This software and documentation contain proprietary information of Informatica Corporation 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 Corporation. This Software is 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(c)(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 and Siperian are trademarks or registered trademarks of Informatica Corporation 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.

This product includes software developed by the Apache Software Foundation (http://www.apache.org/), and other software which is licensed under the Apache License, Version 2.0 (the "License"). You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0. Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

This product includes software which is licensed under 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.

This product includes software which is licensed under the CDDL (the "License"). You may obtain a copy of the License at http://www.sun.com/cddl/cddl.html. The materials 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. See the License for the specific language governing permissions and limitations under the License.

This product includes software which is licensed under the BSD License (the "License"). You may obtain a copy of the License at http://www.opensource.org/licenses/bsd-license.php. 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. See the License for the specific language governing permissions and limitations under the License.

This product includes software Copyright (c) 2003-2008, Terence Parr, all rights reserved which is licensed under the BSD License (the "License"). You may obtain a copy of the License at http://www.amtr.org/license.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. See the License for the specific language governing permissions and limitations under the License.

This product includes software Copyright (c) 2000 - 2009 The Legion Of The Bouncy Castle (http://www.bouncycastle.org) which is licensed under a form of the MIT License (the "License"). You may obtain a copy of the License at http://www.bouncycastle.org/license.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. See the License for the specific language governing permissions and limitations under the License.

DISCLAIMER: Informatica Corporation provides this documentation "as is" without warranty of any kind, either express or implied, including, but not limited to, the implied warranties of non-infringement, merchantability, or use for a particular purpose. Informatica Corporation 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") may include 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.

Part Number: MDM-SIF-91000-0001
# Table of Contents

**Preface**  
Learning About Informatica MDM Hub. ................................................................. v  
Informatica Resources. ......................................................................................... vii  
  Informatica Customer Portal. ........................................................................ vii  
  Informatica Documentation. .......................................................................... vii  
  Informatica Web Site. ................................................................................. vii  
  Informatica How-To Library. ....................................................................... vii  
  Informatica Knowledge Base. ................................................................. vii  
  Informatica Multimedia Knowledge Base. ........................................... vii  
  Informatica Global Customer Support. ................................................ viii  

**Chapter 1: Introduction** .............................................................................. 1  
Overview of Services Integration Framework (SIF). ........................................ 1  
About Informatica MDM Hub and External Applications. .......................... 1  
  How External Applications Interact with Informatica MDM Hub. ............... 2  
  About Real-time Processing. ................................................................. 2  
About Services Integration Framework (SIF). ......................................... 3  
  Benefits of Using SIF. .......................................................................... 4  
  About SiperianClient Library. ............................................................... 5  
About SIF Access Protocols. ................................................................. 6  
  Using Web Services. ........................................................................... 8  
  Using XML Over HTTP. ................................................................. 8  
About Informatica MDM Hub Requests. .................................................. 9  
  How SIF Requests Are Processed. ...................................................... 9  
  Types of SIF Requests. .................................................................... 10  
  Using SIF SDK to Interface with SIF Classes. .................................... 13  

**Chapter 2: Setting Up the SIF SDK** ............................................................ 14  
Overview of Setting Up the SIF SDK. ........................................................ 14  
Before You Begin. ....................................................................................... 14  
Deploying the SIF SDK. ............................................................................ 14  
About SIF API Javadoc. ........................................................................... 15  
Building Web Services. ............................................................................. 16  

**Chapter 3: Using the SIF SDK** ................................................................. 17  
Overview of Using the SIF SDK. ............................................................... 17  
About SIF SDK. ....................................................................................... 17  
  Informatica MDM Hub SIF Manager. ................................................ 18
Chapter 5: SIF API Reference ................................................................. 46

Functional SIF API Listing ................................................................. 46
Reference SIF API Listing ................................................................. 50

AcceptUnmatchedRecordsAsUnique ................................................. 50
AddRelationship .............................................................................. 51
ApplyChangeList .............................................................................. 51
AssignUnmergedRecords ................................................................. 51
Audit ........................................................................................................ 52
Authenticate ........................................................................................ 53
CanUnmergeRecords .......................................................................... 53
Cleanse ............................................................................................... 54
CleansePut .......................................................................................... 54
ClearAssignedUnmergedRecords .................................................... 56
CreateChangeList .............................................................................. 56
Delete ................................................................................................... 56
DeleteRelationship ............................................................................. 57
ExecuteBatchGroup ........................................................................... 58
Get ........................................................................................................ 58
GetAssignedRecords ......................................................................... 59
GetBatchGroupStatus ........................................................................ 60
GetBvt ............................................................................................... 60
GetEntityGraph ................................................................................. 61
GetLookupValue ............................................................................... 61
GetLookupValues .............................................................................. 62
GetMatchedRecords ......................................................................... 62
GetMergeHistory ............................................................................... 63
GetOneHop ........................................................................................ 63
GetOrsList ........................................................................................ 64
GetOrsMetadata ............................................................................... 64
GetSearchResults ............................................................................. 64
GetSiperianObjectCompatibility ..................................................... 65
GetSystemTrustSettings ..................................................................... 65
GetTasks .............................................................................................. 65
GetTrustGraphData ........................................................................... 66
GetTrustScore ........................................................  66
GetUnmergedRecordCount ...............................................  67
Link ...............................................................  67
ListSiperianObjects ....................................................  67
Merge .............................................................  68
MultiMerge ..........................................................  69
PromotePendingXrefs ...................................................  69
Put ...............................................................  70
ReassignRecords ......................................................  71
RegisterUsers ........................................................  72
ResetBatchGroup ......................................................  72
Restore ............................................................  72
SearchHmQuery ......................................................  73
SearchLookupValues ...................................................  73
SearchMatch ........................................................  74
SearchQuery ........................................................  74
SearchRequestBase ....................................................  75
SearchResponseBase ...................................................  76
SetPassword ........................................................  76
SetRecordState .......................................................  76
Tokenize ...........................................................  77
Unlink .............................................................  77
Unmerge ...........................................................  77
UnregisterUsers .......................................................  78
UpdateRelationship ....................................................  79
ValidateChangeList .....................................................  80
ValidateMetadata .....................................................  80

Index ...............................................................  81
Preface

Welcome to the Informatica MDM Hub Services Integration Framework Guide. This guide explains how to use the Services Integration Framework (SIF) to integrate Informatica MDM Hub functionality with your applications and how to create applications using the data provided by Informatica MDM Hub. SIF allows you to integrate Informatica MDM Hub smoothly with your organization’s applications.

Learning About Informatica MDM Hub

What’s New in Informatica MDM Hub

What’s New in Informatica MDM Hub describes the new features in this Informatica MDM Hub release.

Informatica MDM Hub Release Notes

The Informatica MDM Hub Release Notes contain important information about this Informatica MDM Hub release. Installers should read the Informatica MDM Hub Release Notes before installing Informatica MDM Hub.

Informatica MDM Hub Overview

The Informatica MDM Hub Overview introduces Informatica MDM Hub, describes the product architecture, and explains core concepts that users need to understand before using the product. All users should read the Informatica MDM Hub Overview first.

Informatica MDM Hub Installation Guide

The Informatica MDM Hub Installation Guide explains to installers how to set up Informatica MDM Hub, the Hub Store, Cleanse Match Servers, and other components. There is an Informatica MDM Hub Installation Guide for each supported platform.

Informatica MDM Hub Upgrade Guide

The Informatica MDM Hub Upgrade Guide explains to installers how to upgrade a previous Informatica MDM Hub version to the most recent version.
Informatica MDM Hub Cleanse Adapter Guide

The Informatica MDM Hub Cleanse Adapter Guide explains to installers how to configure Informatica MDM Hub to use the supported adapters and cleanse engines.

Informatica MDM Hub Data Steward Guide

The Informatica MDM Hub Data Steward Guide explains to data stewards how to use Informatica Hub tools to consolidate and manage their organization's data. Data stewards should read the Informatica MDM Hub Data Steward Guide after having reading the Informatica MDM Hub Overview.

Informatica MDM Hub Administrator Guide

The Informatica MDM Hub Administrator Guide explains to administrators how to use Informatica MDM Hub tools to build their organization’s data model, configure and execute Informatica MDM Hub data management processes, set up security, provide for external application access to Informatica MDM Hub services, and other customization tasks. Administrators should read the Informatica MDM Hub Administrator Guide after having reading the Informatica MDM Hub Overview.

Informatica MDM Hub Services Integration Framework Guide

The Informatica MDM Hub Services Integration Framework Guide explains to developers how to use the Informatica MDM Hub Services Integration Framework (SIF) to integrate Informatica Hub functionality with their applications, and how to create applications using the data provided by Informatica MDM Hub. SIF allows developers to integrate Informatica MDM Hub smoothly with their organization's applications. Developers should read the Informatica MDM Hub Services Integration Framework Guide after having reading the Informatica MDM Hub Overview.

Informatica MDM Hub Metadata Manager Guide

The Informatica MDM Hub Metadata Manager Guide explains how to use the Informatica MDM Hub Metadata Manager tool to validate their organization’s metadata, promote changes between repositories, import objects into repositories, export repositories, and related tasks.

Informatica MDM Hub Resource Kit Guide

The Informatica MDM Hub Resource Kit Guide explains how to install and use the Informatica Hub Resource Kit, which is a set of utilities, examples, and libraries that assist developers with integrating the Informatica Hub into their applications and workflows. This document also provides a description of the various sample applications that are included with the Resource Kit.

Informatica Training and Materials

Informatica provides live, instructor-based training to help professionals become proficient users as quickly as possible. From initial installation onward, a dedicated team of qualified trainers ensure that an organization’s staff is equipped to take advantage of this powerful platform. To inquire about training classes or to find out where and when the next training session is offered, please visit Informatica’s web site (http://www.informatica.com) or contact Informatica directly.
Informatica Resources

Informatica Customer Portal
As an Informatica customer, you can access the Informatica Customer Portal site at http://mysupport.informatica.com. The site contains product information, user group information, newsletters, access to the Informatica customer support case management system (ATLAS), the Informatica How-To Library, the Informatica Knowledge Base, the Informatica Multimedia Knowledge Base, Informatica Product Documentation, and access to the Informatica user community.

Informatica Documentation
The Informatica Documentation team takes every effort to create accurate, usable documentation. If you have questions, comments, or ideas about this documentation, contact the Informatica Documentation team through email at info_documentation@informatica.com. We will use your feedback to improve our documentation. Let us know if we can contact you regarding your comments.

The Documentation team updates documentation as needed. To get the latest documentation for your product, navigate to Product Documentation from http://mysupport.informatica.com.

Informatica Web Site
You can access the Informatica corporate web site at http://www.informatica.com. The site contains information about Informatica, its background, upcoming events, and sales offices. You will also find product and partner information. The services area of the site includes important information about technical support, training and education, and implementation services.

Informatica How-To Library
As an Informatica customer, you can access the Informatica How-To Library at http://mysupport.informatica.com. The How-To Library is a collection of resources to help you learn more about Informatica products and features. It includes articles and interactive demonstrations that provide solutions to common problems, compare features and behaviors, and guide you through performing specific real-world tasks.

Informatica Knowledge Base
As an Informatica customer, you can access the Informatica Knowledge Base at http://mysupport.informatica.com. Use the Knowledge Base to search for documented solutions to known technical issues about Informatica products. You can also find answers to frequently asked questions, technical white papers, and technical tips. If you have questions, comments, or ideas about the Knowledge Base, contact the Informatica Knowledge Base team through email at KB_Feedback@informatica.com.

Informatica Multimedia Knowledge Base
As an Informatica customer, you can access the Informatica Multimedia Knowledge Base at http://mysupport.informatica.com. The Multimedia Knowledge Base is a collection of instructional multimedia files that help you learn about common concepts and guide you through performing specific tasks. If you have questions, comments, or ideas about the Multimedia Knowledge Base, contact the Informatica Knowledge Base team through email at KB_Feedback@informatica.com.
Informatica Global Customer Support

You can contact a Customer Support Center by telephone or through the Online Support. Online Support requires a user name and password. You can request a user name and password at http://mysupport.informatica.com.

Use the following telephone numbers to contact Informatica Global Customer Support:

<table>
<thead>
<tr>
<th>North America / South America</th>
<th>Europe / Middle East / Africa</th>
<th>Asia / Australia</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Toll Free</strong></td>
<td><strong>Toll Free</strong></td>
<td><strong>Toll Free</strong></td>
</tr>
<tr>
<td>Brazil: 0800 891 0202</td>
<td>France: 00800 4632 4357</td>
<td>Australia: 1 800 151 830</td>
</tr>
<tr>
<td>Mexico: 001 888 209 8853</td>
<td>Germany: 00900 4632 4357</td>
<td>New Zealand: 1 800 151 830</td>
</tr>
<tr>
<td>North America: +1 877 463 2435</td>
<td>Israel: 00800 4632 4357</td>
<td>Singapore: 001 800 4632 4357</td>
</tr>
<tr>
<td><strong>Standard Rate</strong></td>
<td>Italy: 800 915 985</td>
<td>Standard Rate</td>
</tr>
<tr>
<td>North America: +1 650 653 6332</td>
<td>Netherlands: 00800 4632 4357</td>
<td>India: +91 80 4112 5738</td>
</tr>
<tr>
<td></td>
<td>Portugal: 800 208 360</td>
<td><strong>Standard Rate</strong></td>
</tr>
<tr>
<td></td>
<td>Spain: 900 813 166</td>
<td>North America: +1 805 804632</td>
</tr>
<tr>
<td></td>
<td>Switzerland: 00800 4632 4357 or 0800 4632 200</td>
<td>Germany: 01805 702702</td>
</tr>
<tr>
<td></td>
<td>United Kingdom: 00800 4632 4357 or 0800 023 4632</td>
<td>Netherlands: 030 6022 797</td>
</tr>
</tbody>
</table>
Introduction

This chapter includes the following topics:

- Overview of Services Integration Framework (SIF), 1
- About Informatica MDM Hub and External Applications, 1
- About Services Integration Framework (SIF), 3
- About SIF Access Protocols, 6
- About Informatica MDM Hub Requests, 9

Overview of Services Integration Framework (SIF)

This chapter introduces the Services Integration Framework (SIF) and describes the environment for application programs that use SIF to interact with Informatica MDM Hub.

About Informatica MDM Hub and External Applications

Informatica MDM Hub is the best platform available today for deploying MDM solutions across the enterprise. Informatica MDM Hub offers an integrated, model-driven, and flexible enterprise MDM platform that can be used to create and manage all kinds of master data.

Informatica MDM Hub is a server that resides at the center of an enterprise software network. It maintains the best version of the truth for a set of entities (for example, customer records) that may be common to several applications on the network. So, for example, Informatica MDM Hub helps keep track of whether Jane Ann Smithe on the sales lead system represents the same customer as John Anders Smith on the SAP system and, if so, how that customer spells their name.

Informatica MDM Hub uses batch processes and manual intervention when necessary to match new information against its version of the information. It also interacts with Informatica applications (for example, the tools in the Data Steward Workbench), other enterprise software packages, or ad hoc applications on an entity-by-entity basis. All of these applications use a client/server model. Informatica MDM Hub accepts requests and sends responses.

Informatica MDM Hub maintains entity-related information in sets of Operational Record Store (ORS) database tables, which it manages in its internal database management system. Though an enterprise can have more than one ORS, typically it has only one, for example, an ORS for its customer data. The enterprise provides a schema that defines the database tables in the ORS.
How External Applications Interact with Informatica MDM Hub

Request/response interactions with Informatica MDM Hub typically read or update the database tables in the ORS database. Informatica MDM Hub provides a generic set of API requests that are independent of the enterprise’s schema. These requests require the client to specify the database records for Informatica MDM Hub to access. Informatica also provides tools that allow you to construct new ORS-specific request that act on logical entities defined in the schema. For example, a generic request might place given data into a specified database record. An ORS-specific request might identify the same data as a name and an email address and place those fields into a customer record, as defined in the schema.

The ORS-specific requests do not exist within Informatica MDM Hub. Instead, they are methods that use the schema to validate the arguments, and then translate the ORS-specific calls into the requests and responses of generic Hub operations. They allow client programs to operate at a logical level that provides greater type safety than the generic operations.

**RELATED TOPICS:**
- “Using ORS-specific APIs” on page 21

About Real-time Processing

For real-time processing, applications that are external to Informatica MDM Hub invoke Informatica MDM Hub operations using the Services Integration Framework (SIF) interface. SIF provides APIs for various Informatica MDM Hub services, such as reading, cleansing, matching, inserting, and updating records.

In Informatica MDM Hub implementations, real-time processing is used as appropriate. For example, real-time processing can be used to update data in the Hub Store whenever a record is added, updated, or deleted in a
source system. Real-time processing can also be used to handle *incremental data loads* (data loads that occur after the initial data load) into the Hub Store.

The following figure shows the overall real-time process flow for processing data in Informatica MDM Hub.

Differences Between Batch and Real-time (SIF) Processing

Validation rules are evaluated differently for batch and online (API) processing, which can result in different outcomes for downgrading trust levels for a given column or piece of data.

- **Batch Process** — Validation rules are evaluated from *top to bottom*. The process steps through all the rules and only the last applicable rule is applied.

- **SIF API (Put request)** — Validation rules are applied in the *reverse order*. The first rule that meets the validation criteria is the only one applied (evaluation stops at that point).

About Services Integration Framework (SIF)

The Services Integration Framework (SIF) is a services framework (SOA* enabled) that can be configured for Informatica MDM Hub that interfaces with client programs. Logically, it serves as a middle tier in the client/server model. It enables you to implement the request/response interactions using SIF access protocols.

**Note:** Only admin users can access private resources through SIF requests.

SIF Framework encapsulates an API-based access in the form of a toolkit to build Web Services-based access to your Informatica MDM Hub data. SIF contains:

- APIs for fine-grained access to data and objects in the Hub
- SIF SDK, a toolkit for building coarse-grained business services
- Set of tools designed to generate and deploy web services
Benefits of Using SIF

Using SIF for real-time Hub processing provides the following benefits:

- Rapid configuration, deployment, and management of applications integrating Informatica MDM Hub and external systems.
- Addressing integration requirements both at logical (business events and services) as well as granular (data events and services) levels.
- Process relevant events for synchronization and propagation
- Build services for custom data management GUI (for example, Data Manager, Merger Manager)
- Build services for custom applications that utilize Hub data
- Build services for consumption in Portals
- Build unified view services for composite applications that need 360° view of the customer
- Build virtual view services that can be integrated with applications such as Siebel (Virtual business component)
- Reuse business interfaces as new sources are added

The SIF APIs can interact with each other directly. For example Data Services can interact with Data Events, Process Services with Data Services, Data Services with Business Services, Data Events with Process Services, and so on.
Here are some examples for these events services:

<table>
<thead>
<tr>
<th>Process Events</th>
<th>Business Events</th>
<th>Data Services</th>
<th>Business Services</th>
<th>Data Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verify Customer</td>
<td>Profile change sync to 10 sources</td>
<td>Insert/Update record</td>
<td>Get Customer object</td>
<td>Name change in master record</td>
</tr>
<tr>
<td>Approve Customer</td>
<td>Profile propagation to Order system</td>
<td>Create Customer record</td>
<td>Get Customer with relationships</td>
<td>Name change in source record</td>
</tr>
<tr>
<td>Provision Customer</td>
<td>Evaluate impact of new relationship</td>
<td>Get Relationship</td>
<td>Get Customer with transactions</td>
<td>New address change</td>
</tr>
<tr>
<td></td>
<td>Evaluate impact of record transactions</td>
<td>Get Transactions</td>
<td></td>
<td>Relationship terminated</td>
</tr>
</tbody>
</table>

Data events can be generated in the external applications or Informatica MDM Hub. These events are handled by Event driven architecture (EDA) capabilities of the Informatica MDM Hub, which includes Event Capture, Event Processing, Event Filtering and Event Generation.

The services provided in the Informatica MDM Hub can be used by the EDA components as well as external applications for Query and Data Synchronization operations. Additionally existing infrastructure such as an Enterprise Service Bus (ESB) as well as Enterprise Application Integration (EAI) technologies such as Tibco, webMethods (and so on), and Message Oriented Middleware can be used in conjunction with Informatica’s Services Integration Framework.

![Image](image)

About SiperianClient Library

The SiperianClient library and the associated Javadoc are installed with the Informatica MDM Hub Resource Kit in the `resourcekit\sifsdk` folder. The SIF SDK can be used for creating custom web services to be deployed along with Informatica MDM Hub server. These custom web services would typically be built using the SiperianClient library. For an example, see the sample code for building web services using the SIF SDK in the `resourcekit\samples` folders.

Using SiperianClient Library

The process method of SiperianClient provides the basic request/response interaction between the client program and Informatica MDM Hub. It accepts any subclass of `com.siperian.sif.message.SiperianRequest` as an argument.
When it processes the request successfully, it returns a `com.siperian.sif.message.SiperianResponse`, which you must cast to the correct response subclass. Otherwise it throws `SiperianServerException`.

The SiperianClient library is provided to create easy and efficient Java applications. There are two parts to the API:

- The `com.siperian.sif.client` package manages the details of communication with the Informatica MDM Hub Server. SiperianClient can be configured for the desired protocol to communicate with the Informatica MDM Hub Server: EJB, SOAP, or XML over HTTP.

