcher 31

J

Java action 58
Java matcher 38

L

latency 20
line terminator 54
listener port 31
Log Message action 59

M

Management Console
  logging in 16
  menu 14
  navigation 14
  overview 13
  tree 15
mask
  batch 62
Mask action
  example 61
  XML functions 87
masking
  example 61
masking function
  column level masking 86
  example 83
  overview 83
  reverse 85
matcher
  All Incoming Connections 28
  Any 37
  Check Database Connection 28
  Check Database DSN 28
  Check Database URL 29
  Check Property 29
  Check Server Name 29
  Client Information 30
  Client/Application Information 30
  Current Target Database 31
  From Clause Object 38
  Incoming DDM Listener Port 31
  Java 38
  Parser 40
  PL/SQL Function 41
  PL/SQL Function example 43
  SQL Syntax 49
  Symbol 49
  Text 51
  Time of Day 56
matching method 27
Microsoft SQL Server
  XML functions 88

N

Nothing action 63

O

Oracle
  XML functions 89

P

Parser matcher 40
PL/SQL Function action 64
PL/SQL Function matcher
  example 43
processing action
  connection rule 34
  security rule 68

R

regular expression
  alias 84
  backslash 53
  character class 53
  col 84
  escapes 53
  line terminator 54
regular expression symbol
  alias 84
  col 84
Replace Table action
  example 66
  symbols 65
reverse masking function 85
Rewrite action 67
rule
  creating 34, 68
  export 24
  import 24
rule action
  Block Statement 56
  connection rule 31
  Define Symbol 57
  Folder 58
  Java 58
  Log Message 59
  Mask 60
  Nothing 63
  PL/SQL Function 64
  Replace Table 65
  Rewrite 67
  Search and Replace 67
  security rule 56
Rule Engine 11
rule folder
  deleting 24
rule set
  security rule set 20
rule tree
  efficiency 20
rules
  components 18
  deleting 24
  disabling 23
  editing 23
  enabling 23
  example 24
  folders 21
  management 23

rules *(continued)*
matcher 18
overview 18
processing action 18
rule action 18
rule tree 19
updating 23

S

scrambling function 85
Search and Replace action 67
security rule
action 56
creating 68
export 69
import 69
matcher 27, 37
overview 36
parameters 37
processing action 68
security rule folder
creating 22
security rule set
creating 20
export 70
import 71
Security Rule Set Simulator
example 76
overview 72
parameters 73
simulating a rule 75
security rule tree 19
security rules
Security Rule Set Simulator 72
Security Rule Set Simulator example 76
Security Rule Set Simulator parameters 73
simulating a rule 75
Server Control 11
SQL statements
rewrite 49
SQL Syntax matcher 49
Sybase
XML functions 90
Symbol matcher 49
symbols
Replace Table action 65
syntax matcher 20

T

tagged expressions 54
text matcher
regular expression 52
wildcard 52
Text matcher 51
Time of Day matcher 56

V

variable bind 49
variables
global 49

W

wildcard 52

X

XML functions
IBM DB2 87
masking 87
Microsoft SQL Server 88
Oracle 89
parser 87
Sybase 90