## Types of SIF API Applications

The SIF API may be used to develop various types of applications, such as:

- Web services that provide a higher level, application/domain specific API.
- Business Process Modeling (BPM) and workflow integration using the Java API or SOAP directly.
- A Java Swing UI to query, view, and edit data in the Informatica MDM Hub.
- Server components in a J2EE application server to get and update data in the Informatica MDM Hub. An EJB can seamlessly integrate with the transactions of the Informatica MDM Hub.
- A JSP or Servlet application to present a portal view including Informatica MDM Hub data.
- Command-line application to run batch jobs and batch groups, to be run manually, scheduled for periodic execution, or included as part of another script or application.

## SIF Services Development Kit (SDK)

The Informatica MDM Hub Resource Kit installer also installs the SIF Services Development Kit (SDK). You copy the SIF SDK to any client system on which you wish to develop and run programs to interact with Informatica MDM Hub using SIF. If you can run a Java virtual machine (JVM) on the client system, you can use the Java classes included in the SIF SDK. This guide refers to these Java classes by the name of the first class you must instantiate, SiperianClient.

You can configure the SDK to use any SIF protocol. If you cannot run a JVM, then you must explicitly use web services (for example, on a pure.NET system) or JMS (for example, on a mainframe system), or XML over HTTP.

## About SIF Access Protocols

The SIF API enables you to implement the request/response interactions using any of the following access protocols:

- Java development environment; tightly-coupled Java remote procedure calls based on Enterprise JavaBeans (EJBs).
- Loosely coupled web services using SOAP protocol: request XML, response XML; WSDL defines the request and response XML. The actual development environment varies: it can be Eclipse, MS Visual Studio (.NET and others), or other web service client tools.
- XML over HTTP (web services minus the SOAP envelope); this is very similar to SOAP/ web services except you’re not using the SOAP protocol to “wrap” the XML request message in a SOAP envelope, nor are you needing to retrieve the response XML from a SOAP message. One advantage of using web services is the concept that “session authentication” is implicit.
- Asynchronous Java Message Service (JMS)-based messages; using XML over HTTP (subscribe/publish).
Each of the above SIF protocols sit on top of the native Informatica MDM Hub protocol, which accepts requests in the form of XML documents or EJBs and returns responses the same way.

SIF also provides a SiperianClient proxy that can be used in a Java development environment to manage the underlying communication protocol to the Informatica MDM Hub APIs. This eliminates the complexity from such a development effort and allows the users to focus on application development by configuring the chosen communication protocol usage.

When you cannot or do not want to use the SiperianClient Java classes, you can interact directly with SIF using other access protocols. This guide does not include information about using the EJB or JMS protocols directly. You can use these protocols only through SiperianClient if you supply an appropriately configured AsynchronousOptions object with a request. Or, you can place the request directly onto the message queue named siperian.sif.jms.queue.

**RELATED TOPICS:**
- “Using Web Services” on page 8
- “Using XML Over HTTP” on page 8
- “Making Asynchronous SIF Requests” on page 31
Using Web Services

Installing Informatica MDM Hub on an application server makes the capabilities of SiperianClient available as a web service on that application server. You can interrogate the web service to obtain Web Services Description Language (WSDL) descriptions of the web service’s operations and arguments. These operations and arguments parallel the methods and arguments of the SiperianClient Java classes that the web service makes available, so you can use the SiperianClient Javadocs for reference information that also applies to the web service.

In general, you use the following generic procedure using a web service interface to implement the SIF API request/response interactions:

1. Prepare the request
2. Submit the request
3. Process the results
4. Perform error handling

Understanding WSDL

If you use a web service interface to Informatica MDM Hub, you use the same classes and methods described in the Javadocs, but through proxies that wrap the interactions in SOAP messages. Use a tool to interpret the WSDL, then look for the corresponding classes in the Javadocs for detailed reference information.

The actual development environment varies: you could use Eclipse, MS Visual Studio (.NET and others), or other web service client tools. This client environment (.NET for example) has tools for reading WSDL and producing proxies that you can call from the programming language you are using (for example, C#). The Eclipse integrated development environment has a web services browser that reads the WSDL and presents the information in a user-friendly way. Simply point the browser at the following URL (where host and port specify the location of the application server supporting Informatica MDM Hub):

http://host:port/cmx/request/wSDL

The proxies communicate with the web service using the SOAP protocol. They receive requests from your application program. They translate the requests into SOAP messages and send them to the web service. The web service decodes the SOAP messages it receives and translates them to Java calls to the SiperianClient running on the application server. The web service receives responses from SiperianClient, encodes them into SOAP messages, and sends them back to the proxies, which return the responses to your application program.

Informatica MDM Hub uses AXIS (version 1.3) to serve up the web services. Axis is java library/tool that is used to configure the SIF API as web services, and then make these web services accessible using a URL. For example, if you use soapUI to enter a URL and view a list of web services, the tool presents the list of web services that was configured in AXIS. Axis gets deployed along with siperian-mrm.ear. For more information, see http://ws.apache.org/axis/.

You can also create and deploy a web service to process ORS-specific requests.

Related Topics:
- “Setting Up the SIF SDK” on page 14

Using XML Over HTTP

Use the Hypertext Transfer Protocol (HTTP) to send requests to Informatica MDM Hub as XML documents and receive responses the same way. The associated requests and responses are for the Informatica MDM Hub classes that appear in Chapter 5, “SIF API Reference” on page 46.
In general, you use the following generic procedure using XML over HTTP to implement the SIF API request/response interactions:

1. Prepare the request
2. Submit the request
3. Process the results
4. Perform error handling

You can access the schemas that describe the requests and responses at the following locations on the application server that hosts Informatica MDM Hub:

http://host:port/cmx/request/xsd/siperian-core.xsd
http://host:port/cmx/request/xsd/siperian-types.xsd
http://host:port/cmx/request/xsd/siperian-metadata.xsd

In these addresses, host:port represents the host name of the computer running the application server and the port on which it accepts Informatica MDM Hub requests. The three schema files provide a logical partition of the schema that governs requests and responses. The siperian-core file contains most of the elements. The siperian-types file contains most of the type definitions, while siperian-metadata describes the objects used in the SIF ListSiperianObjects and DescribeSiperianObjects classes.

After using the schema to construct an XML request message, use the HTTP POST method to send the request to the following address (where host:port is as described above):

http://host:port/cmx/request

The body of the HTTP response is the Informatica MDM Hub response, encoded in XML according to the above schema.

About Informatica MDM Hub Requests

Informatica MDM Hub SIF provides a set of request/response API classes. Each Informatica MDM Hub request has a basic name. A naming convention enables you to predict the names associated with the various SIF protocols from the basic name. For example, the request and response EJBs corresponding to the SIF Put class are called PutRequest and PutResponse. These names correspond in a predictable way to the Java classes used by SiperianClient. They also correspond in a predictable way to the web services described by the Web Services Description Language (WSDL) included in the SDK.

SIF provides access to these requests in the ways described in “About Services Integration Framework (SIF)” on page 3.

How SIF Requests Are Processed

Each SIF class is represented by a request-response pair of objects. Request object represents the action to be performed and a response object contains the result of that action. Request and response objects can have a Java
or XML representations. Java representation requires no additional processing. XML requests will be converted to Java internally.

You can invoke SIF classes using Java/XML representations using any of the following protocols: SOAP, HTTP and EJB. Multiple API calls can participate in a single transaction when using the EJB protocol.

SIF API offers a generic infrastructure to support various components of the Informatica MDM Hub (core MRM, AM and HM).

**Types of SIF Requests**

Informatica MDM Hub SIF provides API classes for the following services. For a complete list of the SIF requests associated with each functional service, and for more information regarding a specific SIF request, see Chapter 5, “SIF API Reference” on page 46.

**Data Steward Services**

Data Steward requests enable developers to build a new breed of applications that rely on the reference data from the Informatica MDM Hub. Can also be used to build data management services managing both data and metadata, including BVT, a single record or sets of records, as well as to perform searches based on match columns. For more information, refer to the following SIF requests:

- “GetLookupValue” on page 61
- “GetLookupValues” on page 62
- “GetMatchedRecords” on page 62
- “GetMatchedHistory” on page 63
- “GetSystemTrustSettings” on page 65
- “GetTrustGraphData” on page 66
- “GetTrustScore” on page 66
- “SearchLookupValues” on page 73
- “SetRecordState” on page 76

**Data Retrieval Services**

Data Retrieval requests enable developers to search for records. For more information, refer to the following SIF requests:

- “GetBvt” on page 60
- “Get” on page 58
Data API requests enable developers to execute Informatica MDM Hub Cleanse, Link, MultiMerge, and Unlink base object requests. For more information, refer to the following SIF requests:

- “Cleanse” on page 54
- “Link” on page 67
- “MultiMerge” on page 69
- “Unlink” on page 77

Data Update/Insert API requests enable developers to execute data updates and inserts on base object records. For more information, refer to the following SIF requests:

- “CleansePut” on page 54
- “Merge” on page 68
- “Put” on page 70
- “Tokenize” on page 77
- “Unmerge” on page 77

Merge Workflow API requests enable developers to execute post-match batch processes, such as search for unmatched or unmerged records. For more information, refer to the following SIF requests:

- “AcceptUnmatchedRecordsAsUnique” on page 50
- “AssignUnmergedRecords” on page 51
- “CanUnmergeRecords” on page 53
- “ClearAssignedUnmergedRecords” on page 56
- “GetAssignedRecords” on page 59
- “GetUnmergedRecordCount” on page 67
- “ReassignRecords” on page 71

Batch Group API requests enable developers to execute batch groups directly without using the Informatica MDM Hub console or stored procedures. For more information, refer to the following SIF requests:

- “ExecuteBatchGroup” on page 58
- “GetBatchGroupStatus” on page 60
- “ResetBatchGroup” on page 72
Metadata Services

Informatica SIF API provides additional services for managing Informatica MDM Hub metadata. For more information, refer to the following SIF requests:

- "ApplyChangeList" on page 51
- "CreateChangeList" on page 56
- "DeleteRelationship" on page 57
- "GetOrsList" on page 64
- "GetOrsMetadata" on page 64
- "ListSiperianObjects" on page 67
- "ValidateChangeList" on page 80
- "ValidateMetadata" on page 80

ORS-specific Services

Packages configured in the ORS databases pave the way for accessing a specific view of Hub data.

- Use of packages is restricted to Informatica internal processes and the Hub Console tool.
- SIF exposes access to Hub data through the configured packages by auto-generating objects (data objects) and services (data services) to access Hub data for the outside world.
- Collectively refers to the auto-generated data objects and associated services.
- Push-button generation from SIF Manager tool.

For more information regarding the ORS-specific APIs, see “Using ORS-specific APIs” on page 21.

State Management Services

Informatica MDM Hub supports workflow tools by storing pre-defined system states for base object and XREF records. By enabling state management on your data, Informatica MDM Hub offers the following additional flexibility:

- Allows integration with workflow integration processes and tools
- Supports a “change approval” process
- Tracks intermediate stages of the process (pending records)

*State management* is the process for managing the system state of base object and XREF records to affect the processing logic throughout the MRM data flow. Informatica MDM Hub supports the following system states: ACTIVE, PENDING, and DELETED.

You can assign a system state to base object and XREF records at various stages of the data flow using the Hub tools that work with records. In addition, you can use the various Hub tools for managing your schema to enable state management for a base object, or to set user permissions for controlling who can change the state of a record.

State Management API requests enable developers to restore state-enabled records with state set to DELETED, as well as promote pending XREF records. In order for a record to be deleted, it must be in either the ACTIVE state for *soft delete* (a base object or an XREF record is marked as deleted in a user attribute or in the HUB_STATE_IND) or the PENDING state for *hard delete* (a base object or XREF record is physically removed from the database). For more information, see the chapter on state management in the *Informatica MDM Hub Administrator Guide*.
RELATED TOPICS:

- “Delete” on page 56
- “PromotePendingXrefs” on page 69
- “Restore” on page 72

User Management Services

User Management API requests enable developers to manage user security. For more information, see the security chapter in Informatica MDM Hub Administrator Guide.

RELATED TOPICS:

- “Authenticate” on page 53

Other Services

In addition, the Informatica SIF API provides additional services for registering and unregistering users, managing the audit trail, and other miscellaneous services. For more information, refer to the following SIF requests:

- “Audit” on page 52
- “GetSiperianObjectCompatibility” on page 65
- “RegisterUsers” on page 72
- “UnregisterUsers” on page 78

Using SIF SDK to Interface with SIF Classes

The class `com.siperian.sif.client.SiperianClient` contains two essentially equivalent static methods for creating an instance of `SiperianClient` that is customized by a properties file or by an equivalent `java.util.Properties` object. The properties determine the protocols that the instance uses to communicate with Informatica MDM Hub.

The `process` method of `SiperianClient` provides the basic request/response interaction between the client program and Informatica MDM Hub. It accepts any subclass of `com.siperian.sif.message.SiperianRequest` as an argument. When it processes the request successfully, it returns a `com.siperian.sif.message.SiperianResponse`, which you must cast to the correct response subclass. Otherwise it throws `SiperianServerException`.

Each Informatica MDM Hub interaction uses a pair of subclasses of `SiperianRequest` and `SiperianResponse`. For example, an interaction to carry out a SIF Put request uses the classes `PutRequest` and `PutResponse`. 
Chapter 2

Setting Up the SIF SDK

This chapter includes the following topics:
- Overview of Setting Up the SIF SDK, 14
- Before You Begin, 14
- Deploying the SIF SDK, 14
- About SIF API Javadoc, 15
- Building Web Services, 16

Overview of Setting Up the SIF SDK

This chapter explains how to set up the environment and tools necessary to use the SIF SDK. It lists the prerequisites for using SIF, deploying the SIF SDK, and building web services.

Before You Begin

Before using the Informatica Services Integration Framework, you must have following software installed on your system:
- Informatica MDM Hub
- Ant 1.6.1 is preferred
- Eclipse Web Tools Platform IDE or 3.2 with WTP Plug-in
- Application server (WebLogic, WebSphere, JBoss)
- AXIS 1.3 (optional)

Refer to the Informatica MDM Hub Release Notes for information about the specific versions of JDK, Ant, and supported application servers.

Deploying the SIF SDK

The environment variable SIP_HOME denotes the directory into which you install Informatica MDM Hub. In that directory is a file named siperian-sifsdk.zip. Unzip this file to a location on your development machine.
The resulting directory structure contains libraries, build files, Javadocs, and everything else you need to build web services as EAR files for deployment on an application server.

### About SIF API Javadoc

The Informatica MDM Hub SIF API includes Javadoc that describe the functionality and use of the individual SiperianClient java classes and objects. **Javadoc** is the Sun Microsystems standard for generating HTML documentation from Java source code.

One you’ve deployed the SIF SDK, you can view the Javadocs for SiperianClient and its associated classes and objects. Open `index.html` to see a right-hand frame and two left-hand frames, as shown in the following illustration. The left frames provide links to the pages for all packages and all classes. The lower left frame displays the links associated with the package you select in the upper left frame. You can select **All Classes** in the upper left frame to see a combined list of classes from all packages in the lower left frame.

The right frame changes to show the pages you select. Begin by exploring the classes of the `com.siperian.sif.message` package. Most of the classes used in application programs are in this package and its subpackages.
Building Web Services

Using the SIF Manager tool produces web services corresponding to the ORS-specific Java classes. You can use the build.xml file included in the SIF SDK to build additional custom web services. A detailed example of how to do this appears in a separate technical note.

**RELATED TOPICS:**

- "Using ORS-specific APIs" on page 21
CHAPTER 3

Using the SIF SDK

This chapter includes the following topics:
- Overview of Using the SIF SDK, 17
- About SIF SDK, 17
- Using SIF API Requests, 19
- Using ORS-specific APIs, 21
- Making Asynchronous SIF Requests, 31
- Using the Security Access Manager (SAM) with SIF, 35
- Using Informatica MDM Hub Metadata Management API, 36
- Using Transactions in the EJB Protocol, 37

Overview of Using the SIF SDK

This chapter provides an overview for the Services Integration Framework API, how to use the SIF requests, Security Access Manager (SAM), ORS-specific APIs, the Metadata Management APIs, and information for working with transactions.

About SIF SDK

The SIF Development Kit (SIF-SDK) is a toolkit for development of web services and Java applications that interact with the Informatica MDM Hub. SIF-SDK is packaged with sample code that can be run in the Eclipse IDE. SIF-SDK is delivered as part of the Resource Kit containing directory structures, libraries and build files in the resourcekit\sifsdk directory of the Hub server installation.

The SIF SDK includes:
- utilities to build and deploy SIF applications
- set of Java API classes for creating services
Informatica MDM Hub SIF Manager

In addition, Informatica MDM Hub also includes the SIF Manager, an Informatica MDM Hub Console tool that generates (and deploys):

- data objects (Client Jar file)
- web services for ORS-specific APIs (WSDL and EAR file)
- ORS-specific JMS Event Messages for the current ORS. The XML schema for these messages can be downloaded or accessed using a URL. For more information about JMS Event Messages, see the Informatica MDM Hub Administrator Guide.
- ORS-specific APIs.

**RELATED TOPICS:**

- "Making Asynchronous SIF Requests" on page 31
- "Using ORS-specific APIs" on page 21

SIF Development Kit (SIF-SDK)

Using the SIF Development Kit (SIF-SDK) for your web services development provides the following advantages:

- automatic generation and deployment of data objects and data services for web services-based interaction
- generation of a "Client Jar file" that includes data objects that can be used in external applications
- creation and management of complex integration scenarios by combining data objects from different Informatica MDM Hub schemas (ORS databases)

You can use the SIF SDK to create data objects, components and client services, business services, and GUI controls for creation and deployment of web-based, rich-client applications.
SIF Client Interface

The siperian-client.jar file provides a Java library for use of the SIF APIs:

- com.siperian.sif.client package manages communication with the Hub server. Through configuration, the same API can use different communication protocols to the Hub Server: EJB, SOAP, or plain HTTP
- com.siperian.sif.message package provides data objects and abstract classes for the Informatica MDM Hub APIs. All requests are sub-classes of SiperianRequest.

Exact Matches on Fuzzy Base Objects

To perform exact matches on fuzzy base objects, you must add the following parameter to cleanse\resources\cmxcleanse.properties:

    cmx.server.match.exact_match_fuzzy_bo_api=1

By default, this parameter is not listed by the Hub install. After adding the parameter and setting it to 1, you can do exact matching on a fuzzy BO in the API.

Note: You must restart the application server for changes to this parameter to take effect.

SIF API Debug Log

The SIF API Debug log is cmxserver.log. You can access this file here:

    <hub installation>/logs/

Using SIF API Requests

Each SIF service has a pair of messages: request and response. The typical usage is to instantiate and populate a request (subclass of SiperianRequest), then call the SiperianClient.process() method, and cast the response (subclass of SiperianResponse), to the corresponding response class.

SiperianRequest Class

The SIF API classes (Java) that implement individual Informatica MDM Hub operations all extend the class SiperianRequest. This class provides access to the following information:

- Username and password of the user associated with the request.
- Every request must have an associated user. This information determines whether or not the request can have access to the records and resources it needs.
- ORS to which the request is directed.
- If the request does not specify an ORS, it goes to the user’s default ORS.
- Interaction identifier for grouping requests into interactions.
- An AsynchronousOptions object containing information necessary for asynchronous requests and responses.

    If the AsynchronousOptions object is null, SIF processes the request synchronously. If the object is not null, SIF processes the request synchronously or asynchronously according to the value of an option in the request object.
When you process a request asynchronously, SIF immediately returns a dummy response with message status that tells you that it is processing the request asynchronously. The actual response goes to a JMS queue or topic that you specify. If you do not specify a queue or topic, SIF discards the actual response.

You can include a correlation ID to enable you to identify the response to this request from among multiple responses.

- **Transaction attribute type**

  The transaction attribute type specifies whether and how a request can participate in transactions. You can get but not set this information. Different request types have different transaction attribute types. The possible transaction attribute types are:

<table>
<thead>
<tr>
<th>Transaction Attribute Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUPPORTS</td>
<td>Works within transactions</td>
</tr>
<tr>
<td>REQUIRED</td>
<td>Requires a transaction</td>
</tr>
<tr>
<td>REQUIRES_NEW</td>
<td>Requires a new transaction</td>
</tr>
<tr>
<td>NOT_SUPPORTED</td>
<td>Does not work with transactions</td>
</tr>
</tbody>
</table>

- **Name of the request**

  You can interrogate a request object to see which specific request type it is an instance of. After setting up the request, pass it to the process method for execution. The process method either throws an exception or returns a response object of the appropriate type.

### Constructing Requests

Every SIF class has an associated Request object. This object contains the description of what action is to be taken by SIF.

For example, the following sample SIF call is a search query request. This request will execute the package “PARTY_ADDRESS_READ_PKG”. It will return no more than five (5) rows and use the filter criteria “PARTY_FULL_NAME LIKE” and an inserted parameter.

```java
SearchQueryRequest request = new SearchQueryRequest();
request.setRecordsToReturn(5); // Required
request.setSuperiorObjectUID("PACKAGE.PARTY_ADDRESS_READ_PKG"); // Required
request.setFilterCriteria("PARTY_FULL_NAME LIKE ?");
```

### Processing Responses

Every SIF request returns an associated Response object. This object contains the results for the request.

For example, the following sample SIF call is a response from a GetOrsMetadataRequest request:

```java
GetOrsMetadataResponse getOrsMetadataResponse = (GetOrsMetadataResponse)
sifClient.process(getOrsMetadataRequest );
System.out.println("ORS Metadata (first line only): " +
getOrsMetadataResponse.getRepositoryXml().substring(0, 80));
```

### About Records

A Record is a collection of Fields, essentially a list of name/value pairs. Each name in the Record must be unique. A Field represents a named value in a Record and is strongly typed. Values can be: a String, BigInteger, BigDecimal or Date.
Using ORS-specific APIs

You use the Hub Console SIF Manager tool to generate and deploy the code to support SIF APIs for packages, remote packages, mappings, and cleanse functions in an ORS database. Once generated, the ORS-specific APIs will be available with SiperianClient by using the client jar and also as a web service.

**Note:** Informatica MDM Hub generates ORS-specific APIs only for objects that are secure.

ORS-specific APIs provide the following additional benefits over the standard SIF API:

- Field names are provided with strongly-typed values
- Provides a simplified use of UIDs:
  - Package UIDs are implicit in the ORS-specific API name
  - Match rule UIDs are selectable
- Allows you to find objects out-of-sync with the generated API

Both Requests and Responses can contain Records.

**Usage Example**

```java
--
CleanseRequest cleanseRequest = new CleanseRequest();
cleanseRequest.setCleanseFunctionName( cleanseFunctionName );

Record record = new Record();
for(int i=2; i<args.length; i=i+2)
{
    Field field = new Field();
    field.setName( args[i] );
    field.setStringValue( args[i+1] );
    record.setField(field);
}
cleanseRequest.setRecord(record);
CleanseResponse cleanseResponse = (CleanseResponse) sipClient.process(cleanseRequest);
--
```

**About RecordKey**

Uniquely identifies a record in the Informatica MDM Hub. A record can be identified by a combination of:

- **rowid**—the ROWID_OBJECT value for a record
- **systemName & pkey**—a system name and primary key value in the system
- **one or more GBIDs**—Global Business identifiers that have been defined for an object

**Usage Example**

```java
--
RecordKey recordKey = new RecordKey();
    recordKey.setSystemName( systemName );
    recordKey.setSourceKey( sourceKey );

    putRequest.setRecordKey(recordKey );
--
```
Generating ORS-specific APIs

Use the SIF Manager tool in the Informatica MDM Hub console to produce ORS-specific APIs for secure objects in the form of Java classes deployed on the application server. You can also generate custom web services and deploy them to the same application server. For more information regarding the SIF Manager, see the Informatica MDM Hub Administrator Guide.

**Note:** This operation requires access to a Java compiler on the application server machine. The Java software development kit (SDK) includes a compiler in tools.jar. The Java runtime environment (JRE) does not contain a compiler.

The following procedure assumes that you have already configured the base objects, packages, and mappings of the ORS. If you subsequently change any of these, regenerate the ORS-specific APIs.

To generate the ORS-specific APIs:

1. In the Hub Console, connect to an Operational Record Store (ORS). To learn more, see Changing the Target Database in the Informatica MDM Hub Administrator Guide.
2. Expand the Informatica Utilities workbench and then click SIF Manager.
   
   The SIF Manager tool displays the following areas:

<table>
<thead>
<tr>
<th>Area</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIF ORS-Specific APIs</td>
<td>Shows the logical name, java name, WSDL URL, and API generation time for the SIF ORS-specific APIs. Use this function to generate and deploy SIF APIs for packages, remote packages, mappings, and cleanse functions in an ORS database.</td>
</tr>
<tr>
<td>Out of Sync Objects</td>
<td>Shows the database objects in the schema that are out of sync. with the generated schema.</td>
</tr>
</tbody>
</table>

3. Acquire a write lock.
4. Enter a value in the Logical Name field.
   
   You can keep the default value, which is the name of the ORS. If you change the logical name, it must be different from the logical name of any other ORS registered on this server.
5. Click Generate and Deploy ORS-specific SIF APIs.
   
   SIF Manager generates the APIs. The time this requires depends on the size of the ORS schema. When the generation is complete, SIF Manager deploys the ORS-specific APIs and displays their URL. You can use the URL to access the WSDL descriptions from your development environment.
6. Click Download Client JAR File.
   
   SIF Manager downloads a file called nameClient.jar, where name is the logical name you provided in step “Generating ORS-specific APIs” on page 22, to a location you specify on your local machine. The JAR file includes the new classes and their Javadocs.
   
   It is customary to place the Jar file in the lib directory of the SIF SDK directory structure on your machine.
7. If you are using an integrated development environment (IDE) and have a project file for building web services, add the JAR file to your build classpath.
8. Modify the SIF SDK build.xml file so that the build_war macro includes the JAR file.
   
   **Note:** SIF API generation requires at least one secure package, remote package, cleanse function or mapping.
   
   Once generated, the ORS-specific APIs will be available with SiperianClient by using the client jar and also as a web service. The logical name is used to name the components of the deployment. SIF Manager deploys the ORS-
specific APIs and displays their URL. You can use the URL to access the WSDL descriptions from your development environment.

Note: To prevent running out of heap space for the associated SIF API Javadocs, you may need to increase the size of the heap. The default heap size is 256M. You can also override this default using the SIF.JVM.HEAP_SIZE parameter.

ORS-specific API Classes

Informatica MDM Hub ORS-specific APIs include the following classes:

- Cleanse [Resource_Name]
- CleansePut [Resource_Name]
- Get [Resource_Name]
- Put [Resource_Name]
- SearchMatchColumn [Resource_Name]
- SearchMatchRecord [Resource_Name]
- SearchQuery [Resource_Name]

Note: The Resource_Name depends on the name of the ORS-specific fields for each secure ORS resource.

Populating SIF API Field Parameters

The following table provides generic parameter information for the ORS-specific APIs Note that the actual list of fields depends on your specific ORS.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>String</td>
<td>User name. [Optional]</td>
</tr>
<tr>
<td>password</td>
<td>String</td>
<td>Password [Optional]</td>
</tr>
<tr>
<td>encrypted</td>
<td>Boolean</td>
<td>Password encrypted? [Optional]</td>
</tr>
<tr>
<td>securityPayload</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>orsID</td>
<td>String</td>
<td>ID of the Operational Record Store (ORS). [Optional]</td>
</tr>
<tr>
<td>interactionId</td>
<td>String</td>
<td>[Optional]</td>
</tr>
<tr>
<td>isAsynchronous</td>
<td>Boolean</td>
<td>Asynchronous? [Optional]</td>
</tr>
<tr>
<td>jmsReplyTo</td>
<td>String</td>
<td>[Optional]</td>
</tr>
<tr>
<td>jmsCorrelationId</td>
<td>String</td>
<td>[Optional]</td>
</tr>
<tr>
<td>sortCriteria</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>recordsToReturn</td>
<td>Int</td>
<td></td>
</tr>
<tr>
<td>returnTotal</td>
<td>Boolean</td>
<td></td>
</tr>
<tr>
<td>Field Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>matchType</td>
<td>MatchType object</td>
<td></td>
</tr>
<tr>
<td>matchRuleSetUid</td>
<td>String</td>
<td>[Optional]</td>
</tr>
<tr>
<td>disablePaging</td>
<td>Boolean</td>
<td>[Optional]</td>
</tr>
<tr>
<td>systemName</td>
<td>String</td>
<td>System name [Optional]</td>
</tr>
<tr>
<td>sourceKey</td>
<td>String</td>
<td>PKEY_SRC_OBJECT of the cross-reference record. [Optional]</td>
</tr>
<tr>
<td>columnUid</td>
<td>String</td>
<td>Column UID</td>
</tr>
<tr>
<td>package</td>
<td>Boolean</td>
<td>Package? [Optional]</td>
</tr>
<tr>
<td>xref</td>
<td>Boolean</td>
<td>XREF? [Optional]</td>
</tr>
<tr>
<td>pendingXref</td>
<td>Boolean</td>
<td>Pending XREF? [Optional]</td>
</tr>
<tr>
<td>deletedXref</td>
<td>Boolean</td>
<td>Deleted XREF? [Optional]</td>
</tr>
<tr>
<td>history</td>
<td>Boolean</td>
<td>History? [Optional]</td>
</tr>
<tr>
<td>xrefHistory</td>
<td>Boolean</td>
<td>XREF History? [Optional]</td>
</tr>
<tr>
<td>raw</td>
<td>Boolean</td>
<td>Raw (table?) [Optional]</td>
</tr>
<tr>
<td>returnTrustScores</td>
<td>Boolean</td>
<td>Return trust score? [Optional]</td>
</tr>
<tr>
<td>returnLineage</td>
<td>Boolean</td>
<td>Return lineage? [Optional]</td>
</tr>
<tr>
<td>generateSourceKey</td>
<td>Boolean</td>
<td>Generate source key?</td>
</tr>
<tr>
<td>lastUpdateDate</td>
<td>Date time</td>
<td>Date of last update [Optional]</td>
</tr>
</tbody>
</table>

**Cleanse[Resource_Name]**

Cleanse invokes a cleanse function defined in Informatica MDM Hub. The request specifies the record and the cleanse function. The response contains a record containing the cleansed data. Use the following syntax for calling the ORS-specific version of Cleanse:

```xml
Cleanse[ResourceName]
```

Where `ResourceName` defines the search results.

**Usage Examples**

For Cleanse [ResourceName] Request:

```xml
<q0:cleanse xmlns:q0="urn:siperian.api">
  <q0:username>siftester</q0:username>
  <q0:password>si</q0:password>
  <q0:encrypted>false</q0:encrypted>
</q0:cleanse>
```

Chapter 3: Using the SIF SDK
CleansePut[Resource_Name]

CleansePut combines the functions of the Cleanse and Put API requests to clean the specified record and update or insert it in the specified table in a single request. CleansePut replicates the Stage and Load batch processes that move data from the landing table, through the cleansing process into the staging table and finally into the base object. The physical landing and staging tables are not used by CleansePut.

Use the following syntax for calling the ORS-specific version of CleansePut:

```
CleansePut[ResourceName]
```

Where ResourceName defines the search results.

Usage Example

For CleansePut[ResourceName] Request:

```
<q0:cleansePut xmlns:q0="urn:siperian.api"
    <q0:username>b_c_m_create_user</q0:username>
    <q0:password>password</q0:password>
    <q0:encrypted>false</q0:encrypted>
    <q0:orsId>weaks01-weeks01-CMX_SIF</q0:orsId>
    <q0:systemName>CRM</q0:systemName>
    <q0:record>
      <q0:field>
        <q0:name>CHAR_SOURCE</q0:name>
        <q0:stringValue>CleanseFTest</q0:stringValue>
      </q0:field>
      <q0:field>
        <q0:name>VCHAR_SOURCE</q0:name>
        <q0:stringValue>0001_ValidMappingInsert</q0:stringValue>
      </q0:field>
      <q0:field>
        <q0:name>INT_SOURCE</q0:name>
        <q0:bigIntegerValue>1548851241</q0:bigIntegerValue>
      </q0:field>
      <q0:field>
        <q0:name>LAST_UPDATE_DATE</q0:name>
        <q0:dateValue>2006-11-26T16:54:40.531Z</q0:dateValue>
      </q0:field>
    </q0:record>
  </q0:cleansePut>
```

For CleansePut[ResourceName] Response:

```
<ns1:cleansePutReturn xmlns="urn:siperian.api" xmlns:ns1="urn:siperian.api">
  <ns1:message>The CLEANSE PUT was processed successfully.</ns1:message>
  <ns1:recordKey>
    <ns1:systemName>CRM</ns1:systemName>
  </ns1:recordKey>
</ns1:cleansePutReturn>
```
Get[Resource_Name]

Retrieves a single record from the specified package using a known key. The Get API can be used against both the regular MRM packages ("PACKAGE." SipersianObjectUid prefix) and the remote packages ("REMOTE_PACKAGE." SipersianObjectUid prefix). Use the following syntax for calling the ORS-specific version of Get:

Get[ResourceName]

Where ResourceName defines the search results.

By default Get returns the package record for the specified key.

Usage Example

For Get[ResourceName] Request:

```xml
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/*
xmlns:urn="urn:sipersian.api"
<soapenv:Header/>
<soapenv:Body>
<urn:get>
<!Optional:-->  
<urn:username>admin</urn:username>  
<!Optional:-->  
<urn:password>  
<urn:password>admin</urn:password>  
<urn:encrypted>false</urn:encrypted>  
</urn:password>  
<!Optional:-->  
<urn:orsId=localhost-0rcl-DS_U11</urn:orsId>  
<urn:sipersianObjectUid>PACKAGE_PKG_PARTY</urn:sipersianObjectUid>  
</urn:recordKey>
<urn:rowid>100</urn:rowid>  
</urn:recordKey>  
<!Optional:-->  
<urn:recordTypes>  
<!Optional:-->  
<urn:package>true</urn:package>  
<!Optional:-->  
<urn:xref>false</urn:xref>  
<!Optional:-->  
<urn:pendingKref>false</urn:pendingKref>  
<!Optional:-->  
<urn:deletedKref>false</urn:deletedKref>  
<!Optional:-->  
<urn:history>false</urn:history>  
<!Optional:-->  
<urn:xrefHistory>false</urn:xrefHistory>  
<!Optional:-->  
<urn:raw>false</urn:raw>  
</urn:recordTypes>  
<!Optional:-->  
<urn:returnTrustScores>false</urn:returnTrustScores>  
<!Optional:-->  
<urn:returnLineage>false</urn:returnLineage>  
</urn:get>
</soapenv:Body>
</soapenv:Envelope>
```

For Get[ResourceName] Response:

```xml
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/*
<soapenv:Body>
<getReturn xmlns="urn:sipersian.api">
```
Put [Resource_Name]

Put adds a new row or updates an existing row of a base object table. The package must be put-enabled. Use the following syntax for the ORS-specific version of Put:

Put [ResourceName]

Where ResourceName defines the search results.

To learn more about creating put-enabled packages, see the Informatica MDM Hub Administrator Guide.

Important: Use the AddRelationship and UpdateRelationship requests to add or update relationship records.

Using the Put request or an ORS-specific put request to update relationships can lead to improperly formed relationship records.

Usage Example

For Put [ResourceName] Request:

```xml
<q0:put xmlns:q0="urn:siperian.api">
  <q0:username>p_h_create_user</q0:username>
  <q0:password></q0:password>
  <q0:encrypted>false</q0:encrypted>
  <q0:password/>
  <q0:orsId>wewks01-wewks01-CM3_SIP</q0:orsId>
  <q0:recordKey>
    <q0:systemName>CRM</q0:systemName>
  </q0:recordKey>
  <q0:record>
    <q0:field>
      <q0:name>CUSTOMER_CLASS</q0:name>
      <q0:stringValue>R</q0:stringValue>
    </q0:field>
    <q0:field>
      <q0:name>FULL_NAME</q0:name>
      <q0:stringValue>PUTM1</q0:stringValue>
    </q0:field>
    <q0:field>
      <q0:name>SECOND_NAME</q0:name>
      <q0:stringValue>W1</q0:stringValue>
    </q0:field>
    <q0:field>
      <q0:name>LAST_NAME</q0:name>
      <q0:stringValue>W1</q0:stringValue>
    </q0:field>
  </q0:record>
</q0:put>
```
For Put\[ResourceName\] Response:

```
<ns1:putReturn xmlns:ns1="urn:siperian.api" xmlns:ns2="urn:siperian.api">
  <ns1:message>The PUT was processed successfully</ns1:message>
  <ns1:recordKey/>
  <ns1:systemName>CRM</ns1:systemName>
  <ns1:rowId>5342</ns1:rowId>
  <ns1:sourceKey>323387000</ns1:sourceKey>
  <ns1:actionType>Insert</ns1:actionType>
</ns1:putReturn>
```

**SearchMatchColumn\[Resource_Name\]**

User specifies values of a match column. Use the following syntax for calling the ORS-specific version of SearchMatchRecord:

```
SearchMatchColumn\[ResourceName\]
```

Where `ResourceName` defines the search results.

In addition:
- User specifies the match column value.
- Standard match rule options apply.
- Results are returned as records of the package.

**Usage Example**

For SearchMatchColumn\[ResourceName\] Request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:urn="urn:datastewarddemo.siperian.api" xmlns:urn1="urn:siperian.api">
  <soapenv:Header/>
  <soapenv:Body>
    <urn:searchMatchColumnPkgParty>
      <!-- Optional: -->
      <urn1:username>admin</urn1:username>
      <!-- Optional: -->
      <urn1:password>admin</urn1:password>
      <urn1:encrypted>false</urn1:encrypted>
    </urn:searchMatchColumnPkgParty>
    <!-- Optional: -->
    <urn1:securityPayload>cid:1224793701596</urn1:securityPayload>
  </soapenv:Body>
</soapenv:Envelope>
```
SearchMatchColumn[Resource_Name] Response:

Results are returned as records of the package.
Standard match rule options apply.
User specifies the values of search package.

SearchMatchRecord[Resource_Name]

User specifies values of a related record (shares the same parent base object). SearchMatchRecord extracts the match column values from the record. Use the following syntax for calling the ORS-specific version of SearchMatchRecord:

SearchMatchRecord[ResourceName]

Where ResourceName defines the search results.

In addition:
- User specifies the values of search package.
- Standard match rule options apply.
- Results are returned as records of the package.

Usage Example

For SearchMatchRecord[ResourceName] Request:

```xml
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/
xmlns:urn="urn:datatestwarddemo.siperian.api" xmlns:urn="urn:siperian.api">
  <soapenv:Header/>
  <soapenv:Body>
    <urn:searchMatchRecordPkgParty>
      <!--Optional:-->  
      <urn:username>admin</urn:username>
      <!--Optional:-->  
      <urn:password>
        <urn1:password>admin</urn1:password>
        <urn1:encrypted>false</urn1:encrypted>
      </urn:password>
      <!--Optional:-->  
      <urn:orsId>localhost-orcl-OR_11</urn:orsId>
      <urn:sortCriteria/>
      <urn:recordsToReturn>100</urn:recordsToReturn>
      <urn:returnTotal>true</urn:returnTotal>
      <urn:matchType>NONE</urn:matchType>
      <!--Optional:-->  
      <urn:pkpOrganization/>
      <!--Optional:-->  
      <urn:pkpOrganization/>
      <!--Optional:-->  
      <urn:pkpParty/>
      <!--Optional:-->  
      <urn:partyType/>
      <!--Optional:-->  
      <urn:pkpParty/>
      <!--Optional:-->  
    </urn:searchMatchRecordPkgParty>
  </soapenv:Body>
</soapenv:Envelope>
```
For SearchQuery[ResourceName] Request:

Usage Example

Results are returned as records of the parameter determines the maximum number of records that can be returned.

For SearchMatchRecord[ResourceName] Response:

SearchQuery[Resource_Name]

SearchQuery searches for records in a package based on an SQL condition clause. The condition clause can reference any columns in the package and can use operators supported by the target database. A system parameter determines the maximum number of records that can be returned.

Note: You can perform a case-insensitive search if the case.insensitive.search property is set to true in the cmxserver.properties.xml file. For example, to perform a case-insensitive search, you can specify a search criterion using the SearchQuery API, such as lower(name)=lower("Jim").

Use the following syntax for calling the ORS-specific version of SearchQuery:

SearchQuery[Resource_Name]

Where the Resource_Name defines the search criteria and search results.

In addition:

- User specifies fields of the package
- Implicit "AND" criteria used across Fields
- Implicit "OR" criteria used across Records
- Results are returned as records of the package

Usage Example

For SearchQuery[ResourceName] Request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/
xmlns:urn="urn:datastewarddemo.siperian.api" xmlns:urn1="urn:siperian.api">
  <soapenv:Header/>
  <soapenv:Body>
    <urn:SearchQueryPkgParty>
      <!--Optional:-->
      <urn1:username>admin</urn1:username>
      <!--Optional:-->
      <urn1:password>admin</urn1:password>
      <urn1:encrypted>false</urn1:encrypted>
    </urn1:password>
    <urn1:orsId>localhost-orcl-DS_UI1</urn1:orsId>
    <urn1:recordsToReturn>10</urn1:recordsToReturn>
    <urn1:returnTotal>true</urn1:returnTotal>
    <!--Zero or more repetitions:-->  
    <urn:pkgParty>
      <!--Optional:-->
      <urn1:displayName>ELIEZER MENDEZ</urn1:displayName>
    </urn:pkgParty>
  </soapenv:Body>
</soapenv:Envelope>
```
Making Asynchronous SIF Requests

There are two ways to make asynchronous requests:

- Set the AsynchronousOptions on a request. When the request is submitted, it is placed on a queue for processing. If the jmsReplyTo field is set, the response is posted to the specified queue. If the jmsCorrelationId is set, this ID is set on the response.

- Place the request directly on the JMS queue (queue/superian.sif.jms.queue). If the replyTo header is set, the response is posted to the specified queue. If the jmsCorrelationId is set, this ID is set on the response.

The response that is placed on the outgoing JMS queue (if configured) will carry the correlation ID as set on the request message.

**Note:** The Services Integration Framework (SIF) uses a message-driven bean (MDB) on the JMS queue (named superian.sif.jms.queue) to process incoming asynchronous SIF requests. This message queue and the connection factory (named superian.mrm.jms.xaconnectionfactory) must be configured for the specific application server you are using for your Informatica MDM Hub implementation. Correctly configured message queues are essential to a fully-functioning Informatica MDM Hub installation. The Informatica installer automatically sets up message queues and connection factory configuration. If you need to manually configure your message queues or connection factories for testing or troubleshooting purposes, see the [Informatica MDM Hub Installation Guide](#).

About JMS Event Messages

Informatica MDM Hub Console includes a JMS Event Schema Manager tool that you can use to generate and deploy ORS-specific JMS event messages for the current ORS. The XML schema for these messages can be downloaded or accessed using a URL. For more information about JMS event messages, see the "Configuring the Publish Process" and "Generating ORS-specific APIs and Message Schemas" chapters in the [Informatica MDM Hub Administrator Guide](#).

JMS ORS-specific event messages:

- use SIF-style XML (utilized Castor)
- provide an associated message schema (this is available using a file or URL)
- enable scheduled auto-generation

```xml
<urn:searchQueryPkgParty>
  <soapenv:Body>
    <soapenv:Envelope>
      <searchQueryPkgPartyReturn xmlns="urn:datastewarddemo.siperian.api">
        <ns1:message xmlns:ns1="urn:siperian.api">The SEARCH QUERY REQUEST was processed successfully</ns1:message>
      </searchQueryPkgPartyReturn>
    </soapenv:Body>
  </soapenv:Envelope>
</urn:searchQueryPkgParty>
```

For `SearchQuery[ResourceName] Response`:

```xml
  <soapenv:Body>
    <searchQueryPkgPartyReturn xmlns="urn:datastewarddemo.siperian.api">
      <ns1:message xmlns:ns1="urn:siperian.api">The SEARCH QUERY REQUEST was processed successfully</ns1:message>
    </searchQueryPkgPartyReturn>
  </soapenv:Body>
</soapenv:Envelope>
```
are backwards-compatible; the message queue can be configured to generate new or legacy format

The JMS Event Schema Manager uses an XML schema that defines the message structure the Hub uses to
generate JMS messages. This XML schema is included as part of the Informatica MDM Hub Resource Kit. (The
ORS-specific schema is available using a URL or downloadable as a file).

**Note:** JMS Event Schema generation requires at least one secure package or remote package.

**Important:** If there are two databases that have the same schema (for example, CMX_ORS), the logical name
(which is the same as the schema name) will be duplicated for JMS Events when the configuration is initially
saved. Consequently, the database display name is unique and should be used as the initial logical name instead
of the schema name to be consistent with the SIF APIs. You must change the logical name before generating the
schema.

Each ORS has an XSD file specific to the ORS that uses the elements from the common XSD file <siperian-mrm-
events.xsd>. The ORS-specific XSD is named as <ors-name>-siperian-mrm-event.xsd. The XSD defines two objects
for each package and remote package in the schema:

<table>
<thead>
<tr>
<th>Object Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[packageName]Event</td>
<td>Complex type containing elements of type EventMetadata and [packageName].</td>
</tr>
</tbody>
</table>
| [packageName]Record| Complex type representing a package and its fields. Also includes an element of type
|                   | Siperevent. This complex type resembles the package record structures defined in
|                   | the Informatica MDM Hub Services Integration Framework (SIF).                 |

**Note:** If legacy XML event message objects are to be used, ORS-specific message object generation is not
required.

**Elements in an XML Message**

The following table describes the elements in an XML message.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root Node</td>
<td></td>
</tr>
<tr>
<td>&lt;siperianEvent&gt;</td>
<td>Root node in the XML message.</td>
</tr>
<tr>
<td>Event Metadata</td>
<td></td>
</tr>
<tr>
<td>&lt;eventMetadata&gt;</td>
<td>Root node for event metadata.</td>
</tr>
<tr>
<td>&lt;messageId&gt;</td>
<td>Unique ID for siperianEvent messages.</td>
</tr>
</tbody>
</table>
| <eventType>         | Type of event. One of the following values: Insert, Update, Update XREF,
<p>|                     | Accept as Unique, Merge, Unmerge, Merge Update.                             |
| &lt;baseObjectUid&gt;     | UID of the base object affected by this action.                             |</p>
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;packageUid&gt;</td>
<td>UID of the package associated with this action.</td>
</tr>
<tr>
<td>&lt;messageDate&gt;</td>
<td>Date/time when this message was generated.</td>
</tr>
<tr>
<td>&lt;orsId&gt;</td>
<td>ID of the Operational Record Store (ORS) associated with this event.</td>
</tr>
<tr>
<td>&lt;triggerUid&gt;</td>
<td>UID of the rule that triggered the event that generated this message.</td>
</tr>
<tr>
<td>Event Details</td>
<td></td>
</tr>
<tr>
<td>&lt;eventTypeEvent&gt;</td>
<td>Root node for event details.</td>
</tr>
<tr>
<td>&lt;sourceSystemName&gt;</td>
<td>Name of the source system associated with this event.</td>
</tr>
<tr>
<td>&lt;sourceKey&gt;</td>
<td>Value of the PKEY_SRC_OBJECT associated with this event.</td>
</tr>
<tr>
<td>&lt;eventDate&gt;</td>
<td>Date/time when the event was generated.</td>
</tr>
<tr>
<td>&lt;rowid&gt;</td>
<td>RowID of the base object record that was affected by the event.</td>
</tr>
<tr>
<td>&lt;xrefKey&gt;</td>
<td>Root node of a cross-reference record affected by this event.</td>
</tr>
<tr>
<td>&lt;systemName&gt;</td>
<td>System name of the cross-reference record affected by this event.</td>
</tr>
<tr>
<td>&lt;sourceKey&gt;</td>
<td>PKEY_SRC_OBJECT of the cross-reference record affected by this event.</td>
</tr>
<tr>
<td>&lt;packageName&gt;</td>
<td>Name of the secure package associated with this event.</td>
</tr>
<tr>
<td>&lt;columnName&gt;</td>
<td>Each column in the package is represented by an element in the XML file.</td>
</tr>
<tr>
<td></td>
<td>Examples: rowidObject and consolidationInd. Defined in the ORS-specific XSD</td>
</tr>
<tr>
<td></td>
<td>that is generated using the JMS Event Schema Manager tool.</td>
</tr>
<tr>
<td>&lt;mergedRowid&gt;</td>
<td>List of ROWID_OBJECT values for the losing records in the merge. This field</td>
</tr>
<tr>
<td></td>
<td>is included in messages for Merge events only.</td>
</tr>
</tbody>
</table>

**JMS Message Format Example**

```xml
<xml version="1.0" encoding="UTF-8"?>
<siperianEvent>
  <eventMetadata>
    <eventType>Update</eventType>
    <baseObjectUid>BASE_OBJECT.CUSTOMER</baseObjectUid>
    <packageUid>PACKAGE.CUSTOMER_PKG</packageUid>
    <messageDate>2008-04-24T15:35:51.000-07:00</messageDate>
    <orsId>localhost-mrm-CM2_ORS</orsId>
    <triggerUid>MESSAGE_QUEUE_RULE.UpdateTrigger</triggerUid>
  </eventMetadata>
  <updateEvent>
    <sourceSystemName>TestSystem123</sourceSystemName>
    <sourceKey>123-1</sourceKey>
    <eventDate>2008-04-24T15:35:51.000-07:00</eventDate>
    <rowid>1</rowid>
    <xrefKey>
      <systemName>Admin</systemName>
      <sourceKey>SV1.161</sourceKey>
    </xrefKey>
    <xrefKey>
      <systemName>System1</systemName>
    </xrefKey>
  </updateEvent>
</siperianEvent>
```
JMS Message Queues for Asynchronous SIF Invocations

This section describes the JMS message queues that Informatica MDM Hub uses to process asynchronous SIF service invocations.

About JMS Message Queues

Informatica supports embedded message queues, which uses the JMS providers that come with application servers. An embedded message queue uses the JNDI name of ConnectionFactory and queue to connect with JMS queue. It requires those JNDI names that have been set up by the application server.

Note: Correctly-configured message queues are essential to a fully-functioning Informatica MDM Hub installation. The Informatica installer automatically sets up message queues and connection factory configuration. If you need to manually configure your message queues or connection factories for testing or troubleshooting purposes, see the Informatica MDM Hub Installation Guide for your platform.

Architecture of JMS Message Queue Used with SIF

You can use asynchronous inbound message queues to handle the asynchronous processing of Informatica MDM Hub service invocations, as shown in the following figure.

The Services Integration Framework (SIF) uses a message-driven bean (MDB) on the JMS queue (named $i$perian.sif.jms.queue) to process incoming asynchronous SIF requests. This message queue and the connection factory (named $i$perian.mrm.jms.xaconnectionfactory) is set up during the installation process, as described in the Informatica MDM Hub Installation Guide for your platform.

For JMS inbound asynchronous requests, the process is very similar to XML over HTTP--you have two options:

- place messages directly on the queue for the web services request
- or when you are making an SIF API call, run it asynchronously: SIF returns a response, then places the request on the incoming JMS queue.

If you are using asynchronous calls, you can view the response published on the outbound JMS queue; the outbound JMS queue is specified in the request (not the JMS queue that we’re publishing to). This JMS queue is
not maintained by Informatica MDM Hub. If you’re passing the request parameter asynchronously and using a JMS queue, you must listen to the specific queue to get the response.

Run-time Processing of Asynchronous SIF Service Invocations

You can use an outbound message queue as a communication channel to feed data changes back to source systems and/or external applications, as shown in the following figure:

For asynchronous SIF service invocations:

1. An external application sends a message containing an Informatica MDM Hub service invocation request to the siperian.sif.jms.queue queue.
   
   You can configure a message rule to control data going to the C_REPOS_MQ_DATA_CHANGE table. For more information, refer to the “Informatica MDM Hub Processes” and “Configuring the Publish Process” chapters in the Informatica MDM Hub Administrator Guide.
   
   Note: The siperian.sif.jms.queue name is reserved by the system. You cannot use this name when creating your own inbound message queues.

2. The application server polls the queue for messages.

3. The Message Driven Bean (MDB) inside the Hub Server forwards the service request to the Informatica MDM Hub for processing.

4. Informatica MDM Hub processes the request and a response is placed on the specified JMS response queue (if any).

5. The external application retrieves the message from the specified message queue and processes it.

Using the Security Access Manager (SAM) with SIF

Informatica MDM Hub Security Access Manager (SAM) is Informatica’s comprehensive security framework for protecting Informatica MDM Hub resources from unauthorized access. At run time, SAM enforces your organization’s security policy decisions for your Informatica MDM Hub implementation, handling user authentication and access authorization according to your security configuration.

Note: SAM security applies primarily to users of third-party applications who want to gain access to Informatica MDM Hub resources. SAM applies only tangentially to Hub Console users. The Hub Console has its own security mechanisms to authenticate users and authorize access to Hub Console tools and resources.
Using Informatica MDM Hub Metadata Management API

The following example highlights Informatica MDM Hub requests that access metadata using the Services Integration Framework (SIF).

SIF Requests

For detailed descriptions of each request, refer to Chapter 5, “SIF API Reference” on page 46 and review the associated Informatica MDM Hub Javadoc for that API class.

<table>
<thead>
<tr>
<th>SIF API Request</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>ApplyChangeList</td>
<td>Applies a change list to the current repository.</td>
<td>“ApplyChangeList” on page 51</td>
</tr>
<tr>
<td>CreateChangeList</td>
<td>Creates a change list in XML format for the current repository.</td>
<td>“CreateChangeList” on page 56</td>
</tr>
<tr>
<td>GetOrsMetadata</td>
<td>Export metadata to a change list XML file.</td>
<td>“GetOrsMetadata” on page 64</td>
</tr>
<tr>
<td>ValidateChangeList</td>
<td>Validates a change list against the current repository.</td>
<td>“ValidateChangeList” on page 80</td>
</tr>
<tr>
<td>ValidateMetadata</td>
<td>Validates the metadata for the current repository.</td>
<td>“ValidateMetadata” on page 80</td>
</tr>
</tbody>
</table>

SIF Metadata Management Example

// See com.siperian.sif.client.sample.SiperianClientExamples.java
// code examples in the SIF SDK for an example of how to set up
// the sifClient object (of type SiperianClient)

//createChangeList
CreateChangeListRequest createChangeListRequest = new CreateChangeListRequest();
createChangeListRequest.setUserName("admin");
createChangeListRequest.setPassword(new com.siperian.sif.message.Password("admin"));
createChangeListRequest.setOrsId("localhost-orcl-CMX_TGT");
createChangeListRequest.setSourceRepositoryId("localhost-orcl-CMX_SRC");

CreateChangeListResponse createChangeListResponse =
(CreateChangeListResponse)sifClient.process(createChangeListRequest);
System.out.println("change list xml: " + createChangeListResponse.getChangeListXml());

//validateChangeList
ValidateChangeListRequest validateChangeListRequest = new ValidateChangeListRequest();
validateChangeListRequest.setUserName("admin");
validateChangeListRequest.setPassword(new com.siperian.sif.message.Password("admin"));
validateChangeListRequest.setOrsId("localhost-orcl-CMX_TGT");
Using Transactions in the EJB Protocol

Informatica MDM Hub runs within an environment that supports EJ Bs. This means that Informatica MDM Hub requests can take place within transactions.

Note: Transactions are only available using the EJB protocol (that is, using a Java application, not using web services).
Transactions

A transaction is a set of procedures or operations that must all complete without generating an error, or return the application to the original state (prior to the transaction).

Composite Services

A composite service is one that requires many requests to create the appropriate response object. For example, a service that returns the complete client profile is created from a profile, one or more addresses, emails, and phone numbers.

The transactions for the composite service can be enabled by changing the SiperianClient properties to use EJBs as the underlying interaction protocol. The composite service simply sends message to trigger a bunch of server side of services through service calls. Note that the Hub does not control transactions for external composite services; instead, these must be managed by your service.

Example

If you are using the EJB protocol, then transactions are available. The SifClient object that you get when using EJB is an instance of EjbSifClient. This object has methods to get the transaction control object:

```
UserTransaction tx = ((EjbSiperianClient)sifClient).createTX(30)
```

Then you can manage the transaction as:

```
tax.begin();
// sif api calls
tax.commit();
```

or

```
tax.rollback()
```
Chapter 4

Using the Security Access Manager with SIF API

This chapter includes the following topics:

- Overview of Using the Security Access Manager, 39
- About SAM and SIF, 39
- Setting Permissions for Specific Roles and Users, 40

Overview of Using the Security Access Manager

This chapter explains the additional requirements the Security Access Manager (SAM) imposes on an application that uses the SiperianClient SIF API. It provides information about permissions to be set for the different roles and users using applications created with SIF API.

About SAM and SIF

If you use an application written with the SIF API requests and also implement SAM, you must ensure that the user using an application has the appropriate permissions. To learn more about setting permissions, see the Security chapter in the Informatica MDM Hub Administrator Guide.

Note: Only admin users can access private resources through SIF requests.

SAM is intended to be relatively transparent to SIF developers. Consider the following:

- Any user using an application that uses SIF API calls to a Informatica MDM Hub implementation that has SAM configured must have the appropriate permissions to access the objects that the SIF calls require. For the specific permissions required, see Chapter 4, “Using the Security Access Manager with SIF API” on page 39. To learn more about granting those permissions, see the Informatica MDM Hub Administrator Guide.

- In some cases, permissions are applied at the column level. An example of this would be if a user is performing Get and Put requests. (Note that column-level privileges are granted using the base object.) In this example the user has the following permissions on a package named P_CUST:
  - READ on column 1
  - READ/CREATE column 2
  - no rights for column 3
For a Get request, only data from columns 1 and 2 would be displayed (column 3 would be effectively hidden). For a Put request, the user would be able to create a new record by inserting data into column 2. In this example the user would have no write access to columns 1 and 3. In addition, the user would have no UPDATE privileges on any of the columns.

For a user to be able to put (insert) a record, they must have CREATE privileges; to update a record, a user must have UPDATE privileges, or the user will get an error. The user can have CREATE/UPDATE only on particular columns, and then the user can only write to those columns. For instance a Put (insert) would fail due to permissions, when a user does not have CREATE privileges on a nullable/non-nullable columns.

Setting Permissions for Specific Roles and Users

The SIF API calls use the underlying Informatica MDM Hub objects and metadata. In order for the call to complete successfully, the user must have permission to access those objects.

Use the following general procedure to ensure that the user using an application has the appropriate permissions:

1. Configure the required resource as secure (versus private). The secure permission makes the object available to the application.
   
   You can set the permissions using the Secure Resources tool.

2. Use the Roles tool to define a role to access that resource. This role includes a set of access privileges to the various objects listed in "Object Types" on page 40.

   For additional information, refer to the Security chapter in the Informatica MDM Hub Administrator Guide.

3. Use the Users and Groups tool to associate the role with a specific user.

Object Types

In the following tables, the abbreviations indicate the object or type of object for which the user requires permission in order to successfully use this API. The degree of permission required is also indicated. The object types referenced by the subsequent permission tables are defined here:

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Audit table</td>
</tr>
<tr>
<td>B</td>
<td>Base object</td>
</tr>
<tr>
<td>BG</td>
<td>Batch group</td>
</tr>
<tr>
<td>C</td>
<td>Column</td>
</tr>
<tr>
<td>F</td>
<td>Function</td>
</tr>
<tr>
<td>H</td>
<td>History</td>
</tr>
<tr>
<td>HP</td>
<td>HM profile</td>
</tr>
<tr>
<td>M</td>
<td>Mapping</td>
</tr>
<tr>
<td>MD</td>
<td>Metadata</td>
</tr>
</tbody>
</table>
### Batch Group API Required Permissions

The Batch Group API calls require these permissions:

<table>
<thead>
<tr>
<th>API</th>
<th>Read</th>
<th>Update</th>
<th>Create</th>
<th>Merge</th>
<th>Delete</th>
<th>Design</th>
<th>Execute</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExecuteBatchGroup</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BG</td>
</tr>
<tr>
<td>GetBatchGroupStatus</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BG</td>
</tr>
<tr>
<td>ResetBatchGroup</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BG</td>
</tr>
</tbody>
</table>

### Data Steward API Required Permissions

The Data Steward APIs require these permissions.

**Note:** This matrix specifies a set of resource types that may be used with the certain SIF requests, but does not specify the exact logical formula—that is, “P,B,C” can mean [P or B] and [C or B], or it can mean any other logical combination.

<table>
<thead>
<tr>
<th>API</th>
<th>Read</th>
<th>Update</th>
<th>Create</th>
<th>Merge</th>
<th>Delete</th>
<th>Design</th>
<th>Execute</th>
</tr>
</thead>
<tbody>
<tr>
<td>AcceptUnmatchedRecordsAsUnique</td>
<td>P, B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AssignUnmergedRecords</td>
<td>P^1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CanUnmergeRecords</td>
<td>P, B, X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ClearAssignedUnmergedRecords</td>
<td>P, B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetAssignedRecords</td>
<td>P, C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetLookupValue</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Data Retrieval API Required Permissions

The Data Retrieval APIs require these permissions:

<table>
<thead>
<tr>
<th>API</th>
<th>Read</th>
<th>Update</th>
<th>Create</th>
<th>Merge</th>
<th>Delete</th>
<th>Design</th>
<th>Execute</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetBvt</td>
<td>P, C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetSearchResults</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SearchMatch</td>
<td>P, MR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SearchQuery</td>
<td>P, C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. GetSearchResult always requires READ access. Objects depend on the primary processor involved.

Data API Required Permissions

The Data API calls require these permissions:

<table>
<thead>
<tr>
<th>API</th>
<th>Read</th>
<th>Update</th>
<th>Create</th>
<th>Merge</th>
<th>Delete</th>
<th>Design</th>
<th>Execute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cleanse</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>F, M</td>
</tr>
<tr>
<td>Link</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P</td>
</tr>
</tbody>
</table>
### Data Update / Insert API Required Permissions

The Data Update / Insert APIs require these permissions:

<table>
<thead>
<tr>
<th>API</th>
<th>Read</th>
<th>Update</th>
<th>Create</th>
<th>Merge</th>
<th>Delete</th>
<th>Design</th>
<th>Execute</th>
</tr>
</thead>
<tbody>
<tr>
<td>MultiMerge</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unlink</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Hierarchy Manager API Required Permissions

The Hierarchy Manager APIs require these permissions:

<table>
<thead>
<tr>
<th>API</th>
<th>Read</th>
<th>Update</th>
<th>Create</th>
<th>Merge</th>
<th>Delete</th>
<th>Design</th>
<th>Execute</th>
</tr>
</thead>
<tbody>
<tr>
<td>CleansePut</td>
<td></td>
<td>B,C,D,M</td>
<td>B,C,D,M</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Merge</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Put</td>
<td></td>
<td>P,C</td>
<td>P,C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tokenize</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unmerge</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AddRelationship</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>HP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeleteRelationship</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>HP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetEntityGraph</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>HP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetOneHop</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>HP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SearchHmQuery</td>
<td></td>
<td>P, C, HP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UpdateRelationship</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>HP</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Merge Workflow API Required Permissions

The Merge Workflow APIs require these permissions:

<table>
<thead>
<tr>
<th>API</th>
<th>Read</th>
<th>Update</th>
<th>Create</th>
<th>Merge</th>
<th>Delete</th>
<th>Design</th>
<th>Execute</th>
</tr>
</thead>
<tbody>
<tr>
<td>AcceptUnmatchedRecordsAsUnique</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P, B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AssignUnmergedRecords</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Metadata API Required Permissions

The Metadata APIs require these permissions:

<table>
<thead>
<tr>
<th>API</th>
<th>Read</th>
<th>Update</th>
<th>Create</th>
<th>Merge</th>
<th>Delete</th>
<th>Design</th>
<th>Execute</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanUnmergeRecords</td>
<td>P, B, X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ClearAssignedUnmergedRecords</td>
<td>P, B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetAssignedRecords</td>
<td>P, C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetUnmergedRecordCount</td>
<td>P, B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ReassignRecords</td>
<td>P, B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Metadata Management API Required Permissions

The Metadata Management APIs require these permissions:

<table>
<thead>
<tr>
<th>API</th>
<th>Read</th>
<th>Update</th>
<th>Create</th>
<th>Merge</th>
<th>Delete</th>
<th>Design</th>
<th>Execute</th>
</tr>
</thead>
<tbody>
<tr>
<td>DescribeSiperianObject</td>
<td>P, RP, B, D, C, F, M, MR, HP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetOrsList (not protected)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ListSiperianObjects</td>
<td>MD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Metadata Manager API Required Permissions

The Metadata Manager API requires these permissions:

<table>
<thead>
<tr>
<th>API</th>
<th>Read</th>
<th>Update</th>
<th>Create</th>
<th>Merge</th>
<th>Delete</th>
<th>Design</th>
<th>Execute</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetOrsMetadata</td>
<td>MD_Man</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## State Management API Required Permissions

The State Management APIs require these permissions:

<table>
<thead>
<tr>
<th>API</th>
<th>Read</th>
<th>Update</th>
<th>Create</th>
<th>Merge</th>
<th>Delete</th>
<th>Design</th>
<th>Execute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>B, if deleting BO record</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X if deleting XREF record</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>B, X if deleting BO and</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>XREF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PromotePendingXrefs</td>
<td>P, C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Restore</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>B, X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## User Management API Required Permissions

The User Management API calls require these permissions:

<table>
<thead>
<tr>
<th>API</th>
<th>Read</th>
<th>Update</th>
<th>Create</th>
<th>Merge</th>
<th>Delete</th>
<th>Design</th>
<th>Execute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authenticate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(none required)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SetPassword</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(none required)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Miscellaneous API Required Permissions

The Informatica MDM Hub miscellaneous APIs require these permissions:

<table>
<thead>
<tr>
<th>API</th>
<th>Read</th>
<th>Update</th>
<th>Create</th>
<th>Merge</th>
<th>Delete</th>
<th>Design</th>
<th>Execute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audit</td>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetSiperianObjectCompatibility</td>
<td>MD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RegisterUsers</td>
<td>U</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UnregisterUsers</td>
<td>U</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SIF API Reference

This chapter includes the following topics:

- Functional SIF API Listing, 46
- Reference SIF API Listing, 50

Functional SIF API Listing

Here are the various Informatica SIF API requests organized by function.

<table>
<thead>
<tr>
<th>SIF Functional Group/Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Batch Group APIs</strong></td>
<td></td>
</tr>
<tr>
<td>Batch Group API requests enable developers to execute batch groups directly without using the MDM Hub Console or stored procedures.</td>
<td></td>
</tr>
<tr>
<td>“ExecuteBatchGroup” on page 58</td>
<td>Executes a set of batch jobs together, some sequentially and some in parallel according to the configuration.</td>
</tr>
<tr>
<td>“GetBatchGroupStatus” on page 60</td>
<td>Get status of most recent execution; polls for status after executing asynchronously.</td>
</tr>
<tr>
<td>“ResetBatchGroup” on page 72</td>
<td>Finds the last execution status of the given batch group, and if its status is failed, sets it to incomplete.</td>
</tr>
<tr>
<td><strong>Data Steward APIs</strong></td>
<td></td>
</tr>
<tr>
<td>Data Steward API requests are intended to make it easy for a developer to write applications with a custom user interface for data stewards. Note that this is not exclusive; you can use any SIF API requests your specific application requires, these are just for convenience.</td>
<td></td>
</tr>
<tr>
<td>“GetLookupValue” on page 61</td>
<td>Retrieves the lookup display name (lookup code description) for the specific lookup values (lookup codes) on specified lookup columns.</td>
</tr>
<tr>
<td>“GetLookupValues” on page 62</td>
<td>Retrieves the list of valid lookup values (lookup codes) and lookup display names (lookup code descriptions) for the specified lookup columns.</td>
</tr>
<tr>
<td>“GetMatchedRecords” on page 62</td>
<td>Retrieves the match candidates for the specified record.</td>
</tr>
<tr>
<td>SIF Functional Group/Class</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>“GetMergeHistory” on page 63</td>
<td>Retrieves a tree representing the history of merges for a specified base object record.</td>
</tr>
<tr>
<td>“GetSystemTrustSettings” on page 65</td>
<td>Retrieves the system-specific trust settings for the specified columns.</td>
</tr>
<tr>
<td>“GetTrustGraphData” on page 66</td>
<td>Retrieves the data to plot the trust decay curve for the specified trust setting.</td>
</tr>
<tr>
<td>“GetTrustScore” on page 66</td>
<td>Retrieves the current trust score for the specified column in a base object record.</td>
</tr>
<tr>
<td>“SearchLookupValues” on page 73</td>
<td>Searches for lookup values that match a given lookup display name (lookup code description).</td>
</tr>
<tr>
<td>“SetRecordState” on page 76</td>
<td>Sets the record state of base object records identified by the specified keys.</td>
</tr>
</tbody>
</table>

**Data APIs**

Data API requests enable developers to execute Informatica MDM Hub Cleanse, Link, MultiMerge, and Unlink base object requests.

| “Cleanse” on page 54                    | Uses cleanse functions defined in the Informatica MDM Hub to transform an input record provided in the request to the output format specified by the cleanse function selected. |
| “Link” on page 67                      | Links two or more base object records using the specified groupRecordKey as the group ID. |
| “MultiMerge” on page 69                | Merges multiple base object records that have been identified as representing the same object and allows specifying the field level overrides for the merged record. |
| “Unlink” on page 77                    | Unlinks two or more base object records with the group id specified in the groupRecordKey field. |

**Data Update / Insert APIs**

Data update / insert API requests enable developers to execute data updates and inserts on base object records.

<p>| “AddRelationship” on page 51           | Enables you to add a relationship between two entities.                     |
| “CleansePut” on page 54                | Inserts or updates a single record identified by a key into a base object. |
| “DeleteRelationship” on page 57       | Deletes a relationship between two entities, by making it inactive and setting the end date to the current date. This does not remove the record from the relationship table. If the relationship is a foreign key relationship rather than a record in a relationship table, the request sets the foreign key value to null. |
| “Merge” on page 68                     | Merges two base object records that have been identified as representing the same object. |
| “Put” on page 70                      | Inserts or updates a single record identified by a key into a base object.  |</p>
<table>
<thead>
<tr>
<th>SIF Functional Group/Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Tokenize&quot; on page 77</td>
<td>Generates match tokens for a base object record that has been updated or inserted.</td>
</tr>
<tr>
<td>&quot;Unmerge&quot; on page 77</td>
<td>Unmerges base object (BO) records.</td>
</tr>
<tr>
<td>&quot;UpdateRelationship&quot; on page 79</td>
<td>Hierarchy Manager request for changing some characteristics of an existing relationship.</td>
</tr>
<tr>
<td>Data Retrieval APIs</td>
<td>Data Retrieval API requests enable developers to retrieve data, including BVT, a single record or sets of records, as well as to perform searches based on match columns.</td>
</tr>
<tr>
<td>&quot;GetBvt&quot; on page 60</td>
<td>Retrieves the best version of truth (BVT) from the specified package using a known key.</td>
</tr>
<tr>
<td>&quot;Get&quot; on page 58</td>
<td>Retrieves a single record from the specified package using a known key.</td>
</tr>
<tr>
<td>&quot;GetEntityGraph&quot; on page 61</td>
<td>Hierarchy Manager request for fetching a graph of entities and relationships related to a specified set of entities.</td>
</tr>
<tr>
<td>&quot;GetOneHop&quot; on page 63</td>
<td>Hierarchy Manager request for fetching information about the entities directly related to a specified group of entities in a specified HM configuration.</td>
</tr>
<tr>
<td>&quot;GetSearchResults&quot; on page 64</td>
<td>Retrieves additional data when the number of records found by the SIF API search queries (SearchMatch, SearchQuery) exceeds the number of records to return specified in the search API request.</td>
</tr>
<tr>
<td>&quot;SearchHmQuery&quot; on page 73</td>
<td>Provides search capabilities for Hierarchy Manager.</td>
</tr>
<tr>
<td>&quot;SearchMatch&quot; on page 74</td>
<td>Searches for records in a package based on match columns and rule definitions.</td>
</tr>
<tr>
<td>&quot;SearchQuery&quot; on page 74</td>
<td>Retrieves a set of record from an MRM package or a remote package satisfying the specified criteria.</td>
</tr>
<tr>
<td>Task APIs</td>
<td>Task APIs are used for task administration.</td>
</tr>
<tr>
<td>GetTasks</td>
<td>Retrieves lists of tasks and task details.</td>
</tr>
<tr>
<td>GetTasksLineage</td>
<td>Retrieves the lineage (history) of the specified task.</td>
</tr>
<tr>
<td>CreateTask</td>
<td>Creates a task.</td>
</tr>
<tr>
<td>UpdateTask</td>
<td>Updates a task.</td>
</tr>
<tr>
<td>Merge Workflow APIs</td>
<td>Merge Workflow API requests enable developers to execute post-match batch processes, such as search for unmatched or unmerged records.</td>
</tr>
<tr>
<td>&quot;AcceptUnmatchedRecordsAsUnique&quot; on page 50</td>
<td>Once the match batch process has been run and records have been placed into match groups, there are often records that did not match any other records in the Hub. Sets the</td>
</tr>
<tr>
<td>SIF Functional Group/Class</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>unmatched records to unique (that is, sets CONSOLIDATION_IND=1)</td>
</tr>
<tr>
<td>&quot;AssignUnmergedRecords&quot; on page 51</td>
<td>Once the match batch process has been run and records have been placed into match groups, the records that were processed and not automatically merged are placed into the UNMERGED state. This is to assign the unmerged records to specified user.</td>
</tr>
<tr>
<td>&quot;CanUnmergeRecords&quot; on page 53</td>
<td>Determines whether the specified cross reference (XREF) record can be unmerged from the consolidated base object.</td>
</tr>
<tr>
<td>&quot;ClearAssignedUnmergedRecords&quot; on page 56</td>
<td>Clears the list of unmerged records that are currently assigned to this user.</td>
</tr>
<tr>
<td>&quot;GetAssignedRecords&quot; on page 59</td>
<td>Get a set of records requiring manual merge decisions that are assigned to the user.</td>
</tr>
<tr>
<td>&quot;GetUnmergedRecordCount&quot; on page 67</td>
<td>Get the number of unmerged records.</td>
</tr>
<tr>
<td>&quot;ReassignRecords&quot; on page 71</td>
<td>Reassigns the specified records assigned for manual merge evaluation to another user.</td>
</tr>
<tr>
<td>Metadata APIs</td>
<td>Metadata API requests enable developers to return metadata for specified objects.</td>
</tr>
<tr>
<td>&quot;DeleteRelationship&quot; on page 57</td>
<td>Request to describe Informatica objects by fetching their metadata.</td>
</tr>
<tr>
<td>&quot;GetOrsList&quot; on page 64</td>
<td>Retrieves a list of operational record stores (ORS) registered in the master database.</td>
</tr>
<tr>
<td>&quot;ListSiperianObjects &quot; on page 67</td>
<td>Returns metadata of Informatica MDM Hub objects.</td>
</tr>
<tr>
<td>Metadata Management APIs</td>
<td>Metadata Management API requests enable developers to manage ORS change lists.</td>
</tr>
<tr>
<td>&quot;ApplyChangeList&quot; on page 51</td>
<td>Applies a change list to the current repository.</td>
</tr>
<tr>
<td>&quot;CreateChangeList&quot; on page 56</td>
<td>Creates a change list in XML format for the current repository.</td>
</tr>
<tr>
<td>&quot;ValidateChangeList&quot; on page 80</td>
<td>Validates a change list against the current repository.</td>
</tr>
<tr>
<td>&quot;ValidateMetadata&quot; on page 80</td>
<td>Validates the metadata for the current repository.</td>
</tr>
<tr>
<td>Metadata Manager APIs</td>
<td>Metadata Manager API requests enable developers to export metadata.</td>
</tr>
<tr>
<td>&quot;GetOrsMetadata&quot; on page 64</td>
<td>Export metadata to a change list XML file.</td>
</tr>
<tr>
<td>State Management APIs</td>
<td>State Management API requests enable developers to delete and restore state-enabled records with state set to DELETE, as well as promote pending XREF records.</td>
</tr>
<tr>
<td>&quot;Delete&quot; on page 56</td>
<td>Deletes the specified record(s) from the Hub.</td>
</tr>
<tr>
<td>SIF Functional Group/Class</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>“PromotePendingXrefs” on page 69</td>
<td>Promotes or flags for promotion the XREF records specified in the request.</td>
</tr>
<tr>
<td>“Restore” on page 72</td>
<td>Restores the specified XREF record(s) in the Hub.</td>
</tr>
<tr>
<td><strong>User Management APIs</strong></td>
<td>User Management API requests enable developers to manage user security.</td>
</tr>
<tr>
<td>“Authenticate” on page 53</td>
<td>Authenticates a user against the specified ORS.</td>
</tr>
<tr>
<td>“SetPassword” on page 76</td>
<td>Changes a user’s password to a new password.</td>
</tr>
<tr>
<td><strong>Miscellaneous APIs</strong></td>
<td>These miscellaneous API requests enable developers to execute audit requests, register and unregister users, and perform other compatibility requests.</td>
</tr>
<tr>
<td>“Audit” on page 52</td>
<td>Add a custom entry to the Hub Audit trail.</td>
</tr>
<tr>
<td>“GetSiperianObjectCompatibility” on page 65</td>
<td>Request to get a checksum that represents the definition of the specified object in Informatica MDM Hub.</td>
</tr>
<tr>
<td>“RegisterUsers” on page 72</td>
<td>Allows for automated provisioning of users that are authenticated externally using one of the registered JAAS login modules.</td>
</tr>
<tr>
<td>“UnregisterUsers” on page 78</td>
<td>Allows for previously provisioned users (see RegisterUsers) to be unregistered.</td>
</tr>
</tbody>
</table>

### Reference SIF API Listing

This section is an alphabetical listing for the SIF API. It provides a description and use case examples for the various SIF API requests. Refer to the SIF Javadocs for details of how to use these API requests with the interfaces that the Informatica Java client provides. If you are using a Web service interface to the requests, refer to the Web Services Description Language (WSDL) descriptions of the Informatica Web service.

**Note:** Only admin users can access private resources through SIF requests.

#### AcceptUnmatchedRecordsAsUnique

AcceptUnmatchedRecordsAsUnique changes the state of records that have no match candidates from Unmerged to Consolidated (unique). Once a record is in the Consolidated state, it will no longer appear in the list of records that needs to be reviewed and it will not be merged by the merge batch process. These records can still be merged manually in the Console or by using the Merge API.

The request specifies the base object table or a package on that table. It also supplies a boolean value indicating whether or not to change only those records assigned to the user.

The response contains the number of records accepted as unique.

**Note:** You can configure AcceptUnmatchedRecordsAsUnique requests only for “no system” when using the Hub Console Audit Manager to audit requests made by external applications. Once auditing for a particular SIF API
request is enabled, Informatica MDM Hub captures each SIF request invocation and response in the audit log. For more information, refer to the Informatica MDM Hub Administrator Guide.

Use Case

This is the common scenario for using the AcceptUnmatchedRecordsAsUnique request:

- **Set unmatched records as unique** — You can use the AcceptUnmatchedRecordsAsUnique request in an application with a custom UI for the data stewards. In the screen that manages the status of records, you might create a button that uses this request to accept all the unmatched records as unique.

AddRelationship

AddRelationship enables you to add a relationship between two entities.

**Note:** This API request applies to Hierarchy Manager. If you have not purchased, configured, and populated Hierarchy Manager, this request will fail.

The request identifies the HM configuration and hierarchy, the relationship type, the records, and a number of optional parameters. Note that this request cannot be used to add a new Relationship with Foreign Key Relationship Type because adding a FK Relationship really involves updating an existing record in the FK Relationship Base Object. For more information, see “UpdateRelationship” on page 79.

The response contains the record key for the added relationship. Informatica MDM Hub infers the types of the entities being related (and thus the base objects containing those entities) from the relationship type.

Use Case

This is the common scenario for using the addRelationship request:

- **Add a relationship between two HM entities** — If you have Hierarchy Manager and have populated it with entities, you can use the addRelationship request to create a relationship between two entities.

Related SIF Requests

- "UpdateRelationship” on page 79
- “DeleteRelationship” on page 57

ApplyChangeList

ApplyChangeList request applies the specified change list to the current repository (ORS). It executes all of the changes in the change list. If an error occurs during the change list execution process, whether or not any changes are applied to the target depends on the rollbackStrategy field:

- **FULL_ROLLBACK**—no changes will have been executed
- **ROLLBACK_TO_LAST_CHANGE**—only the changelist item that failed with be rolled back (that is, all other changes will have been executed).

AssignUnmergedRecords

AssignUnmergedRecords assigns records in the unmerged state to the specified user. It assigns no more than the requested number of records. Optionally, you can specify a WHERE clause to select Unmerged records from the package. The Unmerged state is equivalent to setting the consolidation indicator to 2 and can also be referred to as the “ready to merge” state. Records are placed into the Unmerged state regardless of whether they matched other records or not. This request is used to assign the records that are in the Unmerged state to a specified user for review and processing. However, any records that are already assigned to a user will not be reassigned by this API.
The response contains the number of records assigned.

Note: Hub Implementers can setup user exits that control how records are assigned. These user exits are invoked when this API is run and will override the standard logic for assignment of records. For information regarding user exits, see Informatica MDM Hub Administrator Guide.

Use Case

This is the common scenario for using the AssignUnmergedRecords request:

- Assigning unmerged records to a user—You can use the AssignUnmergedRecords request in an application with a custom UI for the data stewards. In the screen that manages the data steward’s queues, you might create a button that uses this request to assign unmerged records.

Audit

Audit adds an entry to the C_REPOS_AUDIT table to record information about some activity involving a record stored in Informatica MDM Hub. You can log similar information about information in your own application programs.

Set the attributes of the new entry (for example, component, action, status, context). Then process the request to add the entry to the audit table. The process method returns an AuditResponse, which contains the rowid of the resulting audit record.

To use this facility, store the name of a project or similar large entity in component, and let action be an element of the component. For example, component might be “SIF API” and action might be AuditRequest.

You can set the audit rowid of the last previous related audit entry. In this way you can build a chain of audit entries. You obtain the rowid of an audit entry from the AuditResponse that comes back when you process an AuditRequest.

Use the status field to convey information useful for determining what to do with the audit record. For example, status values might be debug, info, warn, error, and fatal.

Use the contextXML and dataXML to add XML-formatted additional information to the audit entry.

Note: You can not configure Audit API requests to audit requests made by external applications. For more information, refer to the Informatica MDM Hub Administrator Guide.

Use Case

This is the common scenario for using the Audit request:

- Adding auditing information to the log—You can use the audit request in an application to record auditing information in the log for reporting or compliance purposes.

Usage Example

```java
// For example, if this is in a Servlet that receives an XML
// to update multiple Hub packages.

AuditRequest request = new AuditRequest();
request.setComponent("mycompany.customerServlet");
request.setAction("POST");
request.setStatus("info");
// from: the same system to be used in other SIF calls
request.setFromSystem("CRM");
request.setToSystem("Admin"); // to: Siperian Hub

// context: any metadata to help understand the entry
request.setContext( dataId ); // example: pKeySource
// context xml: complex metadata, for debug, may impact performance
request.setContextXML("<metadata>
  + "url" + httpServletRequest.getRequestURI() + "</url>
  + "</metadata>";
```
Authenticate

Authenticate allows you to determine a user’s rights to access an ORS. If the user has the right to access the ORS, the message in the response object is STATUS_GRANTED. Otherwise it is STATUS_DENIED. The response contains a list of the roles assigned to the user and information about the user’s password—if and when it expires and whether it is externally authenticated using a service such as LDAP.

Use Case

This is the common scenario for using the authenticate request:

- **Determine a user’s access rights to an ORS**—Before using a request that requires specific access privileges, you can use authenticate to determine if the user possesses the required rights.

CanUnmergeRecords

CanUnmergeRecords determines whether or not specified records can be unmerged from the consolidated base object. The request contains a package and a key identifying the XREF to unmerge. The response contains a boolean value that is true if the records can be unmerged, false if they cannot.

Cross reference records can be added to a base object record either by consolidating two base object records or by adding them directly using the ROWID_OBJECT of a base object record. If a cross reference is added using the ROWID_OBJECT and no PKEY_SOURCE_OBJECT, and there is not already a cross reference for that base object record for the specified system, a new cross reference record is added that is considered an “edit” cross reference.

An unmerge is not allowed if the specified cross reference is not an edit cross reference and all the other cross references for that base object are edit cross references. If there are at least two cross references that are not edit cross references, the cross reference can be unmerged.

**Note:** You can configure CanUnmergeRecords requests according to a specific system when using the Hub Console Audit Manager to audit requests made by external applications. Once auditing for a particular SIF API request is enabled, Informatica MDM Hub captures each SIF request invocation and response in the audit log. For more information, refer to the Informatica MDM Hub Administrator Guide.

Use Case

This is the common scenario for using the CanUnmergeRecords request:

- **Determining whether a given record can be unmerged**—You can use the CanUnmergeRecords request in an application with a custom UI for the data stewards to determine whether two records can be unmerged before attempting to do so.
Cleanse

Cleanse invokes a cleanse function defined in Informatica MDM Hub. The request specifies the record and the cleanse function. The response contains a record containing the cleansed data.

Available cleanse functions can be viewed in the Hub Console in the Cleanse Function Manager. Additionally, you can use the ListSiperianObject request to retrieve the list of available cleanse functions. Cleanse function details, including parameters, can be retrieved using the DescribeSiperianObject request.

You can specify the name of the cleanse function to use in a Cleanse request in two ways:

- a cleanse function UID: “CLEANSE_FUNCTION.[Cleanse Library Name][Cleanse Function Name]”
- or “[Cleanse Library Name][Cleanse Function Name]”.

For example, in order to use the Concatenate cleanse function that resides in the String Functions cleanse library, the cleanse function would be identified as either “CLEANSE_FUNCTION.String Functions|Concatenate” or “String Functions|Concatenate”.

Mappings defined in the Hub may also be accessed and used as cleanse functions. Mappings are automatically placed in the “Mappings” library and can be accessed using the UID “CLEANSE_FUNCTION.Mappings|[mapping name]”.

Related Topics:

- “CleansePut” on page 54

Use Cases

These are the common scenarios for using the cleanse request:

- Data cleansing for external applications — An external application can use the cleanse request independently of the Informatica MDM Hub master record functionality. External applications can invoke cleanse to interface with data quality facilities provided by Informatica to process input data.

- Address verification for external applications — Informatica MDM Hub provides the functionality to validate and standardize addresses. These facilities can be used by external applications to improve the quality of the address data that is entered into them.

- Cleanse used in combination with put — The most common use of the cleanse request is to cleanse an individual field before the record is passed to the put request.

- Cleanse used in combination with match — The match request provides access to the matching rules and allows you to search Informatica MDM Hub for records that contain values that are similar, but not necessarily identical to the search criteria. To improve the quality of matches returned, you can cleanse the search criteria before passing them to the match request.

Related SIF Requests

“CleansePut” on page 54

CleansePut

CleansePut combines the functions of the “Cleanse ” on page 54 and “Put ” on page 70 API requests to cleanse the specified record and update or insert it in the specified table in a single request. CleansePut replicates the Stage and Load batch processes that move data from the landing table, through the cleansing process into the staging table and finally into the base object. The physical landing and staging tables are not used by CleansePut.
The input record is based on a mapping, which defines the transformation of data from a landing table structure to a staging table structure. The staging table associated with the mapping determines which base object the resulting data is inserted or updated in.

A staging table is associated with a source system. If the system name isn’t specified, the program uses the staging table source system.

A source key is generated for this record if it hasn’t been specified and the generateSourceKey option is true. The response contains the key of the added or updated record.

**Note:** Both Put and Cleanse Put requests process null values; for example when no value is specified for a field, it will be set to null.

Special characters don’t need to be escaped before making the CleansePut API call. However, if you have custom code that used escaped special characters in the past, you must update your custom code to remove the escaped special characters.

You can configure CleansePut requests in all systems when using the Hub Console Audit Manager to audit requests made by external applications. Once auditing for a particular SIF API request is enabled, Informatica MDM Hub captures each SIF request invocation and response in the audit log. For more information, refer to the *Informatica MDM Hub Administrator Guide*.

**State Management**

When state management is enabled on the base object that will receive the record from the CleansePut operation, the mapping used must adhere to the following rules:

- If a new record is inserted, then the HUB_STATE_IND column of the mapping may (optionally) be used to specify the state of the record to insert. If no state is specified through the mapping, the state is assumed to be ACTIVE.
- If an existing record is being updated by the request, then the HUB_STATE_IND column cannot be used to specify a state change for the existing record. To change the state of an existing record, please reference the javadocs for the following SIF classes: DeleteRequest, RestoreRequest and PromotePendingXrefsRequest.

For more information regarding how to enable state management, refer to *Informatica MDM Hub Data Steward Guide* or *Informatica MDM Hub Administrator Guide*.

**Filtered Request**

A CleansePut request goes through a delta detection process and can also be filtered so that no changes are made to the ORS. For more information, refer to the Javadocs for CleansePut.

**Use Cases**

These are the common scenario for using the CleansePut request:

- **CleansePut used to insert or update base object records in Informatica MDM Hub in a single request** — CleansePut allows for reduced round trips between the client and Informatica MDM Hub, delivering improved performance.

- **CleansePut used with tokenize to insert or update base object records in Informatica MDM Hub and then encode the records for matching** — CleansePut, followed by a call to “Tokenize ” on page 77 cleanses the new row of data, inserts or updates it in the base object and then encodes it for matching. The resulting row is ready for matching. Informatica MDM Hub.

**Note:** When using time values, Informatica MDM Hub rounds all values to the nearest second.

**Related SIF Requests**

“Cleanse ” on page 54, “Put ” on page 70 “Tokenize ” on page 77
ClearAssignedUnmergedRecords

ClearAssignedUnmergedRecords clears a user’s assigned unmerged records for the specified base object, making those records available for assignment to another user.

Note: There are no parameters for this request. All the unmerged records assigned to this user making the request will now be available to be assigned to another user. If there is a specific user that the records should be assigned to the ReassignRecordsRequest should be used.

Use Case

This is the common scenario for using the ClearAssignedUnmergedRecords request:

- Clearing the queue of unmerged records to a given user—You can use the ClearAssignedUnmergedRecords request in an application with a custom UI for the data stewards. In the screen that manages the data steward’s queues, you might create a button that uses this request to remove unmerged records from a user’s queue.

CreateChangeList

CreateChangeList creates a change list in XML format for the current repository (it requires a sourceRepositoryId to be specified as the basis for the comparison). The generated change list contains a list of actions.

Delete

Delete removes the specified record(s) from the Hub. If the deleteBORecord flag is specified then the BO record is deleted even if only a sourceKey and systemName are specified.

State Management

When an XREF record is deleted, the state of the BO record will be calculated as the greatest of the states of its XREFs. The order of precedence for state is ACTIVE, PENDING, DELETED. The following list describes the behavior of this request based on various XREF states:

- Active records will be transitioned to the DELETED state.
- Pending records will be hard deleted.
- Deleted records will remain unchanged.

Use Case

Record A has two XREFs that are ACTIVE. If one of the XREFs is deleted, then the result will be that record A now has one ACTIVE xref and one DELETED XREF. Since the ACTIVE state has higher precedence than the DELETED state, the state of BO record A after the delete operation is ACTIVE. If the remaining ACTIVE XREF is then deleted, record A will have two deleted XREFs and the state of BO record A will be DELETED.

Usage Example

The following example deletes the XREF record with sourceKey=1234 and system=CRM from the package CUSTOMER_UPDATE. If the XREF record is PENDING, it will be hard deleted. If the XREF record is ACTIVE, it will be soft deleted. If the record is already in the DELETED state, the record will remain as is.

Note: Delete throws an exception if you attempt to delete a record that is in the DELETED state.

```java
DeleteRequest request = new DeleteRequest();
RecordKey recordKey = new RecordKey();
recordKey.setSourceKey("1234");
recordKey.setSystemName("CRM");
ArrayList recordKeys = new ArrayList();
recordKeys.add(recordKey);
request.setRecordKeys(recordKeys); // Required
```
DeleteRelationship

DeleteRelationship deletes a relationship between two entities. This request does not remove the record from the relationship table. If the relationship is a foreign key relationship rather than a record in a relationship table, the request sets the foreign key value to null.

This request behaves differently when used with Foreign Key Relationship Types. Since all Relationship records of a Foreign Key Relationship Type use the same End Date, instead of setting the End Date this request sets the foreign key value in the FK Relationship Base Object to null.

The request provides the Hierarchy Manager configuration, the record key, and the relationship type of the relationship to be removed.

Note: This request requires Hierarchy Manager. If you have not purchased, installed, configured Hierarchy Manager, then this request will fail.

Use Case

This is the common scenario for using the DeleteRelationship request:

* Delete a relationship between two HM entities — If you have Hierarchy Manager and have populated it with data, you can use the DeleteRelationship request to delete an existing relationship between two entities.

Related SIF Requests

“PromotePendingXrefs” on page 69, “Restore” on page 72

DescribeSiperianObject

DescribeSiperianObject request retrieves the metadata about Informatica MDM Hub objects.

Each object defined in Informatica MDM Hub has a unique identifier of the form [objectType].[objectName]. For example, PACKAGE.CUSTOMER_READ. The request contains the UIDs of the objects to describe. The response contains a list of metadata objects.

When describing a match column, DescribeSiperianObject returns the following additional fields:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>matchKey</td>
<td>Boolean</td>
<td>True if the match column is the match key for this base object. A base object can only have a single match key.</td>
</tr>
<tr>
<td>fuzzyColumn</td>
<td>Boolean</td>
<td>True if the match column is a fuzzy column</td>
</tr>
<tr>
<td>columns</td>
<td>List</td>
<td>The list of UIDs for the columns that make up this match column. UIDs are formed as COLUMN.bo_name</td>
</tr>
</tbody>
</table>

Optionally, you can ask for the MATCH_KEY of a base object directly by calling DescribeSiperianObject with MATCH_KEY.BO_NAME.

The result from this call is the MATCH_COLUMN object that represents the match key for the specified base object.

MATCH_COLUMN returns:

* whether the match column is the match key
• whether the match column is a fuzzy column
• list of physical columns that make up the match column

When MATCH_KEY. BO_NAME is requested, DescribeSiperianObject returns the match column that is the match key for the specified base object.

Use Case

This is the common scenario for using the DescribeSiperianObject request:

• **Obtaining metadata about an object prior to attempting to manipulate it**—You can use the DescribeSiperianObject request to learn a bit about an object before you attempt to do any operations on it.

**ExecuteBatchGroup**

ExecuteBatchGroup executes a batch group. A batch group is a set of batch jobs executed together, some sequentially and some in parallel according to the configuration. When one job has an error, the group will stop; that is, no more jobs will be started, but running jobs will run to completion. There are two other related services in this request:

• “ResetBatchGroup” on page 72
• “GetBatchGroupStatus” on page 60

**Note:** In addition to these Java APIs and the SOAP and HTTP XML protocols always available for SIF, these three (3) Batch Group requests also have database stored procedures available:

• cmxbg.execute_batchgroup
• cmxbg.reset_batchgroup
• cmxbg.get_batchgroup_status

For more information, see the *Informatica MDM Hub Administrator Guide*.

Use Case

This is the common scenario for using the executeBatchGroup request:

• **ExecuteBatchGroup with getBatchGroupStatus** — After calling ExecuteBatchGroup, wait and then use “GetBatchGroupStatus” on page 60 to see if the batch group executed successfully.

**Related SIF Requests**

“GetBatchGroupStatus” on page 60, “ResetBatchGroup” on page 72

**Get**

Get uses a record key to retrieve a single row of data from the specified package. The row can include data from base objects and from child records (that is, content metadata such as History, Xref, Xref History, and Raw) associated with the base object. You can use this request against both the regular MRM packages (“PACKAGE.” SiperianObjectUid prefix) and the remote packages (“REMOTE_PACKAGE.” SiperianObjectUid prefix).

You can also get lineage and trust information. The trust scores are returned for the package record and the cross reference records. The lineage information returned as indicator on the trust enabled fields indicating whether the specific field of the cross reference record has won over other cross references and is used on the base object.
For MRM packages, you can use this request to retrieve the following types of the content metadata for underlying primary base object of the package and the trust score and the lineage information for the trust enabled columns.

<table>
<thead>
<tr>
<th>SiperianObjectType</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XREF</td>
<td>Cross Reference data. If state management is enabled for the parent of the package, then this option will return only the cross reference records that are in the ACTIVE state.</td>
</tr>
<tr>
<td>PENDING_XREF</td>
<td>Cross Reference data that is in the PENDING state. This option is only valid when state management is enabled for the parent of the package. Otherwise, an exception is thrown.</td>
</tr>
<tr>
<td>DELETED_XREF</td>
<td>Cross Reference data that is in the DELETED state. This option is only valid when state management is enabled for the parent of the package. Otherwise, an exception is thrown.</td>
</tr>
<tr>
<td>XREF_HISTORY</td>
<td>Previous values for each of the underlying cross references of the specified base object. Note: Base object history has to be enabled.</td>
</tr>
<tr>
<td>HISTORY</td>
<td>Previous values for the specified base object record. Note: Base object history has to be enabled.</td>
</tr>
<tr>
<td>RAW</td>
<td>Raw records associated with the specific base object record. Note: Raw retention needs to be enabled on at least one staging table belonging to the specified base object.</td>
</tr>
</tbody>
</table>

If the package is based on a query that joins multiple base objects, content metadata will be returned only for the primary base object.

**Use Case**

This is the common scenario for using the Get request:

- **Get used to retrieve a row and its associated child records**—The most common use of get is to retrieve a single row of data, with any associated child records.

**Usage Example**

The following example gets a record with ROWID_OBJECT key 782 from the package PARTY_ADDRESS_READ_PKG.

```java
GetRequest request = new GetRequest();
RecordKey recordKey = new RecordKey();
recordKey.setRowId("782");
request.setRecordKey(recordKey); //Required
request.setSiperianObjectUID("PACKAGE.PARTY_ADDRESS_READ_PKG"); //Required
GetResponse response = sigClient.process(request);
```

**Related SIF Requests**

“GetSearchResults ” on page 64, “Put ” on page 70, “SearchQuery ” on page 74

**GetAssignedRecords**

GetAssignedRecords fetches the current user’s records that were assigned by an “AssignUnmergedRecords” on page 51 request. Can request records in either the Unmerged or the Onhold state.

The request contains a package, a record state (UNMERGED or ON_HOLD), and a maximum number of records to return. The response contains a set of records and a token to use to fetch more results. Use “GetSearchResults ” on page 64 to get subsequent sets of records.
Use Case
This is the common scenario for using the GetAssignedRecords request:

- **GetAssignedRecords** used to retrieve assigned records for display in the user interface of a custom-designed application—The most common use of GetAssignedRecords is to retrieve the records that are assigned to a specific user for display in a custom-designed UI.

Usage Example
The following example requests the UNMERGED records for the CUSTOMER_UPDATE package that are assigned to the user making this request.

```java
GetAssignedRecordsRequest request = new GetAssignedRecordsRequest();
request.setSuperiorObjectUID("PACKAGE.CUSTOMER_UPDATE");
request.setRecordsToReturn(10);
request.setRecordState(RecordState.UNMERGED);
request.setReturnTotal(true);

GetAssignedRecordsResponse response = (GetAssignedRecordsResponse) sipClient.process(request);
```

Related SIF Requests
"AssignUnmergedRecords" on page 51, "ClearAssignedUnmergedRecords" on page 56

GetBatchGroupStatus
GetBatchGroupStatus returns the status of a batch group; polls for status after executing asynchronously. To learn more about batch groups, see the Informatica MDM Hub Administrator Guide.

**Note:** When making an asynchronous call, the runStatus of 0 (success) means that GetBatchGroupStatus was successfully placed in the async queue. To see the actual runStatus of the batch group, you can also specify a value in the jmsReplyTo field when making the call. The SIF response message containing the run status of the batch group will be returned on this queue. Alternatively, you can also use the Audit Manager in the Hub Console to enable the audit for "No System: GetBatchGroupStatus" and enable the audit XML. Then, use the GetBatchGroupStatus call again and then check C_REPOS_AUDIT:DATA.XML for the SIF response. The response will show the batch group’s “failed” status. For more information regarding the Audit Manager, refer to the Informatica MDM Hub Administrator Guide.

Use Case
This is the common scenario for using the GetBatchGroupStatus request:

- **GetBatchGroupStatus with ExecuteBatchGroup** — After calling "ExecuteBatchGroup" on page 58, wait and then use GetBatchGroupStatus to see if the batch group executed successfully.

Related SIF Requests
"ExecuteBatchGroup" on page 58, "ResetBatchGroup" on page 72

GetBvt
GetBvt retrieves the best version of truth (BVT) from the specified package using a known key. The specified package must have a base object (BO) as its parent and the base object must be a link style BO instead of a merge style BO. This option can be configured in the schema manager of the Hub Console. The BVT is calculated on the set of records belonging to the same link group as the input record key.

**Note:** You can configure GetBvt requests in all systems when using the Hub Console Audit Manager to audit requests made by external applications. Once auditing for a particular SIF API request is enabled, Informatica MDM Hub captures each SIF request invocation and response in the audit log. For more information, refer to the Informatica MDM Hub Administrator Guide.
State Management

You can include pending records in the BVT calculation if state management is enabled on the parent base object by adding setIncludePending(TRUE) to the request. For more information regarding how to enable state management, refer to Informatica MDM Hub Data Steward Guide or Informatica MDM Hub Administrator Guide.

Related SIF Requests

“Get” on page 58, “Link” on page 67 “Unlink” on page 77

GetEntityGraph

GetEntityGraph Hierarchy Manager request fetches a graph of entities and relationships related to a specified set of entities. The Entities and Relationships fetched can be one or multiple hops away from the Entities in the request.

The request supplies the HM configuration, a list of entities, and a filter. The response contains lists of entities and relationships.

Note: This request requires Hierarchy Manager. If you have not purchased, installed, configured Hierarchy Manager, then this request will fail.

Use Case

This is the common scenario for using the GetEntityGraph request:

- Fetch the entities and relationships associated with a specific HM entity or entities — If you have Hierarchy Manager and have populated it with data, you can use GetEntityGraph to get the entities and relationships associated with one or more entities.

Related SIF Requests

“GetOneHop” on page 63

GetLookupValue

GetLookupValue enables an application program to obtain the display value corresponding to a key value for the specified object columns. This API is used to retrieve the user friendly descriptions for specific code values when a package contains only the code value and the developer needs to display the user friendly description of the code in the user interface. This request is also useful when displaying an individual record.

The request contains a list of LookupFields. Each LookupField contains an identifier for the column and a foreign key value.

The response contains a record that has a field for each LookupField. The order of the fields matches the order of the LookupFields in the request. In each field, the name is the lookup (foreign key) value and the value is the lookup display name.

This request is intended to be used together with the “GetLookupValues” on page 62 and “SearchLookupValues” on page 73 requests. The difference between these APIs is that the GetLookupValue API retrieves descriptions only for the specified code values, while the GetLookupValuesRequest and the SearchLookupValuesRequest return the list of valid lookup code values and lookup code descriptions for the specified lookup column.
Use Case

This is the common scenario for using the GetLookupValue request:

- **Fetch the valid values for a particular field and display them in a UI**—In a custom UI, you can use GetLookupValue to fetch a list of valid values for a field. You can then display these values as a set of selections for the user.

Related SIF Requests

“GetLookupValues” on page 62, “SearchLookupValues” on page 73, “DeleteRelationship” on page 57

GetLookupValues

GetLookupValues enables an application program to populate fields of a user interface with a list of values for a given column. This request is similar to the “GetLookupValue” on page 61 request, but the response contains a list of lists rather than a single list.

This request can be used on any foreign key column. A foreign key to a lookup table has a limited set of values. Other foreign keys can have large numbers of possible values. This request is intended and most useful for lookup tables, when you want to display the list of acceptable values to a user.

The response contains a record for each column that has fields with the lookup information. In each field, the name is the lookup (foreign key) value and the value is the lookup display name.

Use Case

This is the common scenario for using the GetLookupValues request:

- **Fetch the valid values for a set of fields and display them in a UI**—In a custom UI, you can use getLookupValues to fetch a list of valid values for a set of fields. You can then display these values as a set of selections for the user.

Related SIF Requests

“GetLookupValue” on page 61, “SearchLookupValues” on page 73, “DeleteRelationship” on page 57

GetMatchedRecords

GetMatchedRecords returns records that are candidates to match a specified record.

The request contains a package and a record. The response contains a collection of potentially matching records from the specified package.

**Note**: You can configure GetMatchedRecords requests in all systems when using the Hub Console Audit Manager to audit requests made by external applications. Once auditing for a particular SIF API request is enabled, Informatica MDM Hub captures each SIF request invocation and response in the audit log. For more information, refer to the *Informatica MDM Hub Administrator Guide*.

State Management

If Hub state is specified in the request (see setRecordStates(ArrayList)), the parent Base Object of the specified package must have state management enabled. For more information regarding how to enable state management, refer to *Informatica MDM Hub Data Steward Guide* or *Informatica MDM Hub Administrator Guide*.
**Use Case**

This is the common scenario for using the GetMatchedRecords request:

- **Fetch the match candidates for a specified record, display them in a UI, and use the merge request to merge the match candidate the user selects**—After using GetMatchedRecords to retrieve candidate matches for a record, you can display the results in a UI for a user. The user can then select a candidate. Use merge to merge the two records.

**Related SIF Requests**

“Merge ” on page 68

**GetMergeHistory**

GetMergeHistory returns a tree representing the merge history for a specified base object record. The root node of the tree is the surviving rowid. The child nodes represent the records that have been merged into the surviving record. Each node contains the rowid and merge date of the record.

The request specifies a package and a key to identify the record. The response contains a tree of (rowid, merge date) pairs.

**Note:** You can configure GetMergeHistory requests according to a specific system when using the Hub Console Audit Manager to audit requests made by external applications. Once auditing for a particular SIF API request is enabled, Informatica MDM Hub captures each SIF request invocation and response in the audit log. For more information, refer to the *Informatica MDM Hub Administrator Guide*.

**Use Case**

This is the common scenario for using the GetMergeHistory request:

- **Fetch the list of merges from which the current record was formed**—get the list of merges, the product of whose cumulative changes have resulted in this record.

**Related SIF Requests**

“Merge ” on page 68, “Unmerge ” on page 77

**GetOneHop**

GetOneHop Hierarchy Manager request fetches information about the entities directly related to a specified group of entities in a specified HM configuration.

The request contains the HM configuration, a list of entity keys, and filtering criteria. The response contains lists of entity records and relationships, and a search token to use in fetching additional information.

**Note:** This request requires Hierarchy Manager. If you have not purchased, installed, configured Hierarchy Manager, then this request will fail.

**Use Case**

This is the common scenario for using the GetOneHop request:

- **Fetch one level of entities and relationships associated with a specific HM entity or entities** — If you have Hierarchy Manager and have populated it with data, you can use GetOneHop to get a single level of the entities and relationships associated with one or more entities.

**Related SIF Requests**

“GetEntityGraph” on page 61, “GetSearchResults ” on page 64
GetOrsList

GetOrsList retrieves a list of operational record stores (ORS) registered in the master database. This API does not require any additional parameters. As this API request operates on the master database, the ORS ID is not required.

Usage Example

The following example gets the list of all registered ORS databases:

```java
GetOrsListRequest request = new GetOrsListRequest();
GetOrsListResponse response = sipClient.process(request);
```

GetOrsMetadata

GetOrsMetadata retrieves the metadata for the current repository. In order to successfully export the repository, your ORS must be in a valid state. This SIF request provides the same functionality of the Export tool in Hub Console. For more information, see the *Informatica MDM Hub Metadata Manager Guide*.

Note: The created change list XML file can be very large (a number of megabytes or more).

GetSearchResults

GetSearchResults enables you to fetch additional sets of records retrieved as a result of any call that supports returning results one page at a time. These requests are:

- SearchQuery
- GetOneHop
- SearchHmQuery
- GetAssignedRecords
- Get Matched Records
- Search LookupValues

The input value is a search token, returned by a prior searchMatch request (or any of the search type SIF requests mentioned above), that identifies the search parameters, the index of the first record to return, and the number of records to return. You must use the search token within a limited period of time after you receive it. The default time limit for a search token is fifteen (15) minutes. To learn more about changing this limit, contact Informatica Global Customer Support.

Use Case

This is the common scenario for using the GetSearchResults request:

- **Fetch the next page of a set of records returned from a request that returns multiple pages**—After using any request that returns the first of multiple pages of a set of records, you can use getSearchResults repeatedly to get the subsequent pages.

Related SIF Requests

- “SearchQuery ” on page 74
- “SearchMatch ” on page 74
- “GetOneHop” on page 63
- “SearchHmQuery” on page 73
- “GetAssignedRecords” on page 59
- “GetMatchedRecords” on page 62
**GetSiperianObjectCompatibility**

GetSiperianObjectCompatibility obtains a checksum that represents the definition of the specified object in Informatica MDM Hub. This is used with ORS-specific APIs.

This API can be used to determine if an object on the server is compatible with a class in the client library for an ORS specific PACKAGE, MAPPING, CLEANSE_FUNCTION, or REMOTE_PACKAGE. ORS specific APIs and objects are generated in the Hub Console’s SIF Manager. This request should be used to determine if an objects definition on the server has changed since the last time ORS specific objects were generated. To resolve an incompatibility between a client object and its server counterpart is to regenerate the ORS specific objects. For more information on generating ORS specific objects, see the *Informatica MDM Hub Data Steward Guide*.

**Use Case**

This is the common scenario for using the GetSiperianObjectCompatibility request:

- **Fetch the checksum for an object to use when using ORS-specific APIs**—If you are using ORS-specific APIs, you can use GetSiperianObjectCompatibility.

**GetSystemTrustSettings**

GetSystemTrustSettings fetches the system-specific trust settings for the specified columns.

The request contains a list of columns and a system. The response contains a list of trust setting objects in the same order as the list of columns.

**Note:** You can configure GetSystemTrustSettings requests according to a specific system when using the Hub Console Audit Manager to audit requests made by external applications. Once auditing for a particular SIF API request is enabled, Informatica MDM Hub captures each SIF request invocation and response in the audit log. For more information, refer to the *Informatica MDM Hub Administrator Guide*.

**Use Case**

This is the common scenario for using the getSystemTrustSettings request:

- **Fetch the system-specific trust settings for a set of columns.**

**Related SIF Requests**

“GetTrustGraphData” on page 66, “GetTrustScore” on page 66

**GetTasks**

GetTasks enables you to retrieve lists of tasks and task details.

By default, GetTask searches for tasks with priority=NORMAL and status=OPEN. To search for tasks with other priorities and status values, specify the TaskPriority class and the TaskStatusEnum class values in the GetTask request.

When GetTasks is executed with the setDisablePaging() property set to false, paging is enabled and a search token is returned. You can use the search token as the input value. The default lifetime for a search token is 15 minutes, within which it must be used. To learn more about changing this limit, contact Informatica Global Customer Support.

GetTasks uses the setDisablePaging() property to disable paging. You can set the setDisablePaging() property to true, to disable paging to get added performance as appropriate.
Use Case
This is the common scenario for using the GetTasks request:

- **Fetch a set of Tasks that match the criteria specified in the request** — You can use GetTasks to retrieve a set of tasks that match the criteria you specify.

Related SIF Request
- GetTasks Lineage

GetTrustGraphData

GetTrustGraphData request provides the information needed to plot a trust decay curve.

The request contains a TrustSetting, which specifies the graph type, the time units, and other parameters of the required graph. The response contains a list of trust values and dates that define the graph.

**Note:** You can configure GetTrustGraphData requests only for "no system" when using the Hub Console Audit Manager to audit requests made by external applications. Once auditing for a particular SIF API request is enabled, Informatica MDM Hub captures each SIF request invocation and response in the audit log. For more information, refer to the *Informatica MDM Hub Administrator Guide*.

Use Case
This is the common scenario for using the GetTrustGraphData request:

- **In an application with a custom-designed UI, display a trust graph** — If you have a custom UI and must display a trust graph, use GetTrustGraphData to get the data on which the graph is based.

Related SIF Requests
- “GetSystemTrustSettings” on page 65, “GetTrustScore” on page 66

GetTrustScore

GetTrustScore computes the trust score for a specified column, based on the specified trust override. The column must be trust-enabled in the Schema Manager of the Hub console. The trust score (type float) of the Admin system will be returned.

The request contains a column UID and a key identifying the base object record. The response contains the trust score.

Use Case
This is the common scenario for using the GetTrustScore request:

- **Compute the trust score for a column** — If you are displaying a record, you can use GetTrustScore to display that information about a column.

Usage Example
The following example retrieves the trust score for column FIRST_NAME on base object C_CONTACT for the record with rowid = 3:

```java
GetTrustScoreRequest request = new GetTrustScoreRequest();
request.setColumnUid("COLUMN.C_CONTACT[FIRST_NAME]"); // Required
request.setRecordKey(RecordKey.rowid("3")); // Required
GetTrustScoreResponse response = (GetTrustScoreResponse) sipClient.process(request);
```

Related SIF Requests
- “GetSystemTrustSettings” on page 65, “GetTrustGraphData” on page 66
GetUnmergedRecordCount

GetUnmergedRecordCount reports the number of records that are not merged—either all such records or those assigned to the current user.

The request supplies the table and a boolean value that specifies whether or not to restrict the count to records assigned to the user. The response contains the number of unmerged records.

Note: You can configure GetUnmergedRecordCount requests only for "no system" when using the Hub Console Audit Manager to audit requests made by external applications. Once auditing for a particular SIF API request is enabled, Informatica MDM Hub captures each SIF request invocation and response in the audit log. For more information, refer to the Informatica MDM Hub Administrator Guide.

Use Case

This is the common scenario for using the GetUnmergedRecordCount request:

* In the data steward queue management screen in a custom UI, display the number of unmerged records
  —If you have a custom UI with a data steward queue management screen in a custom UI, you can use this request to display the number of unmerged records.

Related SIF Requests

"AssignUnmergedRecords" on page 51, "ReassignRecords" on page 71

Link

Link links two or more base object records using the specified groupRecordKey as the group ID. Unlike a merge operation, when records are linked, the original base object records continue to exist and the cross reference records are not directly associated with the grouping record. However, the cross reference records are grouped together in a link group with the rowid of the groupRecordKey specified in the LinkRequest. If the records specified for linking have been previously linked, then nothing is changed and the API returns a success message.

In order to be able to use the Link request on a base object, the base object must first be configured to be a link-style BO instead of a merge-style BO. This option can be configured in the Schema Manager of the Hub Console.

In order to use a link group, the "GetBvt" on page 60 request must be invoked. This retrieves the best version of truth (BVT) for the specified link group accounting for the combined cross reference records of all base object records in the link group.

Related SIF Requests

"GetBvt" on page 60, "Unlink" on page 77

ListSiperianObjects

ListSiperianObjects returns a list of objects of the specified type. An optional UID argument in the request restricts the list to objects that are children of the parent specified by the UID.

This request is used to obtain basic metadata for a list of Informatica MDM Hub objects that have been configured as a secure resource (using the Security Access Manager) accessible by the user. The request can be further restricted to objects that have a specific secure resource privilege enabled for the user. The metadata contains basic information such as SiperianObjectUID, display name, and description. To obtain the complete metadata, use "DeleteRelationship" on page 57.

Note: Only admin users can access private resources through SIF requests.
Use Case

This is the common scenario for using the ListSiperianObjects request:

- **For a given base object, get a list of packages**—If you have a custom UI, you can use this request to get a list of packages for a given base object so you can allow a user to choose which one they'll use for the current operation.

Related SIF Requests

“DeleteRelationship” on page 57

Merge

Merge merges two base object records, creating a single, consolidated base object record by merging all the XREF records from the two base objects.

When two records are merged, one is designated the source record, one is designated the target record. The request merges the source record into the target record. This means that after the merge the rowid for the combined record is that of the target record. All foreign keys pointing to the source record now point to the target record.

For example, there may be one base object record with the name “Alex Watson” and another with the name “Alexander Watson”; each base object record has its own set of cross reference records. These records are determined to represent the same person so the records are merged. The result is a single base object record that has all the cross reference records from the original two base object records. The consolidated value for each field in the record is determined by the trust configuration.

**Important:** When you merge two records, Informatica MDM Hub does not check the match status of the records, it just merges the records as you specify. Using this class, it is possible to merge two completely dissimilar records, resulting in a nonsense record. To learn more about merging, see the *Informatica MDM Hub Administrator Guide*.

**Note:** An alternate to Merge is “MultiMerge” on page 69 which can be used to merge two or more records in a single operation.

For more information, refer to the Merge Settings Tab on the Match/Merge Setup Details dialog in the Informatica MDM Hub Schema Manager, the *Informatica MDM Hub Administrator Guide*, and the *Informatica MDM Hub Data Steward Guide*.

State Management

When merging records in a base object with state management enabled, records in any state can be merged together. However, the following rules apply specifying a PENDING record as the target of a merge.

- If both records being merged together are PENDING, the records are merged as normal.
- If a PENDING target record is merged with an ACTIVE or DELETED source record, the target and source records are automatically swapped so the PENDING record becomes the source and the ACTIVE or DELETED record becomes the target.

For more information on state management, see the *Informatica MDM Hub Data Steward Guide* and *Informatica MDM Hub Administrator Guide*.

Use Case

This is the common scenario for using the merge request:

- **Merge used with “GetMatchedRecords” on page 62** — You can use GetMatchedRecords to get a list of match candidates for a specified record. You can then display that list in a UI. If the user selects one of the candidate records, you can use merge to merge the two records.
MultiMerge

MultiMerge merges multiple base object records that have been identified as representing the same object and allows specifying the field level overrides for the merged record. MultiMerge is a more generic form of the "Merge" request and should be used for merging groups of records. Merge API should be used for pair-wise merges.

For example, there may be multiple base object records with the same account number "1234567" and account type "Personal". Each base object record has its own set of cross reference records. When these records are merged with a call to the MultiMerge request, the result is a single BO record with that has the cross references from all the merged base object records. The consolidated value for each field in the merged record is either determined through the survivorship rules based on the cross references of the records that are being merged or they are specified through the override values in the API.

PromotePendingXrefs

PromotePendingXref promotes or flags for promotion the XREF records specified in the request.

State Management

Promote means to change the state of a record from PENDING to ACTIVE. When the flagForPromote option is set, then this API request will queue the specified xref records for promotion using the next run of the PROMOTE batch process. Otherwise, the request will immediately promote the specified xref records from PENDING to ACTIVE. Here's the behavior of this request based on various XREF states:

- ACTIVE and DELETED records will return an error.
- PENDING records will be made ACTIVE.

Usage Example

The following example immediately promotes the "FIRST_NAME" and "LAST_NAME" fields for the XREF record with sourceKey=1234 and system=CRM in the package CUSTOMER_UPDATE. If the XREF record is ACTIVE or DELETED, an error will be returned. If the XREF record is PENDING, it will be made ACTIVE.

```java
PromotePendingXrefsRequest request = new PromotePendingXrefsRequest();
ArrayList columnNames = new ArrayList();
columnNames.add("FIRST_NAME");
columnNames.add("LAST_NAME");
request.setColumnNames(columnNames); // Optional
XrefKey xrefKey = new XrefKey();
xrefKey.setSourceKey("1234");
xrefKey.setSystemName("CRM");
ArrayList xrefKeys = new ArrayList();
xrefKeys.add(xrefKey);
request.setXrefKeys(xrefKeys); // Required
request.setSuperiorObjectUID("PACKAGE.CUSTOMER_UPDATE"); // Required
PromotePendingXrefsResponse response = (PromotePendingXrefsResponse) sipClient.process(request);
```

The following example flags the XREF record with sourceKey=1234 and system=CRM in the package CUSTOMER_UPDATE for promotion the next time the Promote batch process is run.

```java
PromotePendingXrefsRequest request = new PromotePendingXrefsRequest();
XrefKey xrefKey = new XrefKey();
xrefKey.setSourceKey("1234");
xrefKey.setSystemName("CRM");
ArrayList xrefKeys = new ArrayList();
xrefKeys.add(xrefKey);
request.setFlagForPromote(true); // Optional
```
Put

Put adds a new row or updates an existing row of a base object table. The package must be put-enabled. To learn more about creating put-enabled packages, see the Informatica MDM Hub Administrator Guide.

A record key identifies the record to update. The record can contain all or a subset of the package columns. Set the record key’s systemName field to identify the system putting the data.

You can specify a change date in the request. If you do not specify a date, Informatica MDM Hub uses the current date and time as the change date.

Note: Informatica MDM Hub rounds all time values to the nearest second.

The request can specify a value to be used as the source key for a new record. This can be useful for keyless systems.

The response contains the record key (the rowid, sourceKey, and system) from the Put request. It also contains an action type string to use as input to a tokenize request to update tokens before the next batch process updates them.

Note: If a value is not provided for a non-nullable column when using the put request, the database will throw an error and the request will fail.

If you are putting a record from a keyless system, you can request that Informatica MDM Hub generate a unique PKEY_SRC_OBJECT for the new object.

If you are inserting a base object, and request a new pkey to be generated, the key generator will generate a value for the pkey that will be passed to the put part of this request.

Important: If you use the Put request on a package that is based on a base object for which tokenization is set on, the put request fails.

Important: Use the "AddRelationship" on page 51 and "UpdateRelationship" on page 79 requests to add or update relationship records. Using the Put request or an ORS-specific put request to update relationships can lead to improperly formed relationship records.

Note: Both Put and Cleanse Put calls process null values; for example when no value is specified for a field, it will be set to null.

Metadata Manager Promote Operation

When you use Metadata Manager to promote changes to design objects such as HMRelationshipType, the PUT API is called. The PUT API in turn invokes the POST_LOAD user exit, which may have user-defined procedures that can result in promotion issues or changes to Hub Server data. The bypassPostLoadUE parameter has therefore been added to the PUT stored procedure to specify whether the POST_LOAD user exit must be called or not. The default value is false, which means that the POST_LOAD user exit will be called. Metadata Manager is the only calling program that should bypass the POST_LOAD user exit. You must not modify the value of the bypassPostLoadUE parameter.

State Management

For packages with state management enabled, state can be specified when inserting a new record. State management is enabled on a package by enabling state management on the parent base object of the package in
the Hub Console. Packages with state management enabled will have a column called HUB_STATE_IND. When inserting a new record, a value can be specified for the HUB_STATE_IND column that will be used as the initial state of the new record.

The possible values for HUB_STATE_IND are 1, 0, and -1 corresponding to ACTIVE, PENDING, and DELETED states respectively. The Put API cannot be used to change the state of an existing record. If a value is provided for the HUB_STATE_IND column when updating an existing record, the Put API will throw an exception. To change the state of an existing record, refer to the following classes: “Delete” on page 56, “Restore” on page 72, and “PromotePendingXrefs” on page 69.

Transaction Support

When executed within an EJB context, this request can be part of a transaction with other requests. If there is a failure in any of the requests within a transaction, the entire transaction is rolled back.

Validation Rules: Load Process vs. SIF Put Requests

The execution sequence for validation rules differs between the load process (as described in the Informatica MDM Hub Administrator Guide) and Put API calls invoked by external applications using SIF. For Put requests, validation rules are executed in order of decreasing downgrade percentage. The simple reason is that it’s faster when working with a single record (Put API) to execute in descending order. When executing through a batch of records, it's better to execute in ascending order.

Use Cases

These are the common scenarios for using the Put request:

- **Put used in combination with cleanse** — The most common use of cleanse is when the individual fields are cleansed before the record is passed on to Put.

- **Put used in combination with tokenize** — A request to put, followed by a request to “Tokenize” on page 77 inserts or updates the record and encodes it for matching.

Usage Example

The following example updates a record with ROWID_OBJECT key 782 using the package ADDRESS_UPDATE.

```java
PutRequest request = new PutRequest();
request.setRecordKey(RecordKey.rowid("782", "SALES"));
Record record = new Record();
record.setSiPerianObjectOfTypeSiPerianObjectUid(SiPerianObjectType.PACKAGE.makeUid("ADDRESS_UPDATE"));
record.setField( new Field("ADDRESS_LINE1", "123 Main St.") );
record.setField( new Field("CITY", "AnyTown") );
PutResponse response = (PutResponse) sipClient.process(request);
```

Related SIF Requests

“CleansePut” on page 54, “Tokenize” on page 77

ReassignRecords

ReassignRecords reassigns the specified records assigned for manual merge evaluation to another user.

The new user will now be responsible for these records.

Use Case

This is the common scenario for using the ReassignRecords request:

- **In a custom UI, allow data stewards to reassign the records in their queue**—If you have a custom UI, in the screen for managing data steward’s queues, you might have a button that uses this request. This would allow data stewards to reassign the records in their queue.
RegisterUsers

RegisterUsers enables an application to register selected users from the enterprise’s authentication system (for example, LDAP) with Informatica MDM Hub. Then Informatica MDM Hub can use its existing access control capabilities to manage tasks like role assignments.

The application provides a list of user names, and Informatica MDM Hub fetches their information from the external system. Informatica MDM Hub ignores additional registrations of the same user profile from the external authentication system. However, it reports errors if the username is already registered using a different, or no, external profile, or if the name does not exist in the external authentication system.

Informatica MDM Hub registers the users within a transaction. If an error occurs, it rolls back all changes.

Provisioned users can be grouped and assigned Informatica MDM Hub security roles using the Informatica MDM Hub Administration Console. For more information, see the Informatica MDM Hub Administrator Guide.

Automatically provisioned users can be removed from the Informatica user database either using the Informatica Administration Console or using "UnregisterUsers" on page 78.

Transaction Support

When executed within an EJB context, this request can be part of a transaction with other requests. If there is a failure in any of the requests within a transaction, the entire transaction is rolled back.

Use Case

This is the common scenario for using the RegisterUsers request:

- **Checking external authentication**—Before you start a logical unit of work, check to see what the user is authorized to do.

Related SIF Requests

"Authenticate" on page 53

ResetBatchGroup

ResetBatchGroup finds the last execution status of the given batch group, and if its status is failed, sets it to incomplete. To learn more about batch groups, see the Informatica MDM Hub Administrator Guide.

Use Case

This is the common scenario for using the ResetBatchGroup request:

- **Resetting a batch group after** “GetBatchGroupStatus” on page 60 returns and unsuccessful status

Related SIF Requests

"ExecuteBatchGroup" on page 58, "GetBatchGroupStatus" on page 60

Restore

Restore reinstates the specified XREF record(s) in the Hub. Restore changes the state of records from DELETED to ACTIVE state. If an attempt is made to restore an active or pending record, an error is returned. After an XREF record is restored, the state of the parent BO record will be active.
SearchHmQuery

SearchHmQuery is used to search HM Entities or Relationships. The filter, aggregate and sort criteria can reference any columns in the Display Packages associated with Entity Type / Relationship Type in the search request. The criteria can use any operators supported by the underlying database.

The value stored in GETLIST_LIMIT column of CMX_SYSTEM.C_REPOS_DATABASE table for the ORS determines the maximum number of records that can be returned. GetSearchResultsRequest can be used to get subsequent pages of records.

The request contains the HM configuration, the type of the entity or relationship sought, and an SQL specification of the query. The response contains the sought records and a search token to use to fetch additional data.

Note: This request requires Hierarchy Manager. If you have not purchased, installed, configured Hierarchy Manager, then this request will fail.

Use Case

This is the common scenario for using the SearchHmQuery request:

- Search for specific HM entity or entities — If you have Hierarchy Manager and have populated it with data, you can use SearchHmQuery to search for entities and relationships associated with one or more entities.

Related SIF Requests

“SearchRequestBase” on page 75, “GetSearchResults” on page 64, “SearchHmQuery” on page 73

SearchLookupValues

SearchLookupValues request searches for lookup values that match a given lookup display name (lookup code description).

This request is typically used with foreign key columns that have a large number of possible values. The request includes a lookup column, a lookup display value to search for, and a comparison operator. It also includes a record whose fields contain the lookup information. In each field, the name is the lookup (foreign key) value, and the value is the lookup display name. For detailed information on configuring relationships and lookups in the Informatica MDM Hub, see the Informatica MDM Hub Administrator Guide.

This request allows search criteria to be specified on the lookup display name compared to “GetLookupValues” on page 62 which retrieves all lookup values for a lookup column.
A system parameter determines the maximum number of records that can be returned. Use “GetSearchResults” on page 64 to get subsequent sets of records.

**Use Case**

This is the common scenario for using the SearchLookupValues request:

- **Search for lookup values**—In a custom UI, you can use SearchLookupValues to find that value from amongst the available lookup values.

**Related SIF Requests**

“GetLookupValue” on page 61, “GetLookupValues” on page 62

### SearchMatch

SearchMatch request searches for records in a package based on match column and rule definitions. The input for the search is one or more package or mapping records. The request uses the columns in these records to generate match columns (including inter-table match columns) that are used by the match server to find match candidates.

By default, the request generates all possible match columns from the input records, but you can specify a smaller number of columns by providing the column identifiers.

**Note:** SearchMatch, unlike the Match batch process, does not create match candidates in the ORS. When the Match batch process is run, the Match candidates are created in the ORS and Data Steward API requests such as GetMatchedRecords can be used to retrieve them. In contrast, SearchMatch only returns a list of match candidates. The information in this list can then be used to perform Merge, MultiMerge or Link request. In order to perform merging or linking for the matching records, you have to invoke the “Merge” on page 68, “MultiMerge” on page 69, or “Link” on page 67 requests.

A system parameter determines the maximum number of records that can be returned. Use “GetSearchResults” on page 64 to get subsequent sets of records.

**Note:** For information on performing exact matches on fuzzy base objects, see “Exact Matches on Fuzzy Base Objects” on page 19.

**Use Case**

This is the common scenario for using the SearchMatch request:

- **Fetch the possible matches in a given package.**

**Related SIF Requests**


### SearchQuery

SearchQuery searches for records in a package based on an SQL condition clause. The condition clause can reference any columns in the package and can use operators supported by the target database.

A system parameter determines the maximum number of records that can be returned. Use the “GetSearchResults” on page 64 request to get subsequent sets of records.

**Retrieving Large Record Sets**

For information on controlling the number of records to be returned by the query and setting the data page size for paging support, see “SearchRequestBase” on page 75.
Case Sensitivity

Under normal conditions, the SearchQuery API is case sensitive. Any filter criteria specified for the request must be in the same case as in the ORS in order for the records to be found. However, the CASE_INDICATOR column in C_REPOS_TABLE can be used to control case sensitivity of SearchQuery. Possible values for this indicator are: UPPER, LOWER and NULL. When specified, the value in this column indicates that all data in the corresponding table is in the specified case. The setting of this causes filter criteria to be automatically converted to the appropriate case. Function-based queries are not required to implement case insensitive searches. In practice, the following behavior can be expected depending on the setting of CASE_INDICATOR:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UPPER</td>
<td>The where clause and any parameters of the query are converted to upper case prior to executing the request. This assumes that all of the data in the package in the request is stored in upper case.</td>
</tr>
<tr>
<td>LOWER</td>
<td>The where clause and any parameters of the query are converted to lower case prior to executing the request. This assumes that all of the data in the package in the request is stored in lower case.</td>
</tr>
<tr>
<td>NULL</td>
<td>The query specified is executed as is with no case conversions. No assumptions are made about the case of data in the package in the request.</td>
</tr>
</tbody>
</table>

Use Case

This is the common scenario for using the SearchQuery request:

- **Search for records in a package**—In a custom UI, use SearchQuery to allow a data steward to find a particular record.

Related SIF Requests

"SearchHmQuery" on page 73, "GetSearchResults " on page 64, "Get" on page 58, "SearchMatch " on page 74

SearchRequestBase

SearchRequestBase is the base class for search requests (SearchQuery and SearchMatch) with parameters for paging and sorting.

Paging Support

The parameters for the paging mechanism to return large result sets are as follows:

- **Maximum number of records returned**—This parameter can be specified at the ORS level and it is stored in the CMX_SYSTEM.C_REPOS_DATABASE.GETLIST_LIMIT parameter. This limit takes precedence over the values specified using setRecordsToReturn(int). The search queries will be limited to the minimum value of the two for any search request. For SearchMatchRequest API, there are also additional parameters that can be specified on the Cleanse/Match server to control the number of matches the Hub will attempt before returning the results.

- **Number of records**—setRecordsToReturn(int). When paging is enabled this parameter specifies the size of the first page of data returned by the search API. Subsequent pages can be returned using the GetSearchResultsRequest API. Alternatively, for requests that have paging disabled this method would specify the limit of the total number of rows returned by the search API.

- **The paging mechanism is enabled by default**—It can be disabled using the setDisablePaging() methods of APIs such as SearchQuery and SearchMatch that support paging. If the setDisablePaging() method is set to false, then a searchToken, which can be used for a search, is returned.
• **The search token is deleted**—After a period of inactivity longer than the `sif.search.result.query.timeToLive.seconds` setting specified in the Hub Server properties. GetSearchResults calls made after the token is expired would result in an error.

For more information, refer to the SIF API Javadocs.

### SearchResponseBase

SearchResponseBase is the base class for search responses.

Each response contains a list of records, a search token to use to fetch more results, and an optional count of matching records.

For more information, refer to the SIF API Javadocs.

### SetPassword

SetPassword request changes the password for this user. The existing password must be specified in the request as well as the new password. Passwords are specified as Password objects. The password specified must to the password policy configured in the Hub.

**Use Case**

This is the common scenario for using the SetPassword request:

- **Allow a Informatica MDM Hub administrator to set a password for a user within Informatica MDM Hub** — In an application for Informatica MDM Hub administrators, you can include functionality to allow the administrator to change passwords for the Informatica MDM Hub users.

### SetRecordState

SetRecordState enables a client application to assign one of a predefined set of state values to a specified set of base object records. This value is stored in the CONSOLIDATION_IND column. The consolidation indicator can be one of the following values:

<table>
<thead>
<tr>
<th>Indicator Value</th>
<th>State Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CONSOLIDATED</td>
<td>This record has been determined to be unique and represents the best version of the truth.</td>
</tr>
<tr>
<td>2</td>
<td>UNMERGED</td>
<td>This record has gone through the match process and is ready to be consolidated.</td>
</tr>
<tr>
<td>3</td>
<td>QUEUED_FOR_MATCH</td>
<td>This record is a match candidate in the match batch that is being processed in the currently-executing match process.</td>
</tr>
<tr>
<td>4</td>
<td>NEWLY_LOADED</td>
<td>This record is new (load insert) or changed (load update) and needs to undergo the match process.</td>
</tr>
<tr>
<td>9</td>
<td>ON_HOLD</td>
<td>The data steward has put this record on hold until further notice. Any record can be put on hold regardless of its consolidation indicator value. The match and consolidate processes ignore on-hold records. For more information, see Informatica MDM Hub Data Steward Guide.</td>
</tr>
</tbody>
</table>

**Note:** This request cannot be used within a transaction.
Use Case

This is the common scenario for using the SetRecordState request:

- **In a client application, explicitly set the state of a record**—In an application, allow a user to explicitly set the state of a given record.

**Tokenize**

Tokenize enables you to generate the match keys that are used by the match engine for fuzzy matches. Once generated, these match keys are used by the match and merge batch processes, as well as by the “SearchMatch” on page 74 request. The Merge request does not use these keys.

You can choose to always perform batch tokenization after the load completes or delay this process by going into the schema tool, in the Informatica MDM Hub Console, and clicking on the advanced base object properties.

If you request tokenize for the same record twice, Informatica MDM Hub only tokenizes the row on the first request. The second request doesn’t fail. It succeeds and reports that it tokenized zero records.

**Use Cases**

These are the common scenarios for using the Tokenize request:

- **Tokenize used with “Put” on page 70** — Since the functionality for inserting the record into the database resides within the database itself and the functionality for tokenization resides in the Cleanse Match Server, you must do these functions separately. Use Put which inserts or updates the record in the package. Then, call tokenize to generate keys for fuzzy matching.

- **Tokenize used with “CleansePut” on page 54** — Since the functionality for inserting the record into the database resides within the database itself and the functionality for tokenization resides in the Cleanse Match Server, you must do these functions separately. Call CleansePut which cleanses the data and then inserts or updates the record in the package. Then, call Tokenize to generate keys for fuzzy matching.

**Related SIF Requests**

“CleansePut” on page 54, “Put” on page 70

**Unlink**

Unlink decouples two or more base object records with the group ID specified in the groupRecordKey field. The records being unlinked must have been previously linked using the Link API.

**Unmerge**

Unmerge unmerges two rows in a base object. This is the same functionality that Data Stewards use when unmerging a record using the Merge Manager. If, for example, a merge of two records was done in error. This request restores all foreign keys updated in the merge.

Unmerging can take a long time to complete, so you may want to run the unmerge asynchronously. You cannot use unmerge within a transaction. For more information about unmerging and merging, refer to the Informatica MDM Hub Administrator Guide.

If the request is unsuccessful, the program throws a Informatica request exception. If the request is successful, then the response also indicates that the unmerge succeeded.

**Note:** You can configure Unmerge requests according to a specific system when using the Hub Console Audit Manager to audit requests made by external applications. Once auditing for a particular SIF API request is enabled, Informatica MDM Hub captures each SIF request invocation and response in the audit log. For more information, refer to the Informatica MDM Hub Administrator Guide.
Cross-reference Added Directly to a Base Object

Typically when a new cross-reference is inserted, a new base object record is created for it. But Put or CleansePut APIs and the “Load By ROWID” batch job can also add a cross-reference record directly to a base object. This means that the cross-reference record was never the only cross-reference for a base object record. When such a cross-reference record is identified to be unmerged it is deleted from the system and there is no independent base object record to reinstate for it.

Linear Unmerge

No attention is paid to the process by which the records were originally merged. The cross-reference for sourceKey 2 is removed from the base object and the base object record for sourceKey 2 is reinstated. The resulting base object records are as follows:

- ROWID_OBJECT=1 has one cross-reference record with sourceKey 1.
- ROWID_OBJECT=2 has two cross-reference records with sourceKeys 2 and 3.

Tree Unmerge

The process by which the records were originally merged determines the outcome. At the time record 2 was merged into record 1, record 3 had already been merged into 2. So the cross-references for sourceKeys 2 and 3 are removed from the base object and the base object record for sourceKey 2 is reinstated. The resulting base object records are as follows:

- ROWID_OBJECT=1 has one cross-reference record with sourceKey 1.
- ROWID_OBJECT=2 has two cross-reference records with sourceKeys 2 and 3.

Note: that in both cases, the consolidated field values in the base object record are recalculated after the unmerge.

Cascade Unmerge

Unmerge performs a cascade unmerge if this feature is enabled for this base object in the Schema Manager in the Hub Console. With cascade unmerge, when records in the parent object are unmerged, Informatica MDM Hub also unmerges affected records in the child base object.

Use Case

This is the common scenario for using the Unmerge request:

- In a custom UI, use unmerge to allow a data steward to manually unmerge records

Related SIF Requests

“Merge ” on page 68

UnregisterUsers

UnregisterUsers enables an application to unregister selected users from Informatica MDM Hub. The application sends a list of user names, presumably representing users in the enterprise’s authentication system (for example, LDAP).

The application provides a list of user names, and Informatica MDM Hub removes them. Informatica MDM Hub ignores unregistrations of users that are not registered in Informatica MDM Hub.

Informatica MDM Hub unregisters the users within a transaction. If an error occurs, it rolls back all changes.

Transaction Support

When executed within an EJB context, this request can be part of a transaction with other requests. If there is a failure in any of the requests within a transaction, the entire transaction is rolled back.
Use Case

This is the common scenario for using the UnregisterUsers request:

- **Bulk unregistering with Informatica MDM Hub** — Based on external authentication information, you can use unregistreUsers to bulk unregister users.

Related SIF Requests

“RegisterUsers” on page 72

UpdateRelationship

UpdateRelationship Hierarchy Manager request updates a Relationship between two Entities. The existing relationship record is updated if the Start Date, End Date, or custom columns for the Relationship record are modified. If the update request changes the Hierarchy, Relationship Type, or one or both Entities in the Relationship, the current Relationship record is deleted (see “DeleteRelationship” on page 57) and a new Relationship record is added (see “AddRelationship” on page 51). When a new Relationship record is added, the RecordKey returned in the UpdateRelationshipResponse will be different from the one specified in the UpdateRelationshipRequest.

**Note:** This API request applies to Hierarchy Manager. If you have not purchased, configured, and populated Hierarchy Manager, this request will fail.

The request identifies the HM configuration and hierarchy, the relationship type, the records, and a number of optional parameters. The response contains the record key for the updated relationship. Informatica MDM Hub infers the types of the entities being related, and thus the base objects containing those entities, from the relationship type.

**Adding a New Relationship for a Foreign Key Relationship Type**

Use UpdateRelationshipRequest instead of AddRelationshipRequest to add a new Relationship for a Foreign Key Relationship Type because adding that Relationship really involves updating an existing record in the FK Relationship Base Object. For example, if there is a FK Relationship Base Object C_PERSON with the following columns:

- Rowid Object: Primary Key for the FK Relationship and the Entity.
- Rowid Company: FK that refers to the records in C_COMPANY Entity Base Object. The value in this column has a non-null when there a HM FK Relationship between C_PERSON and C_COMPANY. The column value is set to null when the Relationship between a Company and a Person is deleted.
- Any other columns needed for “Person” Entity Type and “Person To Company” FK Relationship Type.

The Put Package associated with the “Person To Company” FK Relationship Type would have the following columns:

- Rowid Object (C_PERSON.Rowid_Object): Primary Key for the FK Relationship
- Rowid BO1 (C_PERSON.Rowid_Object): BO1 in the FK Relationship
- Rowid BO2 (C_PERSON.Rowid_Company): BO2 in the FK Relationship

In other words either rowid BO1 or rowid BO2 in the Put Package maps to the Rowid Object column in the Base Object. Updating the FK column (Rowid BO2 / C_PERSON.Rowid_Company in the example) to a non-null value is equivalent to adding a new FK Relationship. Also note that the RecordKey specified in setRecordKey() and setBo1RecordKey() in this example would be the same.

**Note:** UpdateRelationshipRequest cannot be used to modify Relationship Type if the old or the new Relationship Type is a FK Relationship Type. To do that, use DeleteRelationshipRequest to delete the old Relationship. Then use UpdateRelationshipRequest if the new Relationship is a FK Relationship Type or AddRelationshipRequest if the new Relationship is not a FK Relationship Type.
Use Case

This is the common scenario for using the UpdateRelationship request:

- **Update a relationship between two HM entities** — If you have Hierarchy Manager and have populated it with entities, you can use the UpdateRelationship request to modify a relationship between two entities.

Related SIF Requests

"AddRelationship" on page 51, "DeleteRelationship" on page 57

**ValidateChangeList**

ValidateChangeList validates a change list against the current ORS. It applies the specified change list to the current repository, executing all of the changes in simulation mode (the repository is not modified), and returning a list of errors, if applicable.

**ValidateMetadata**

ValidateMetadata request validates the metadata for the current repository and returns a list of issues.

**Note:** A successful ValidateMetadata response will not return a message. ValidateMetadata only returns a message if metadata issues are found. You must iterate through the list of messages to determine:

- whether the ValidateMetadata request ran without exceptions
- if there are any metadata issues.
INDEX

A
AcceptUnmatchedRecordsAsUnique request 50
access protocols
using SIF 6
addRelationship operation 51
AddRelationship request 51
ApplyChangeList
rollbackStrategy field 51
applyChangeList call 51
ApplyChangeList request 51
AssignUnmergedRecords request 51
asynchronous requests
making 31
asynchronous SIF service invocations
run-time processing 35
Audit request 52
Authenticate request 53

B
base objects, fuzzy
exact matches 19
Batch Group APIs, about 46
Batch Group Services, about 11
build_war macro 22

C
CanUnmergeRecords request 53
cascade unmerge
Unmerge 77
Cleanse 24
cleanse request 54
CleansePut
state management 55
cleansePut request 25, 54
ClearAssignedUnmergedRecords request 56
com.siperian.sif.client package 19
com.siperian.sif.message package 19
composite services
about 38
consolidation
indicator 76
state 76
content metadata
DELETED_XREF 58
HISTORY 58
PENDING_XREF 58
RAW 58
XREF 58
XREF_HISTORY 58
createChangeList call 56
CreateChangeList request 56

D
Data APIs, about 46
Data Retrieval APIs, about 46
Data Retrieval Services, about 10
Data Services, about 11
Data Steward APIs, about 46
Data Steward Services, about 10
Data Update / Insert APIs, about 46
Data Update / Insert Services, about 11
debug log 19
Delete
state management 56
DELETED_XREF
content metadata 58
DeleteRelationship request 57
DeleteRequest 56
DescribeSiperianObject request 57

E
Eclipse 8
ExecuteBatchGroup request 58
external applications
interacting with Informatica MDM Hub 2

F
foreign key relationship type
adding new relationship, UpdateRelationship 79
fuzzy base objects
exact matches 19

G
Get 26
Get request 58
GetAssignedRecords request 59
GetBatchGroupStatus request 60
GetBvt
state management 60
GetBvt request 60
GetEntityGraph request 61
GetLookupValue request 61
GetLookupValues request 62
GetMatchedRecords
state management 62
GetMatchedRecords request 62
GetMergeHistory request 63
GetOneHop request 63
GetOrsList request 64
getOrsMetadata call 64
GetOrsMetadata request 64
GetSearchResults request 64, 65
GetSiperianObjectCompatibility request 65
getSystemTrustSettings request 65
GetTrustGraphData request 66
GetTrustScore request 66
GetUnmergedRecordCount request 67
GetSystemTrustSettings request 66

H
HISTORY
content metadata 58

I
incremental data loads, about 2
index.html 15
Informatica MDM Hub
external applications, how to interact with 2
real-time processing 2

J
Java archive (JAR) files
tools.jar 22
Java compilers 22
Javadoc
about 15
JMS Event Messages
about 31
JMS Message Queues for Asynchronous SIF Invocations, about 34

L
lib directory 22
linear unmerge
Unmerge 77
Link request 67
ListSiperianObjects request 67
Load Process vs. SIF Put
validation rules 70

M
Merge
state management 68
Merge request 68
Merge Workflow APIs, about 46
Merge Workflow Services, about 11
Metadata APIs, about 46
metadata management API
using 36
Metadata Management APIs, about 46
Metadata Manager APIs, about 46
Metadata Services, about 12
Miscellaneous APIs, about 46
MultiMerge request 69

O
Operational Record Store (ORS) database, about 1
ORS-specific APIs
classes 23
generating 22

P
paging support
SearchRequestBase 75
PENDING_XREF
content metadata 58
process method, about 13
PromotePendingXref request 69
PromotePendingXrefs
state management 69
proxies 8
Put
state management 70
transaction support 70
Put request 70

R
RAW
content metadata 58
real-time processing, about 2
ReassignRecords request 71
RecordKey
about 21
Records
about 20
RegisterUsers
transaction support 72
RegisterUsers request 72
ResetBatchGroup request 72
Restore request 72
rollbackStrategy field (ApplyChangeList) 51

S
SAM
using with SIF 35
schema
defines the database tables in an ORS 1
SearchHmQuery request 73
SearchLookupValues request 73
SearchMatch request 74
SearchMatchColumn 28
SearchMatchRecord 29
searchQuery
case sensitivity 74
retrieving large record sets 74
SearchQuery 30
searchQuery operation 30
searchQuery request 74
SearchRequestBase
paging support 75
SearchRequestBase request 75
SearchResponseBase request 76
Security Access Manager (SAM)
using with SIF 35
Services Development Kit (SDK), about 6
Services Integration Framework (SIF) 3
SetPassword request 76
SetRecordState request 76